

Masterclass

mc

Bernhard Korte
Jens Vygen

Kombinatorische Optimierung

Theorie und Algorithmen

3. Auflage



Springer Spektrum

Masterclass

Weitere Bände in der Reihe <http://www.springer.com/series/8645>

Bernhard Korte · Jens Vygen

Kombinatorische Optimierung

Theorie und Algorithmen

3. Auflage

Bernhard Korte
Jens Vygen

Universität Bonn
Deutschland

Übersetzt von Ulrich Brenner und Rabe von Randow, Universität Bonn, Deutschland

Masterclass

ISBN 978-3-662-57690-8 ISBN 978-3-662-57691-5 (eBook)
<https://doi.org/10.1007/978-3-662-57691-5>

Die Deutsche Nationalbibliothek verzeichnetet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Mathematics Subject Classification (2010): 90C27, 68R10, 05C85, 68Q25

Springer Spektrum

Deutsche Übersetzung der 6. englischen Originalausgabe, erschienen bei Springer-Verlag GmbH, 2018
© Springer-Verlag GmbH Deutschland, ein Teil von Springer Nature 2008, 2012, 2018

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags.
Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen. Der Verlag bleibt im Hinblick auf geografische Zuordnungen und Gebietsbezeichnungen in veröffentlichten Karten und Institutionsadressen neutral.

Verantwortlich im Verlag: Annika Denkert

Springer Spektrum ist ein Imprint der eingetragenen Gesellschaft Springer-Verlag GmbH, DE und ist ein Teil von Springer Nature

Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany

Vorwort zur dritten deutschen Auflage

Wir freuen uns sehr, dass mit dieser Neuauflage auch die deutsche Ausgabe wieder auf dem neuesten Stand ist und der aktuellen 6. englischen Auflage entspricht. Wir danken Ulrich Brenner und Rabe von Randow sehr herzlich für ihre gründliche und hervorragende Übersetzung.

Bonn, im Mai 2018

Bernhard Korte und Jens Vygen

Vorwort zur zweiten deutschen Auflage

Diese zweite deutsche Auflage unseres Buches ist inhaltlich identisch mit der soeben erschienenen fünften englischen Auflage. Rabe von Randow hat wieder die Übersetzung ins Deutsche vorgenommen; ihm möchten wir an dieser Stelle besonders danken. Dank gebührt auch Ulrich Brenner für die kritische Durchsicht.

Diese deutsche Neuauflage ist insgesamt an vielen Stellen ergänzt, aktualisiert und verbessert. So wird dieses Buch hoffentlich weiterhin gute Dienste beim Lehren und Lernen der Kombinatorischen Optimierung im deutschsprachigen Raum leisten.

Bonn, im Dezember 2011

Bernhard Korte und Jens Vygen

Vorwort zur ersten deutschen Auflage

Wir freuen uns, dass unser Buch nun auch in unserer Muttersprache erscheint.

Dies ist eine Übersetzung der vierten Auflage der englischen Originalausgabe, in die einige seit dem Erscheinen der 4. Auflage vorgenommenen Ergänzungen, Verbesserungen und Aktualisierungen bereits eingearbeitet sind.

Wir danken Rabe von Randow für die ausgezeichnete Übersetzung und hoffen, dass sie zu einer noch stärkeren Verbreitung unseres Buches an Universitäten des deutschsprachigen Raumes beiträgt.

Bonn, im Mai 2008

Bernhard Korte und Jens Vygen

Vorwort zur sechsten englischen Auflage

Nach sechs Jahren war es wieder Zeit für eine neue Auflage. Neben Aktualisierungen, neuen Aufgaben und einigen Korrekturen enthält sie das folgende neue Material:

Abschnitt 7.4 widmet sich den seichten leichten Bäumen. Abschnitt 14.6 enthält den neuen 2-Approximationsalgorithmus für die Maximierung submodularer Funktionen. Abschnitt 17.5 erörtert den Nemhauser-Ullmann-Algorithmus und die geglättete Analyse. In Abschnitt 20.3 zeigen wir einen $(\ln 4 + \epsilon)$ -Approximationsalgorithmus für das Steinerbaum-Problem. Schließlich enthält Abschnitt 20.7 das VPN-Theorem. Es gibt außerdem kleinere Ergänzungen, z.B. zur Ganzzahligkeitslücke in Abschnitt 5.1 und zur Kern-Reduktion in Abschnitt 15.7.

Wir möchten Maxim Babenko, Steffen Böhmer, Ulrich Brenner, György Dósa, Michael Etscheid, Jean Fonlupt, Michel Goemans, Stephan Held, Stefan Hougardy, Jochen Könemann, Solomon Lo, Jens Maßberg, Neil Olver, Dieter Rautenbach, Heiko Röglin, Jan Schneider, Sophie Spirkl und Uri Zwick für Kommentare zur vorigen Auflage bzw. für das Korrekturlesen der neuen Teile danken.

Wir hoffen, dass mit dieser neuen Auflage das Buch viele Jahre lang für Forschung und Lehre nützlich bleiben wird.

Bonn, September 2017

Bernhard Korte und Jens Vygen

Vorwort zur fünften englischen Auflage

Als wir an der ersten Auflage dieses Buches schrieben, vor mehr als zehn Jahren, strebten wir zwei Ziele an: Es sollte als fortgeschrittenes Lehrbuch nützlich sein, aber auch als Nachschlagewerk für die Forschung. Mit jeder neuen Auflage müssen wir entscheiden, wie wir das Buch weiter verbessern können. Natürlich ist es aber immer weniger möglich, das wachsende Gebiet umfassend zu beschreiben.

Berücksichtigen wir alles, was uns gefällt, so könnte das Buch nicht mehr in einem Band erscheinen. Da es vielfach für Vorlesungen benutzt wird, mittlerweile sogar manchmal im Bachelorstudium, haben wir beschlossen, dass es vielleicht sinnvoller ist, einige klassische Themen hinzuzufügen als eine Auswahl der neuesten Resultate.

Neu hinzugekommen sind in dieser Auflage ein Beweis von Cayleys Formel, mehr über blockierende Flüsse, der neue schnellere b -Matching-Separationsalgorithmus, ein Approximationsschema für das multidimensionale Knapsack-Problem, und Resultate zum Multicommodity-Max-Flow-Min-Cut-Quotienten und dem Sparsest-Cut-Problem. Weitere kleine Verbesserungen sind an vielen Stellen eingeflossen und mehr als sechzig neue Übungsaufgaben sind hinzugekommen. Selbstverständlich haben wir die Literaturangaben aktualisiert und Hinweise auf die neuesten Arbeiten aufgenommen sowie einige kleine Fehler bereinigt, die entdeckt wurden.

Wir möchten Takao Asano, Ulrich Brenner, Benjamin Bolten, Christoph Buchheim, Jean Fonlupt, András Frank, Michael Gester, Stephan Held, Stefan Hougardy, Hiroshi Iida, Klaus Jansen, Alexander Karzanov, Alexander Kleff, Niko Klewinghaus, Stefan Knauf, Barbara Langfeld, Jens Maßberg, Marc Pfetsch, Klaus Radke, Rabe von Radow, Tomás Salles, Jan Schneider, Christian Schulte, Andras Sebő, Martin Skutella, Jácint Szabó, und Simon Wedeking für wertvolle Kommentare zur vorherigen Auflage danken.

Wir freuen uns, dass dieses Buch auf eine so gute Resonanz stößt und in weitere Sprachen übersetzt wird. Japanische, französische, italienische, deutsche, russische und chinesische Ausgaben sind seit 2009 erschienen oder werden bald erscheinen. Wir hoffen, dass unser Buch weiterhin gute Dienste in Forschung und Lehre leisten wird.

Bonn, September 2011

Bernhard Korte und Jens Vygen

Vorwort zur vierten englischen Auflage

Die vorliegende vierte englische Auflage und Übersetzungen in vier weitere Sprachen zeigen das große Interesse an unserem Buch, über das wir uns natürlich freuen. Wiederum sind Verbesserungen, Aktualisierungen und wesentliche Ergänzungen in diese Auflage eingeflossen. Wir haben einige klassische Themen eingefügt, die der Leser bisher vielleicht vermisst hat, insbesondere im Bereich der linearen Optimierung, des Netzwerk-Simplexalgorithmus und des Max-Cut-Problems. Ferner haben wir einige neue Aufgaben und aktuelle Literaturangaben hinzugefügt. Wir hoffen, dass damit unser Buch als Grundlage für Forschung und Lehre noch besser geeignet ist.

Wir danken wiederum der Union der Deutschen Akademien der Wissenschaften und der Nordrhein-Westfälischen Akademie der Wissenschaften für die Unterstützung im Rahmen des Langzeitforschungsprojekts „Diskrete Mathematik und Anwendungen“. Weiter danken wir allen, die uns wertvolle Kommentare zur dritten Auflage gegeben haben, insbesondere Takao Asano, Christoph Bartoschek, Bert Besser, Ulrich Brenner, Jean Fonlupt, Satoru Fujishige, Marek Karpinski, Jens Maßberg, Denis Naddef, Sven Peyer, Klaus Radke, Rabe von Radow, Dieter Rautenbach, Martin Skutella, Markus Struzyna, Jürgen Werber, Minyi Yue und Guochuan Zhang.

Auf der Internetseite www.or.uni-bonn.de/~vygen/co.html werden wir die Leser weiterhin auf dem Laufenden halten.

Bonn, im August 2007

Bernhard Korte und Jens Vygen

Vorwort zur dritten englischen Auflage

Nach fünf Jahren war es an der Zeit, eine vollständig überarbeitete und wesentlich erweiterte neue Auflage herauszugeben. Die wesentlichste Ergänzung ist ein ganz neues Kapitel über Standortprobleme. Für diese wichtige Klasse von *NP*-schweren Problemen waren noch vor acht Jahren keine Approximationsalgorithmen mit konstanter Gütegarantie bekannt. Heute gibt es einige sehr verschiedene, interessante Methoden, die zu guten Approximationsgarantien führen. Dadurch hat dieses Gebiet eine große Anziehungskraft entwickelt, auch für Lehrzwecke. In der Tat ist das neue Kapitel aus einer Spezialvorlesung über Standortprobleme hervorgegangen.

Viele der anderen Kapitel sind wesentlich ergänzt worden. Neu hinzugekommen sind insbesondere Fibonacci-Heaps, Fujishiges neuer Algorithmus für maximale Flüsse, zeitabhängige Flüsse, Schrijvers Algorithmus zur Minimierung submodularer Funktionen und der Approximationsalgorithmus für Steinerbäume von Robins und Zelikovsky. Mehrere Beweise sind geglättet und viele neue Aufgaben und Literaturhinweise hinzugefügt worden.

Wir danken allen, die uns mit ihren wertvollen Kommentaren zur zweiten Auflage geholfen haben, insbesondere Takao Asano, Yasuhito Asano, Ulrich Brenner, Stephan Held, Tomio Hirata, Dirk Müller, Kazuo Murota, Dieter Rautenbach, Martin Skutella, Markus Struzyna und Jürgen Werber. Besonders Takao Asanos Anmerkungen und Jürgen Werbers sorgfältige Durchsicht von Kapitel 22 haben wesentlich zur Verbesserung beigetragen.

Auch möchten wir hier wieder der Union der deutschen Akademien der Wissenschaften und der Nordrhein-Westfälische Akademie der Wissenschaften für die ununterbrochene Unterstützung im Rahmen des langfristigen Forschungsprojekts „Diskrete Mathematik und Anwendungen“ danken, das vom Bundesminister für Bildung und Forschung und vom Land Nordrhein-Westfalen finanziert wird.

Bonn, im Mai 2005

Bernhard Korte und Jens Vygen

Vorwort zur zweiten englischen Auflage

Es war für uns eine große Überraschung, dass die erste Auflage des vorliegenden Buches bereits nach rund einem Jahr vergriffen war. Die vielen positiven und sogar enthusiastischen Bemerkungen und Mitteilungen von Kollegen und aus der Leserschaft haben uns gefreut. Einige unserer Kollegen waren uns bei der Fehlersuche behilflich; insbesondere danken wir Ulrich Brenner, András Frank, Bernd Gärtner und Rolf Möhring. Selbstverständlich haben wir alle Korrekturen in diese Auflage eingearbeitet. Die Literaturverzeichnisse wurden sorgfältig aktualisiert.

Im Vorwort zur ersten Auflage haben wir zwar alle Personen erwähnt, die uns bei der Erstellung des Buches geholfen haben, es aber leider versäumt, uns für die Unterstützung durch verschiedene Institutionen zu bedanken. Das wollen wir hier nachholen.

Es ist selbstverständlich, dass ein Projekt wie dieses Buch, das sich über einen Zeitraum von sieben Jahren erstreckt hat, nicht ohne die Unterstützung verschiedener Forschungsförderer zustande kommen konnte. Dankend erwähnen möchten wir besonders das durch die Ungarische Akademie der Wissenschaften und die Deutsche Forschungsgemeinschaft geförderte bilaterale Forschungsprojekt, zwei Sonderforschungsbereiche der Deutschen Forschungsgemeinschaft, den Ministère Français de la Recherche et de la Technologie und die Alexander-von-Humboldt-Stiftung für ihre Unterstützung durch den Prix Alexandre de Humboldt sowie die EU-Kommission für die Teilnahme an zwei DONET-Projekten. Ganz besonders haben wir der Union der Deutschen Akademien der Wissenschaften und der Nordrhein-Westfälischen Akademie der Wissenschaften zu danken. Ihr langfristiges, vom Bundesministerium für Bildung und Forschung (BMBF) und vom Land Nordrhein-Westfalen unterstütztes Projekt „Diskrete Mathematik und Anwendungen“ spielte eine besondere Rolle bei der Entstehung dieses Buches.

Bonn, im Oktober 2001

Bernhard Korte und Jens Vygen

Vorwort zur ersten englischen Auflage

Kombinatorische Optimierung ist eines der jüngsten und aktivsten Gebiete der diskreten Mathematik und heute wahrscheinlich ihre treibende Kraft. Seit rund fünfzig Jahren ist die kombinatorische Optimierung ein eigenständiges Fachgebiet.

Das vorliegende Buch beschreibt die wichtigsten Ideen, theoretischen Resultate und Algorithmen der kombinatorischen Optimierung. Wir haben es als ein fortgeschrittenes Lehrbuch konzipiert, das auch als aktuelles Nachschlagewerk der neuesten Forschungsergebnisse dienen kann. Es enthält die grundlegenden Definitionen und Resultate der Graphentheorie, der linearen und ganzzahligen Optimierung und der Komplexitätstheorie. Es deckt sowohl die klassischen als auch die neuesten Gebiete der kombinatorischen Optimierung ab. Wir haben den Schwerpunkt auf theoretische Resultate und beweisbar gute Algorithmen gelegt. Anwendungen und Heuristiken werden nur gelegentlich erwähnt.

Die kombinatorische Optimierung hat ihre Wurzeln in der Kombinatorik, im Operations Research und in der theoretischen Informatik. Sie wird getragen von einer Vielzahl konkreter Anwendungsprobleme, die als abstrakte kombinatorische Optimierungsprobleme formuliert werden können. Wir konzentrieren uns auf das detaillierte Studium klassischer Probleme, die in vielfachen Zusammenhängen auftreten, und auf die zugrunde liegende Theorie.

Die meisten Probleme der kombinatorischen Optimierung können auf natürliche Weise als Graphenproblem oder als (ganzzahliges) lineares Programm formuliert werden. Daher beginnt dieses Buch, nach einem einführenden Kapitel, mit den Grundlagen der Graphentheorie und denjenigen Ergebnissen aus der linearen und ganzzahligen Optimierung, die für die kombinatorische Optimierung besonders relevant sind.

Danach werden die klassischen Gebiete der kombinatorischen Optimierung eingehend betrachtet: minimale aufspannende Bäume, kürzeste Wege, Netzwerkflüsse, Matchings und Matroide. Die meisten in den Kapiteln 6–14 besprochenen Probleme haben polynomielle (“effiziente”) Algorithmen, während die Mehrzahl der in den Kapiteln 15–21 besprochenen Probleme *NP*-schwer sind, d. h. ein polynomieller Algorithmus existiert vermutlich nicht. In vielen Fällen kann man jedoch Approximationsalgorithmen mit einer gewissen Gütegarantie finden. Wir erwähnen auch einige andere Strategien zur Behandlung solcher “schweren” Probleme.

Der Umfang dieses Buches geht in verschiedener Hinsicht über normale Lehrbücher der kombinatorischen Optimierung hinaus. Beispielsweise behandeln wir

auch die Äquivalenz von Optimierung und Separation (für volldimensionale Polytope), $O(n^3)$ -Implementierungen von Matching-Algorithmen, basierend auf Ohrenzerlegungen, Turingmaschinen, den (schwachen) Perfekte-Graphen-Satz von Lovász, MAXSNP-schwere Probleme, den Karmarkar-Karp-Algorithmus für Bin-Packing sowie neuere Approximationssalgorithmen für Mehrgüterflüsse, Survivable-Network-Design und das Euklidische Traveling-Salesman-Problem. Alle Resultate werden vollständig bewiesen.

Natürlich kann kein Buch über kombinatorische Optimierung das Gebiet erschöpfend behandeln. Beispiele von Themen, die wir nur kurz streifen oder gar nicht erwähnen, sind Baumzerlegungen, Separatoren, submodulare Flüsse, Path-Matchings, Delta-Matroide, das Paritätsproblem für Matroide, Standort- und Scheduling-Probleme, nichtlineare Optimierung, semidefinite Optimierung, die Average-Case-Analyse von Algorithmen, kompliziertere Datenstrukturen, parallele und randomisierte Algorithmen sowie die Theorie der probabilistisch prüfbaren Beweise (den *PCP*-Satz zitieren wir ohne Beweis).

Am Ende eines jeden Kapitels befinden sich Übungsaufgaben, die vielfach weitere Resultate und Anwendungen enthalten. Die vermutlich schwierigeren Aufgaben sind mit einem Sternchen markiert. Jedes Kapitel schließt mit einem Literaturverzeichnis, das auch weiterführende Literatur enthält.

Dieses Buch ist aus mehreren Vorlesungen über kombinatorische Optimierung und aus Spezialvorlesungen über Themen wie polyedrische Optimierung und Approximationssalgorithmen entstanden. Somit bietet es sowohl Material für einführende als auch für fortgeschrittene Lehrveranstaltungen.

Selbstverständlich haben wir viel aus Diskussionen mit Kollegen und Freunden und von deren Vorschlägen gelernt, und natürlich auch aus der vorhandenen Literatur. Besonderen Dank schulden wir András Frank, László Lovász, András Recski, Alexander Schrijver und Zoltán Szigeti. Unsere Kollegen und Studenten in Bonn, Christoph Albrecht, Ursula Bünnagel, Thomas Emden-Weinert, Mathias Hauptmann, Sven Peyer, Rabe von Randow, André Rohe, Martin Thimm und Jürgen Werber, haben verschiedene Versionen des Manuskripts sorgfältig gelesen und viele Verbesserungsvorschläge gemacht. Schließlich möchten wir dem Springer-Verlag für die äußerst effiziente Zusammenarbeit danken.

Bonn, im Januar 2000

Bernhard Korte und Jens Vygen

Inhaltsverzeichnis

1	Einführung	1
1.1	Enumeration	2
1.2	Die Laufzeit von Algorithmen	5
1.3	Lineare Optimierungsprobleme	9
1.4	Sortieren	10
	Aufgaben	12
	Literatur	13
2	Graphen	15
2.1	Grundlegende Definitionen	15
2.2	Bäume, Kreise und Schnitte	20
2.3	Zusammenhang	27
2.4	Eulersche und bipartite Graphen	35
2.5	Planarität	38
2.6	Planare Dualität	47
	Aufgaben	49
	Literatur	54
3	Lineare Optimierung	57
3.1	Polyeder	59
3.2	Der Simplexalgorithmus	63
3.3	Implementierung des Simplexalgorithmus	66
3.4	Dualität	70
3.5	Konvexe Hüllen und Polytope	74
	Aufgaben	75
	Literatur	78
4	Algorithmen für lineare Optimierung	81
4.1	Die Größe von Ecken und Seitenflächen	81
4.2	Kettenbrüche	84
4.3	Gauß-Elimination	87
4.4	Die Ellipsoidmethode	91
4.5	Der Satz von Khachiyan	97
4.6	Separation und Optimierung	99
	Aufgaben	105
	Literatur	107

5 Ganzzahlige Optimierung	109
5.1 Die ganzzahlige Hülle eines Polyeders	111
5.2 Unimodulare Transformationen	116
5.3 Vollständige duale Ganzahligkeit (TDI)	118
5.4 Vollständig-unimodulare Matrizen	122
5.5 Schnittebenen	127
5.6 Lagrange-Relaxierung	132
Aufgaben	134
Literatur	138
6 Aufspannende Bäume und Arboreszenzen	141
6.1 Aufspannende Bäume mit minimalem Gewicht	142
6.2 Arboreszenzen mit minimalem Gewicht	148
6.3 Polyedrische Darstellungen	152
6.4 Das Packen von aufspannenden Bäumen und Arboreszenzen	156
Aufgaben	160
Literatur	164
7 Kürzeste Wege	167
7.1 Kürzeste Wege von einer Quelle aus	168
7.2 Kürzeste Wege zwischen allen Knotenpaaren	173
7.3 Kreise mit minimalem durchschnittlichem Kantengewicht	176
7.4 Seichte leichte Bäume	178
Aufgaben	180
Literatur	182
8 Netzwerkflüsse	185
8.1 Das Max-Flow-Min-Cut-Theorem	186
8.2 Der Satz von Menger	190
8.3 Der Edmonds-Karp-Algorithmus	193
8.4 Dinic', Karzanovs, und Fujishiges Algorithmus	194
8.5 Der Goldberg-Tarjan-Algorithmus	199
8.6 Gomory-Hu-Bäume	204
8.7 Die minimale Kapazität eines Schnittes in einem ungerichteten Graphen	210
Aufgaben	213
Literatur	221
9 Flüsse mit minimalen Kosten	227
9.1 Formulierung des Problems	227
9.2 Ein Optimalitätskriterium	230
9.3 Der Minimum-Mean-Cycle-Cancelling-Algorithmus	232
9.4 Der Sukzessive-Kürzeste-Wege-Algorithmus	236
9.5 Orlins Algorithmus	240
9.6 Der Netzwerk-Simplexalgorithmus	245
9.7 Zeitabhängige Flüsse	249

Aufgaben	252
Literatur	255
10 Kardinalitätsmaximale Matchings	259
10.1 Bipartite Matchings	260
10.2 Die Tutte-Matrix	262
10.3 Der Satz von Tutte	264
10.4 Ohrenzerlegungen faktorkritischer Graphen	268
10.5 Edmonds' Matching-Algorithmus	274
Aufgaben	284
Literatur	288
11 Gewichtete Matchings	293
11.1 Das Zuordnungsproblem	294
11.2 Abriss des gewichteten Matching-Algorithmus	296
11.3 Implementierung des gewichteten Matching-Algorithmus	299
11.4 Postoptimierung	313
11.5 Das Matching-Polytop	314
Aufgaben	318
Literatur	320
12 <i>b</i>-Matchings und <i>T</i>-Joins	323
12.1 <i>b</i> -Matchings	323
12.2 <i>T</i> -Joins mit minimalem Gewicht	327
12.3 <i>T</i> -Joins und <i>T</i> -Schnitte	332
12.4 Der Satz von Padberg und Rao	336
Aufgaben	339
Literatur	343
13 Matroide	345
13.1 Unabhängigkeitssysteme und Matroide	345
13.2 Andere Matroidaxiome	350
13.3 Dualität	354
13.4 Der Greedy-Algorithmus	359
13.5 Der Schnitt von Matroiden	364
13.6 Matroid-Partitionierung	369
13.7 Gewichteter Schnitt von Matroiden	371
Aufgaben	375
Literatur	378
14 Verallgemeinerungen von Matroiden	381
14.1 Greedoide	381
14.2 Polymatroide	385
14.3 Die Minimierung submodularer Funktionen	390
14.4 Schrijvers Algorithmus	392
14.5 Symmetrische submodulare Funktionen	396

14.6 Die Maximierung submodularer Funktionen	398
Aufgaben	401
Literatur	405
15 NP-Vollständigkeit	409
15.1 Turingmaschinen	410
15.2 Die Church'sche These	412
15.3 <i>P</i> und <i>NP</i>	417
15.4 Der Satz von Cook	422
15.5 Einige grundlegende <i>NP</i> -vollständige Probleme	426
15.6 Die Klasse <i>coNP</i>	433
15.7 <i>NP</i> -schwere Probleme	436
Aufgaben	441
Literatur	446
16 Approximationsalgorithmen	449
16.1 Das Set-Covering-Problem	450
16.2 Das Max-Cut-Problem	455
16.3 Färbung	462
16.4 Approximationsschemata	470
16.5 Maximum-Satisfiability	473
16.6 Der <i>PCP</i> -Satz	478
16.7 L-Reduktionen	482
Aufgaben	489
Literatur	493
17 Das Knapsack-Problem	499
17.1 Das gebrochene Knapsack-Problem und das gewichtete Median-Problem	499
17.2 Ein pseudopolynomieller Algorithmus	502
17.3 Ein voll-polynomielles Approximationsschema	504
17.4 Das multidimensionale Knapsack-Problem	507
17.5 Der Nemhauser-Ullmann-Algorithmus	509
Aufgaben	513
Literatur	514
18 Bin-Packung	517
18.1 Greedy-Heuristiken	518
18.2 Ein asymptotisches Approximationsschema	523
18.3 Der Karmarkar-Karp-Algorithmus	527
Aufgaben	531
Literatur	533

19 Mehrgüterflüsse und kantendisjunkte Wege	537
19.1 Mehrgüterflüsse	538
19.2 Algorithmen für Mehrgüterflüsse	543
19.3 Das Sparsest-Cut-Problem und der Max-Flow-Min-Cut-Quotient	548
19.4 Der Satz von Leighton und Rao	549
19.5 Das gerichtete Kantendisjunkte-Wege-Problem	552
19.6 Das ungerichtete Kantendisjunkte-Wege-Problem	556
Aufgaben	563
Literatur	568
20 Netzwerk-Design-Probleme	573
20.1 Steinerbäume	574
20.2 Der Robins-Zelikovsky-Algorithmus	579
20.3 Rundung des Gerichtete-Komponenten-LP	586
20.4 Survivable-Network-Design	592
20.5 Ein primal-dualer Approximationsalgorithmus	595
20.6 Jains Algorithmus	604
20.7 Das VPN-Problem	611
Aufgaben	614
Literatur	618
21 Das Traveling-Salesman-Problem	623
21.1 Approximationsalgorithmen für das TSP	623
21.2 Das euklidische TSP	628
21.3 Lokale Suche	636
21.4 Das Traveling-Salesman-Polytop	642
21.5 Untere Schranken	649
21.6 Branch-and-Bound	652
Aufgaben	655
Literatur	659
22 Standortprobleme	663
22.1 Das unbeschränkte Standortproblem	663
22.2 Rundung von LP-Lösungen	665
22.3 Primal-duale Algorithmen	667
22.4 Skalierung und Greedy-Augmentierung	673
22.5 Beschränkung der Standortanzahl	676
22.6 Lokale Suche	680
22.7 Beschränkte Standortprobleme	686
22.8 Das universelle Standortproblem	689
Aufgaben	697
Literatur	699
Symbolverzeichnis	701
Personenverzeichnis	705
Stichwortverzeichnis	717



1 Einführung

Beginnen wir mit zwei Beispielen.

Eine Firma betreibt eine Maschine, mit der Löcher in Leiterplatten gebohrt werden. Da sie viele dieser Platinen herstellt, möchte sie erreichen, dass die Maschine jede Leiterplatte möglichst schnell fertigstellt. Die Bohrzeit selbst können wir nicht verringern, wohl aber die von der Maschine verbrauchte Zeit, um zu den Bohrpunkten zu kommen. Meist verfügen Bohrmaschinen über zwei verschiedene Bewegungsrichtungen: Das Objekt bewegt sich horizontal und der Bohrarm vertikal. Beide Bewegungen können gleichzeitig erfolgen, also ist die von der Maschine benötigte Zeit zwischen zwei aufeinander folgenden Bohrpunkten proportional zur größeren der beiden Weglängen: horizontal und vertikal. Dies ist der sogenannte ℓ_∞ -Abstand. (Ältere Maschinen verfügen auch über die zwei Bewegungsrichtungen: horizontal und vertikal, jedoch nicht gleichzeitig. Hier ist die Bewegungszeit zwischen zwei aufeinander folgenden Bohrpunkten proportional zum sogenannten ℓ_1 -Abstand, der Summe der horizontalen und der vertikalen Weglänge.)

Ein optimaler Weg zwischen den Bohrpunkten wird durch eine bestimmte Reihenfolge der Bohrpunkte p_1, \dots, p_n gegeben, mit der Eigenschaft, dass $\sum_{i=1}^{n-1} d(p_i, p_{i+1})$ minimal ist, wobei d der ℓ_∞ -Abstand ist: Für zwei Punkte $p = (x, y)$ und $p' = (x', y')$ in der Ebene schreiben wir $d(p, p') := \max\{|x - x'|, |y - y'|\}$. Eine solche Reihenfolge der Bohrpunkte kann man mittels einer Permutation darstellen, d. h. einer Bijektion $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$.

Welche Permutation die beste ist, hängt natürlich von den Positionen der Bohrpunkte ab; für jede Liste von Bohrpunktpositionen haben wir eine andere Instanz des Problems. Wir sagen: Eine Instanz unseres Problems besteht aus einer Liste von Punkten in der Ebene, d. h. der Koordinaten der Bohrpunkte. Das Problem kann nun folgendermaßen formal angegeben werden:

BOHRPUNKTPROBLEM

Instanz: Eine Menge von Punkten $p_1, \dots, p_n \in \mathbb{R}^2$.

Aufgabe: Bestimme eine Permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$, so dass $\sum_{i=1}^{n-1} d(p_{\pi(i)}, p_{\pi(i+1)})$ minimal ist.

Nun werden wir unser zweites Beispiel beschreiben. Hier geht es um eine Anzahl von Arbeiten (Jobs), die erledigt werden müssen, wobei jede eine gewisse Zeit in Anspruch nimmt. Jeder Job kann von einer bestimmten Teilmenge der Arbei-

ter erledigt werden und die Arbeiter unterscheiden sich nicht bezüglich Effizienz. Mehrere Arbeiter können gleichzeitig an demselben Job tätig sein und ein Arbeiter kann auch bei mehreren Jobs mitwirken, jedoch nicht gleichzeitig. Ziel ist es, mit allen Jobs so früh wie möglich fertig zu werden.

Hier genügt es, für jeden Arbeiter anzugeben, welchen Jobs er zugeteilt wird und jeweils für welchen Zeitraum. Die Reihenfolge, in der er dann an den ihm zugeteilten Jobs tätig ist, ist nicht wichtig, da der Zeitpunkt, zu dem alle Jobs fertig sind, offensichtlich nur von der längsten der Gesamtarbeitszeiten der einzelnen Arbeiter abhängt. Wir müssen demnach das folgende Problem lösen:

JOB-ZUORDNUNGSPROBLEM

Instanz: Eine Zahlenmenge $t_1, \dots, t_n \in \mathbb{R}_+$ (jeweils die für die n Jobs benötigte Zeit), eine Anzahl $m \in \mathbb{N}$ von Arbeitern und für jeden Job $i \in \{1, \dots, n\}$ eine nichtleere Teilmenge $S_i \subseteq \{1, \dots, m\}$ der Arbeiter.

Aufgabe: Bestimme Zahlen $x_{ij} \in \mathbb{R}_+$ für alle $i = 1, \dots, n$ und $j \in S_i$, so dass $\sum_{j \in S_i} x_{ij} = t_i$ für $i = 1, \dots, n$ und $\max_{j \in \{1, \dots, m\}} \sum_{i: j \in S_i} x_{ij}$ minimal ist.

Dies sind zwei typische Probleme aus der kombinatorischen Optimierung. In diesem Buch werden wir nicht darauf eingehen, wie man ein Problem aus der Praxis als kombinatorisches Optimierungsproblem darstellt; in der Tat gibt es hierfür kein allgemeines Rezept. Neben der präzisen Formulierung des Inputs und des gewünschten Outputs ist es oft wichtig, unnötige Komponenten des Problems zu ignorieren (z. B. im ersten Beispiel die Bohrzeiten, die wir nicht ändern können, oder im zweiten Beispiel die Reihenfolge, in der die Arbeiter ihre Jobs verrichten).

Hier sind wir natürlich nicht an der Lösung eines konkreten Bohrpunktproblems oder Job-Zuordnungsproblems für eine gewisse Firma interessiert, sondern an Lösungswegen für diese typischen kombinatorischen Optimierungsprobleme. Zunächst betrachten wir das BOHRPUNKTPROBLEM.

1.1 Enumeration

Wie könnte eine Lösung des BOHRPUNKTPROBLEMS aussehen? Da es unendlich viele Instanzen (endliche Punktmengen in der Ebene) gibt, ist es aussichtslos, eine Liste der optimalen Lösungen für alle Instanzen zu erstellen. Stattdessen brauchen wir einen Algorithmus, der für eine gegebene Instanz eine optimale Lösung bestimmt. Ein solcher Algorithmus existiert in der Tat: Für eine gegebene Menge von n Punkten braucht man nur für jede der $n!$ möglichen Reihenfolgen die ℓ_∞ -Länge des entsprechenden Weges zu berechnen.

Es gibt verschiedene Formulierungen eines Algorithmus, die sich hauptsächlich durch ihre Detailliertheit und die formale Darstellung unterscheiden. Die folgende Formulierung würde z. B. nicht als Algorithmus genügen: „Gegeben eine Menge von n Punkten, finde einen optimalen Weg und gib ihn als Output an.“ Hier wird

überhaupt nicht angegeben, wie man zu einer optimalen Lösung kommen soll. Der obige Vorschlag, alle $n!$ möglichen Reihenfolgen zu enumerieren, ist schon um einiges nützlicher, aber es bleibt weiterhin unklar, wie man diese Enumeration vornehmen soll. Eine Möglichkeit wäre die folgende:

Man enumeriert alle n -Tupel der Zahlen $1, \dots, n$, d.h. alle n^n Vektoren der Menge $\{1, \dots, n\}^n$. Diese kann man einfach abzählen: Beginnend mit $(1, \dots, 1, 1)$, dann $(1, \dots, 1, 2)$, zählt man bis zu $(1, \dots, 1, n)$, dann $(1, \dots, 1, 2, 1)$, und so weiter. Genauer: Wir erhöhen die n -te Komponente schrittweise um 1 bis sie n erreicht, dann gehen wir zur letzten Komponente die kleiner als n ist, erhöhen diese um 1 und setzen alle folgenden Komponenten auf 1. Diese Methode ist auch unter dem Namen „Backtracking“ bekannt und die resultierende Reihenfolge der Vektoren aus $\{1, \dots, n\}^n$ heißt die lexikographische Ordnung:

Definition 1.1. Seien x und $y \in \mathbb{R}^n$ zwei Vektoren. Dann nennt man x **lexikographisch kleiner** als y , falls es einen Index $j \in \{1, \dots, n\}$ mit $x_i = y_i$ für $i = 1, \dots, j - 1$ und $x_j < y_j$ gibt.

Wissen wir, wie man alle Vektoren aus $\{1, \dots, n\}^n$ enumeriert, so brauchen wir bei jedem Vektor nur zu überprüfen, ob seine Komponenten paarweise verschieden sind und, falls ja, ob der durch diesen Vektor gegebene Weg kürzer als der kürzeste bisher gefundene Weg ist.

Da dieser Algorithmus alle n^n Vektoren enumeriert, wird er mindestens n^n Schritte (in der Tat sogar mehr) benötigen. Es geht aber besser. Es gibt nur $n!$ Permutationen von $\{1, \dots, n\}$ und $n!$ ist wesentlich kleiner als n^n . (Nach der Stirlingformel ist $n! \approx \sqrt{2\pi n} \frac{n^n}{e^n}$ (Stirling [1730]); siehe Aufgabe 1). Wir werden nun zeigen, wie man alle Wege mit ungefähr $n^2 \cdot n!$ Schritten enumerieren kann. Betrachte den folgenden Algorithmus, der alle Permutationen in lexikographischer Ordnung enumeriert:

WEG-ENUMERATIONS-ALGORITHMUS

Input: Eine natürliche Zahl $n \geq 3$. Eine Menge $\{p_1, \dots, p_n\}$ von Punkten in der Ebene.

Output: Eine Permutation $\pi^* : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$, die $\text{cost}(\pi^*) := \sum_{i=1}^{n-1} d(p_{\pi^*(i)}, p_{\pi^*(i+1)})$ minimiert.

- ① Setze $\pi(i) := i$ und $\pi^*(i) := i$ für $i = 1, \dots, n$. Setze $i := n - 1$.
- ② Sei $k := \min(\{\pi(i) + 1, \dots, n + 1\} \setminus \{\pi(1), \dots, \pi(i - 1)\})$.
- ③ **If** $k \leq n$ **then**:
 - Setze $\pi(i) := k$.
 - If** $i = n$ und $\text{cost}(\pi) < \text{cost}(\pi^*)$ **then** setze $\pi^* := \pi$.
 - If** $i < n$ **then** setze $\pi(i + 1) := 0$ und $i := i + 1$.
 - If** $k = n + 1$ **then** setze $i := i - 1$.
 - If** $i \geq 1$ **then** go to ②.

Beginnend mit $(\pi(i))_{i=1,\dots,n} = (1, 2, 3, \dots, n-1, n)$ und $i = n-1$, findet der Algorithmus schrittweise den nächsten möglichen Wert von $\pi(i)$ (ohne von $\pi(1), \dots, \pi(i-1)$ Gebrauch zu machen). Ist dies unmöglich (d. h. $k = n+1$), dann verringert der Algorithmus i um 1 (Backtracking), sonst setzt er $\pi(i)$ gleich dem neuen Wert. Ist $i = n$, so liegt die neue Permutation vor, sonst wird der Algorithmus alle möglichen Werte von $\pi(i+1), \dots, \pi(n)$ durchprobieren und beginnt dies, indem er $\pi(i+1) := 0$ setzt und dann i schrittweise um 1 erhöht.

Auf diese Weise werden alle Permutationsvektoren $(\pi(1), \dots, \pi(n))$ in lexicographischer Ordnung erzeugt. Die ersten Iterationen sind z. B. für $n = 6$:

$$\begin{aligned}
 \pi &:= (1, 2, 3, 4, 5, 6), & i &:= 5 \\
 k &:= 6, \quad \pi := (1, 2, 3, 4, 6, 0), & i &:= 6 \\
 k &:= 5, \quad \pi := (1, 2, 3, 4, 6, 5), & & \text{cost}(\pi) < \text{cost}(\pi^*)? \\
 k &:= 7, & i &:= 5 \\
 k &:= 7, & i &:= 4 \\
 k &:= 5, \quad \pi := (1, 2, 3, 5, 0, 5), & i &:= 5 \\
 k &:= 4, \quad \pi := (1, 2, 3, 5, 4, 0), & i &:= 6 \\
 k &:= 6, \quad \pi := (1, 2, 3, 5, 4, 6), & & \text{cost}(\pi) < \text{cost}(\pi^*)?
 \end{aligned}$$

Da der Algorithmus die Kosten eines jeden Weges mit denen von π^* , dem besten bisher gefundenen Weg, vergleicht, ergibt der Output tatsächlich den optimalen Weg. Wie viele Schritte durchläuft dieser Algorithmus? Natürlich hängt dies davon ab, was wir einen Schritt nennen. Da wir vermeiden wollen, dass die Anzahl der Schritte von der gewählten Implementierung abhängt, werden wir konstante Faktoren ignorieren. Auf jedem vernünftigen Rechner wird ① mindestens $2n + 1$ Schritte in Anspruch nehmen (dies ist die Anzahl der vorgenommenen Variablenzuweisungen) und höchstens cn Schritte für eine gewisse Konstante c . Die folgende Notation für die Vernachlässigung konstanter Faktoren ist üblich:

Definition 1.2. Seien $f, g : D \rightarrow \mathbb{R}_+$ zwei Funktionen. Man sagt: f ist $O(g)$ (und schreibt gelegentlich $f = O(g)$ und auch $g = \Omega(f)$), falls es Konstanten $\alpha, \beta > 0$ mit $f(x) \leq \alpha g(x) + \beta$ für alle $x \in D$ gibt. Ist $f = O(g)$ und $g = O(f)$, so sagt man auch: $f = \Theta(g)$ (oder natürlich auch $g = \Theta(f)$). Dann haben f und g dieselbe **Wachstumsrate**.

Beachte, dass der Gebrauch des Gleichheitszeichens in der O -Notation nicht symmetrisch ist. Zur Erläuterung dieser Definition sei $D = \mathbb{N}$ und $f(n)$ die Anzahl der elementaren Schritte in ① und $g(n) = n$ ($n \in \mathbb{N}$). Offensichtlich haben wir hier $f = O(g)$ (in der Tat sogar $f = \Theta(g)$); man sagt: ① hat eine $O(n)$ (oder lineare) Laufzeit. Eine einzige Ausführung von ③ braucht eine konstante Anzahl von Schritten (man spricht von $O(1)$ oder konstanter Laufzeit), außer wenn $k \leq n$ und $i = n$; in diesem Fall müssen die Kosten der zwei Wege verglichen werden, was eine $O(n)$ -Laufzeit benötigt.

Wie sieht es bei ② aus? Eine naive Implementierung, nämlich die Überprüfung von $j = \pi(h)$ für jedes $j \in \{\pi(i)+1, \dots, n\}$ und jedes $h \in \{1, \dots, i-1\}$, braucht $O((n-\pi(i))i)$ Schritte, was bis zu $\Theta(n^2)$ sein kann. Eine bessere Implementierung von ② benutzt ein Hilfsarray $\text{aux}(j)$, $j = 1, \dots, n$:

② **For** $j := 1$ **to** n **do** $\text{aux}(j) := 0$.
For $j := 1$ **to** $i - 1$ **do** $\text{aux}(\pi(j)) := 1$.
Setze $k := \pi(i) + 1$.
While $k \leq n$ und $\text{aux}(k) = 1$ **do** $k := k + 1$.

Mit dieser Implementierung braucht eine einzige Ausführung von ② offensichtlich nur $O(n)$ -Laufzeit. Mit solchen einfachen Methoden wie dieser werden wir uns in diesem Buch gewöhnlich nicht aufzuhalten; wir gehen davon aus, dass man solche Implementierungen selbst konstruieren kann.

Nachdem wir nun die Laufzeiten der verschiedenen Schritte besprochen haben, können wir uns der Gesamlaufzeit zuwenden. Da die Anzahl der Permutationen gleich $n!$ ist, müssen wir nur die Arbeit abschätzen, die zwischen zwei Permutationen notwendig ist. Der Zeiger i bewegt sich z. B. von n zurück auf den Index i' , wo ein neuer Wert $\pi(i') \leq n$ gefunden wird. Dann bewegt er sich wieder vorwärts bis auf $i = n$. Während der Zeiger i still steht, werden ② und ③ beide genau einmal ausgeführt, außer wenn $k \leq n$ und $i = n$, wo sie beide zweimal ausgeführt werden. Somit beläuft sich die Gesamtarbeit zwischen zwei Permutationen auf höchstens $4n$ -mal ② und ③, d. h. $O(n^2)$. Also hat der WEG-ENUMERATIONS-ALGORITHMUS eine Gesamlaufzeit von $O(n^2 n!)$.

Die Laufzeit kann man sogar noch etwas verbessern. Eine eingehende Untersuchung zeigt, dass die Laufzeit nur $O(n \cdot n!)$ ist (siehe Aufgabe 4).

Dennoch ist der Algorithmus für große n viel zu zeitaufwendig. Das Problem bei der Enumeration aller Wege ist, dass die Anzahl der Wege exponentiell mit der Anzahl der Punkte wächst; bereits für 20 Punkte gibt es $20! = 2\,432\,902\,008\,176\,640\,000 \approx 2,4 \cdot 10^{18}$ verschiedene Wege und selbst die schnellsten Rechner würden einige Jahre benötigen, um alle Wege zu bearbeiten. Somit ist die vollständige Enumeration bereits für mittelgroße Instanzen gänzlich ungeeignet.

Das Kernproblem der kombinatorischen Optimierung besteht darin, bessere Algorithmen für Probleme wie diesem zu suchen. Oft möchte man das beste Element aus einer endlichen Menge von zulässigen Lösungen finden (in unserem Beispiel Bohrpunktreihenfolgen oder Permutationen). Für diese Menge gibt es keine explizite Liste, sie hängt aber implizit von der Struktur des Problems ab. Ein Algorithmus muss deswegen diese Struktur ausnutzen.

Im Falle des BOHRPUNKTPROBLEMS steckt die ganze Information einer Instanz mit n Punkten in den gegebenen $2n$ Koordinaten. Während der naive Algorithmus alle $n!$ Wege enumeriert, könnte es durchaus einen effizienteren Algorithmus geben, z. B. einen, der die optimale Lösung in n^2 Schritten findet. Es ist nicht bekannt, ob es einen solchen Algorithmus gibt (einige Resultate in Kapitel 15 deuten jedoch darauf hin, dass dies eher unwahrscheinlich ist). Trotzdem gibt es viel bessere Algorithmen für dieses Problem als den naiven.

1.2 Die Laufzeit von Algorithmen

Man kann eine formale Definition eines Algorithmus geben und dies werden wir in Abschnitt 15.1 auch tun. Solche formalen Modelle führen jedoch zu sehr langen und

auch aufwendigen Beschreibungen, sobald Algorithmen etwas komplizierter sind. In dieser Hinsicht ähnelt die Situation derjenigen bei mathematischen Beweisen: Obwohl der Begriff des Beweises formal gegeben werden kann, benutzt niemand einen solchen Formalismus, um einen Beweis zu schreiben, da ein solcher sehr lang und praktisch unlesbar wäre.

Aus diesem Grunde werden Algorithmen in diesem Buch nicht formal hingeschrieben. Damit sollte es dem nicht gänzlich unerfahrenen Leser möglich sein, die Algorithmen auf jedem Rechner ohne allzu viel Mühe zu implementieren.

Da wir bei der Abschätzung von Laufzeiten nicht an konstanten Faktoren interessiert sind, brauchen wir uns nicht auf ein konkretes Rechenmodell festzulegen. Wir zählen zwar die elementaren Schritte, sind aber nicht sonderlich daran interessiert, wie diese im Detail aussehen. Solche elementaren Schritte sind z.B. Variablenzuweisungen, der wahlfreie Zugriff (random access) auf eine Variable, deren Index in einer anderen Variable gespeichert ist, bedingte Sprünge (if – then – go to) und einfache arithmetische Operationen wie Addition, Subtraktion, Multiplikation, Division und Zahlenvergleich.

Ein Algorithmus besteht aus einer Menge von zulässigen Inputs und einer Folge von Instruktionen, die aus elementaren Schritten bestehen können, so dass der Rechenvorgang des Algorithmus für jeden zulässigen Input eine eindeutig bestimmte endliche Folge elementarer Schritte ist, die einen gewissen Output liefert. Normalerweise sind wir nicht mit einer bloß endlichen Berechnung zufrieden, sondern wünschen uns eine gute von der Inputgröße abhängige obere Schranke für die Anzahl der ausgeführten elementaren Schritte.

Der Input eines Algorithmus besteht üblicherweise aus einer Zahlenliste. Sind alle Zahlen der Liste ganze Zahlen, so können wir sie binär kodieren, wobei wir $O(\log(|a| + 2))$ Bits für eine ganze Zahl a benötigen. Für rationale Zahlen kodieren wir Zähler und Nenner separat. Die **Inputgröße** $\text{size}(x)$ einer Instanz x mit rationalen Daten ist die Gesamtanzahl der für die binäre Darstellung benötigten Bits.

Definition 1.3. Sei A ein Algorithmus, der Inputs aus einer Menge X annimmt. Sei ferner $f : \mathbb{N} \rightarrow \mathbb{R}_+$. Gibt es Konstanten $\alpha, \beta > 0$, so dass A für jeden Input $x \in X$ die Berechnung nach höchstens $\alpha f(\text{size}(x)) + \beta$ elementaren Schritten (arithmetische Operationen eingeschlossen) terminiert, dann sagen wir, dass A eine **Laufzeit** oder **Zeitkomplexität** von $O(f)$ hat.

Definition 1.4. Ein Algorithmus mit rationalem Input hat eine **polynomielle Laufzeit** oder ist **polynomiell**, falls es eine ganze Zahl k gibt, so dass erstens der Algorithmus eine $O(n^k)$ -Laufzeit hat, wobei n die Inputgröße ist, und zweitens alle in den Zwischenrechnungen auftretenden Zahlen mit $O(n^k)$ Bits gespeichert werden können.

Ein Algorithmus mit beliebigem Input ist **streu polynomiell**, falls es eine ganze Zahl k gibt, so dass der Algorithmus erstens für einen aus n Zahlen bestehenden Input eine $O(n^k)$ -Laufzeit hat und zweitens für einen rationalen Input polynomiell ist. Ist $k = 1$, so hat man einen **Linearzeit-Algorithmus**.

Ein Algorithmus mit polynomieller aber nicht streu polynomieller Laufzeit heißt **schwach polynomiell**.

Beachte, dass die Laufzeit für verschiedene Instanzen der gleichen Größe durchaus verschieden sein kann (dies war jedoch nicht der Fall für den WEG-ENUMERATIONS-ALGORITHMUS). Es ist naheliegend, die Worst-Case-Laufzeit zu betrachten, d.h. die Funktion $f : \mathbb{N} \rightarrow \mathbb{N}$, wobei $f(n)$ die maximale Laufzeit für Instanzen mit Inputgröße n ist. Für manche Algorithmen kennen wir die Wachstumsrate von f nicht, sondern haben nur eine obere Schranke.

Die Worst-Case-Laufzeit mag ein zu pessimistisches Maß sein, wenn diese schlimmsten Fälle (worst cases) nur selten auftreten. Manchmal ist eine durchschnittliche Laufzeit (average-case running time), verbunden mit einem stochastischen Modell, sinnvoller, wir werden aber hierauf nicht weiter eingehen.

Ist A ein Algorithmus, der für jeden Input $x \in X$ den Output $f(x) \in Y$ berechnet, so sagt man, dass A die **Funktion** $f : X \rightarrow Y$ **berechnet**. Wird eine Funktion mit einem polynomiellen Algorithmus berechnet, so sagt man, sie ist **in polynomieller Zeit berechenbar**.

Tabelle 1.1.

n	$100n \log n$	$10n^2$	$n^{3,5}$	$n \log n$	2^n	$n!$
10	3 μ s	1 μ s	3 μ s	2 μ s	1 μ s	4 ms
20	9 μ s	4 μ s	36 μ s	420 μ s	1 ms	76 Jahre
30	15 μ s	9 μ s	148 μ s	20 ms	1 s	$8 \cdot 10^{15}$ J.
40	21 μ s	16 μ s	404 μ s	340 ms	1 100 s	
50	28 μ s	25 μ s	884 μ s	4 s	13 Tage	
60	35 μ s	36 μ s	2 ms	32 s	37 Jahre	
80	50 μ s	64 μ s	5 ms	1 075 s	$4 \cdot 10^7$ J.	
100	66 μ s	100 μ s	10 ms	5 Stunden	$4 \cdot 10^{13}$ J.	
200	153 μ s	400 μ s	113 ms	12 Jahre		
500	448 μ s	2,5 ms	3 s	$5 \cdot 10^5$ J.		
1 000	1 ms	10 ms	32 s	$3 \cdot 10^{13}$ J.		
10^4	13 ms	1 s	28 Stunden			
10^5	166 ms	100 s	10 Jahre			
10^6	2 s	3 Stunden	3 169 J.			
10^7	23 s	12 Tage	10^7 J.			
10^8	266 s	3 Jahre	$3 \cdot 10^{10}$ J.			
10^{10}	9 Stunden	$3 \cdot 10^4$ J.				
10^{12}	46 Tage	$3 \cdot 10^8$ J.				

Polynomielle Algorithmen werden auch als „gut“ oder „effizient“ bezeichnet. Diese Bezeichnung geht auf Cobham [1964] und Edmonds [1965] zurück. Tabelle 1.1 legt diese Terminologie nahe, indem sie hypothetische Laufzeiten von Algorithmen mit unterschiedlichen Zeitkomplexitäten gegenüberstellt. Für verschiedene

Inputgrößen n geben wir die Laufzeiten für Algorithmen an, die $100n \log n$, $10n^2$, $n^{3,5}$, $n^{\log n}$, 2^n und $n!$ elementare Schritte benötigen, unter der Annahme, dass jeder elementare Schritt eine Nanosekunde dauert. In diesem Buch wird \log immer den Logarithmus zur Basis 2 bezeichnen.

Wie der Tabelle 1.1 zu entnehmen ist, sind polynomielle Algorithmen für genügend große Instanzen schneller. Ferner zeigt sie, dass mittelgroße Konstanten bei der Betrachtung des asymptotischen Wachstums der Laufzeit keine große Rolle spielen.

Tabelle 1.2 gibt die maximalen innerhalb einer Stunde lösbarer Inputgrößen für die obigen sechs hypothetischen Algorithmen an. Zeile (a) gilt, wie oben, unter der Annahme, dass jeder elementare Schritt eine Nanosekunde dauert, während (b) für eine zehnmal schnellere Maschine gilt. Polynomielle Algorithmen können größere Instanzen in akzeptabler Zeit erledigen. Hingegen würde eine sogar zehnfach verbesserte Rechnergeschwindigkeit keine signifikante Verbesserung in der Größe lösbarer Instanzen für Algorithmen mit exponentieller Laufzeit mit sich bringen, wohl aber für polynomielle Algorithmen.

Tabelle 1.2.

	$100n \log n$	$10n^2$	$n^{3,5}$	$n^{\log n}$	2^n	$n!$
(a)	$1,19 \cdot 10^9$	60 000	3 868	87	41	15
(b)	$10,8 \cdot 10^9$	189 737	7 468	104	45	16

Im Grunde suchen wir (streng) polynomielle Algorithmen, womöglich sogar Algorithmen mit linearer Laufzeit. Es gibt aber Probleme, für die es erwiesenermaßen keinen polynomiellen Algorithmus gibt, und auch Probleme, für die gar kein Algorithmus existiert. (Es gibt z. B. ein Problem, welches in endlicher Zeit gelöst werden kann, jedoch nicht polynomiell, nämlich die Entscheidungsfrage, ob ein sogenannter regulärer Ausdruck die leere Menge definiert; siehe Aho, Hopcroft und Ullman [1974]. Ein Problem, für welches kein Algorithmus existiert, nämlich das sogenannte HALTEPROBLEM, wird in Aufgabe 1, Kapitel 15, besprochen.)

Fast alle in diesem Buch betrachteten Probleme gehören jedoch einer der folgenden zwei Klassen an. Für jedes Problem der ersten Klasse gibt es einen polynomiellen Algorithmus. Für jedes Problem der zweiten Klasse besteht die offene Frage, ob es für dieses Problem einen polynomiellen Algorithmus gibt. Wir wissen aber: Hat nur ein einziges Problem in der zweiten Klasse einen polynomiellen Algorithmus, so gilt dies für alle Probleme in dieser Klasse. Eine präzise Formulierung dieser Aussage und ein Beweis werden in Kapitel 15 gegeben.

Das JOB-ZUORDNUNGSPROBLEM gehört zu der ersten Klasse, das BOHRPUNKT-PROBLEM jedoch zu der zweiten.

Die zwei oben definierten Problemklassen teilen dieses Buch grob gesehen in zwei Teile. Zunächst behandeln wir Probleme, für die es polynomielle Algorithmen gibt. Dann, von Kapitel 15 an, betrachten wir Probleme der anderen Klasse. Für

diese sind keine polynomiellen Algorithmen bekannt, jedoch gibt es oft viel bessere Methoden als die vollständige Enumeration. Ferner kann man für viele dieser Probleme (auch für das BOHRPUNKTPROBLEM) mit polynomiellen Algorithmen Näherungslösungen finden, die um nicht mehr als einen gewissen Prozentsatz vom Optimum abweichen.

1.3 Lineare Optimierungsprobleme

Nun wenden wir uns dem zweiten der beiden am Anfang geschilderten Probleme zu, nämlich dem JOB-ZUORDNUNGSPROBLEM. Dabei werden wir kurz auf einige zentrale Themen eingehen, die wir dann in späteren Kapiteln ausführlich besprechen werden.

Das JOB-ZUORDNUNGSPROBLEM unterscheidet sich grundsätzlich vom BOHRPUNKTPROBLEM, z.B. gibt es jetzt unendlich viele zulässige Lösungen für jede Instanz (außer für triviale Fälle). Das Job-Zuordnungsproblem können wir durch die Einführung einer Variable T für den Zeitpunkt der Beendigung aller Jobs folgendermaßen neu formulieren:

$$\begin{aligned} \min \quad & T \\ \text{bzgl.} \quad & \sum_{j \in S_i} x_{ij} = t_i \quad (i \in \{1, \dots, n\}) \\ & x_{ij} \geq 0 \quad (i \in \{1, \dots, n\}, j \in S_i) \\ & \sum_{i: j \in S_i} x_{ij} \leq T \quad (j \in \{1, \dots, m\}). \end{aligned} \tag{1.1}$$

Sowohl die Zahlen t_i als auch die Mengen S_i ($i = 1, \dots, n$) sind gegeben, und wir suchen nach Werten für die Variablen x_{ij} und T . Ein solches Optimierungsproblem mit einer linearen Zielfunktion und linearen Nebenbedingungen heißt ein **lineares Programm** oder kurz ein **LP**. Es ist leicht zu sehen, dass die Menge der zulässigen Lösungen von (1.1) eine konvexe Menge (ein sogenanntes **Polyeder**) ist, und man kann beweisen, dass es immer eine optimale Lösung gibt, die eine der endlich vielen Extrempunkte dieser Menge ist. Somit kann ein LP theoretisch auch mittels vollständiger Enumeration gelöst werden. Es gibt jedoch viel bessere Verfahren, wie wir später sehen werden.

Obwohl es mehrere Algorithmen zur Lösung allgemeiner LPs gibt, sind solche allgemeinen Verfahren meistens weniger effizient als spezielle Methoden, die die besondere Struktur des Problems ausnutzen. Im vorliegenden Fall ist es zweckmäßig, die Mengen S_i , $i = 1, \dots, n$, mittels eines **Graphen** darzustellen. Für jeden Job i und jeden Arbeiter j haben wir einen Punkt (Knoten), und wir verbinden Arbeiter j und Job i mit einer Kante, falls er an diesem Job arbeiten kann (d.h. falls $j \in S_i$). Graphen sind eine fundamentale kombinatorische Struktur; viele kombinatorische Optimierungsprobleme lassen sich auf natürliche Weise mittels Graphen darstellen.

Nehmen wir zunächst einmal an, dass die für jeden Job benötigte Zeit eine Stunde beträgt. Können wir dann alle Jobs innerhalb von einer Stunde beenden? Hier suchen wir Zahlen x_{ij} ($i \in \{1, \dots, n\}$, $j \in S_i$), so dass $0 \leq x_{ij} \leq 1$ für alle i und j , $\sum_{j \in S_i} x_{ij} = 1$ für $i = 1, \dots, n$, und $\sum_{i:j \in S_i} x_{ij} \leq 1$ für $j = 1, \dots, n$. Man kann zeigen: Falls es eine solche Lösung gibt, so gibt es auch eine ganzzahlige, d. h. alle x_{ij} sind 0 oder 1. Dies bedeutet, dass man jeden Job einem Arbeiter zuordnet und keinem Arbeiter mehr als einen Job. Graphentheoretisch ausgedrückt suchen wir ein alle Jobs überdeckendes sogenanntes **Matching**. Das Problem, optimale Matchings zu finden, ist eines der bekanntesten kombinatorischen Optimierungsprobleme.

Die Grundlagen der Graphentheorie und der linearen Optimierung werden wir in den Kapiteln 2 und 3 behandeln. In Kapitel 4 beweisen wir, dass LPs in polynomieller Zeit gelöst werden können, und in Kapitel 5 besprechen wir ganzzahlige Polyeder. In den darauf folgenden Kapiteln werden wir dann einige klassische kombinatorische Optimierungsprobleme detailliert behandeln.

1.4 Sortieren

Wir möchten dieses Kapitel mit der Betrachtung eines Spezialfalls des BOHR-PUNKTPROBLEMS beschließen, nämlich des Falles, wo alle zu bohrenden Löcher auf einer Geraden liegen. Also werden die Bohrpunkte mittels nur einer Zahl p_i , $i = 1, \dots, n$, festgelegt. Hier ist die Lösung besonders einfach: Man braucht die Bohrpunkte nur nach ihrer Koordinate zu sortieren, und der Bohrer bewegt sich dann schrittweise von links nach rechts. Obwohl es $n!$ Permutationen gibt, brauchen wir offensichtlich nicht alle zu betrachten, um die optimale Reihenfolge, d. h. die sortierte Liste, zu bestimmen. Es ist sehr leicht, n Zahlen in aufsteigender Reihenfolge mittels eines Sortierverfahrens mit $O(n^2)$ Laufzeit zu sortieren.

Etwas schwieriger ist es, n Zahlen in $O(n \log n)$ -Zeit zu sortieren. Es gibt einige Algorithmen, die dies bewerkstelligen; hier besprechen wir den bekannten MERGE-SORT-ALGORITHMUS, der folgendermaßen abläuft. Zuerst wird die Liste in zwei etwa gleich große Teillisten geteilt. Dann wird jede dieser beiden Teillisten sortiert (dies geschieht rekursiv mit demselben Algorithmus). Schließlich werden die beiden sortierten Teillisten wieder zusammengeführt („merged“). Diese allgemeine Strategie, oft „divide and conquer“ genannt, ist sehr nützlich. In Abschnitt 17.1 befindet sich ein weiteres Beispiel.

Wir haben rekursive Algorithmen bisher nicht besprochen. In der Tat brauchen wir dies auch nicht zu tun, da jeder rekursive Algorithmus ohne Erhöhung der Laufzeit in sequentieller Form dargestellt werden kann. Manche Algorithmen lassen sich jedoch leichter mittels Rekursion formulieren (und implementieren), weswegen wir manchmal davon Gebrauch machen werden.

MERGE-SORT-ALGORITHMUS

Input: Eine Liste a_1, \dots, a_n reeller Zahlen.

Output: Eine Permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ mit $a_{\pi(i)} \leq a_{\pi(i+1)}$ für alle $i = 1, \dots, n - 1$.

-
- ① If $n = 1$ then setze $\pi(1) := 1$ und stop (return π).
 - ② Setze $m := \lfloor \frac{n}{2} \rfloor$.
Sei $\rho := \text{MERGE-SORT}(a_1, \dots, a_m)$.
Sei $\sigma := \text{MERGE-SORT}(a_{m+1}, \dots, a_n)$.
 - ③ Setze $k := 1, l := 1$.
While $k \leq m$ und $l \leq n - m$ **do**:
 If $a_{\rho(k)} \leq a_{m+\sigma(l)}$ **then** setze $\pi(k + l - 1) := \rho(k)$ und $k := k + 1$
 else setze $\pi(k + l - 1) := m + \sigma(l)$ und $l := l + 1$.
While $k \leq m$ **do**: Setze $\pi(k + l - 1) := \rho(k)$ und $k := k + 1$.
While $l \leq n - m$ **do**: Setze $\pi(k + l - 1) := m + \sigma(l)$ und $l := l + 1$.
-

Als Beispiel betrachten wir die Liste „69, 32, 56, 75, 43, 99, 28“. Als erstes teilt der Algorithmus diese Liste in zwei etwa gleich große Teile, nämlich „69, 32, 56“ und „75, 43, 99, 28“, und sortiert diese dann rekursiv. Dies liefert die Permutationen $\rho = (2, 3, 1)$ und $\sigma = (4, 2, 1, 3)$, entsprechend den sortierten Teillisten „32, 56, 69“ und „28, 43, 75, 99“. Diese beiden sortierten Teillisten werden nun wie folgt zusammengefügt („merged“):

$$\begin{array}{llll}
 & k := 1, & l := 1 \\
 \rho(1) = 2, & \sigma(1) = 4, & a_{\rho(1)} = 32, & \pi(1) := 7, \quad l := 2 \\
 \rho(1) = 2, & \sigma(2) = 2, & a_{\rho(1)} = 32, & \pi(2) := 2, \quad k := 2 \\
 \rho(2) = 3, & \sigma(2) = 2, & a_{\rho(2)} = 56, & a_{\sigma(2)} = 43, \quad \pi(3) := 5, \quad l := 3 \\
 \rho(2) = 3, & \sigma(3) = 1, & a_{\rho(2)} = 56, & a_{\sigma(3)} = 75, \quad \pi(4) := 3, \quad k := 3 \\
 \rho(3) = 1, & \sigma(3) = 1, & a_{\rho(3)} = 69, & a_{\sigma(3)} = 75, \quad \pi(5) := 1, \quad k := 4 \\
 & \sigma(3) = 1, & & a_{\sigma(3)} = 75, \quad \pi(6) := 4, \quad l := 4 \\
 & \sigma(4) = 3, & & a_{\sigma(4)} = 99, \quad \pi(7) := 6, \quad l := 5
 \end{array}$$

Satz 1.5. Der MERGE-SORT-ALGORITHMUS arbeitet korrekt und hat $O(n \log n)$ -Laufzeit.

Beweis: Die Korrektheit ist klar. Sei $T(n)$ die Laufzeit (d.h. die Anzahl der Schritte), die für Instanzen bestehend aus n Zahlen benötigt wird; beachte, dass $T(1) = 1$ und $T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 3n + 6$. (Die Konstanten in dem Term $3n + 6$ hängen von der genauen Definition eines Rechenschrittes ab, sind aber nicht wichtig.)

Wir behaupten nun, dass $T(n) \leq 12n \log n + 1$. Da dies für $n = 1$ trivial ist, benutzen wir Induktion über n . Angenommen, die Ungleichung gelte für $1, \dots, n - 1$. Dann folgt für $n \geq 2$:

$$\begin{aligned}
 T(n) &\leq 12 \left\lfloor \frac{n}{2} \right\rfloor \log \left(\frac{2}{3} n \right) + 1 + 12 \left\lceil \frac{n}{2} \right\rceil \log \left(\frac{2}{3} n \right) + 1 + 3n + 6 \\
 &= 12n(\log n + 1 - \log 3) + 3n + 8 \\
 &\leq 12n \log n - \frac{13}{2}n + 3n + 8 \leq 12n \log n + 1,
 \end{aligned}$$

da $\log 3 \geq \frac{37}{24}$. □

Natürlich sortiert dieser Algorithmus die Elemente einer jeden vollständig geordneten Menge, unter der Annahme, dass je zwei Elemente in konstanter Zeit miteinander verglichen werden können. Kann es überhaupt einen schnelleren Algorithmus geben, etwa mit linearer Laufzeit? Angenommen, der einzige Weg zur Information bezüglich der unbekannten Reihenfolge führt über den Vergleich zweier Elemente. Dann kann man zeigen, dass kein Algorithmus im schlechtesten Fall (worst case) mit weniger als $\Theta(n \log n)$ Vergleichen auskommen kann. Das Ergebnis eines Vergleiches kann man als Null bzw. Eins angeben. Dann ist das Ergebnis aller von dem Algorithmus vorgenommenen Vergleiche ein 0-1-String (eine endliche Folge von Nullen und Einsen). Beachte, dass verschiedene Inputreihenfolgen zu verschiedenen 0-1-Strings führen, sonst könnte der Algorithmus nicht zwischen diesen zwei Reihenfolgen unterscheiden. Für einen gegebenen Input von n Elementen gibt es $n!$ verschiedene Reihenfolgen, also gibt es auch $n!$ verschiedene 0-1-Strings. Da die Anzahl der 0-1-Strings mit Länge kürzer als $\lfloor \frac{n}{2} \log \frac{n}{2} \rfloor$ gleich $2^{\lfloor \frac{n}{2} \log \frac{n}{2} \rfloor} - 1 < 2^{\frac{n}{2} \log \frac{n}{2}} = (\frac{n}{2})^{\frac{n}{2}} \leq n!$ ist, folgt, dass die maximale Länge der 0-1-Strings, somit auch der Berechnung, mindestens $\frac{n}{2} \log \frac{n}{2} = \Theta(n \log n)$ ist.

In dieser Hinsicht ist also die Laufzeit des MERGE-SORT-ALGORITHMUS bis auf einen konstanten Faktor optimal. Es gibt jedoch ein Sortierverfahren für ganze Zahlen (oder auch für die lexikographische Sortierung von Strings), dessen Laufzeit linear in der Inputgröße ist; siehe Aufgabe 8. Ein Algorithmus zum Sortieren von n ganzen Zahlen mit $O(n \log \log n)$ -Laufzeit ist von Han [2004] vorgeschlagen worden.

Untere Schranken wie die obige sind für nur sehr wenige Probleme bekannt (abgesehen von trivialen linearen Schranken). Oft ist eine Einschränkung der Menge der Operationen notwendig, um eine superlineare untere Schranke herzuleiten.

Aufgaben

1. Man beweise für alle $n \in \mathbb{N}$, dass

$$e \left(\frac{n}{e} \right)^n \leq n! \leq e n \left(\frac{n}{e} \right)^n.$$

Hinweis: Man benutze $1 + x \leq e^x$ für alle $x \in \mathbb{R}$.

2. Man beweise, dass $\log(n!) = \Theta(n \log n)$.
 3. Man beweise, dass $n \log n = O(n^{1+\epsilon})$ für jedes $\epsilon > 0$.
 4. Man zeige, dass der WEG-ENUMERATIONS-ALGORITHMUS $O(n \cdot n!)$ Laufzeit hat.
 5. Man zeige, dass es genau dann einen polynomiellen Algorithmus für das BOHRPROBLEM mit d gleich dem ℓ_1 -Abstand gibt, wenn es einen für das BOHRPROBLEM mit d gleich dem ℓ_∞ -Abstand gibt.
- Bemerkung:* Die Existenz beider ist unwahrscheinlich, da diese Probleme beide NP-schwer sind (bewiesen von Garey, Graham und Johnson [1976], siehe Kapitel 15).

6. Angenommen, man hat einen Algorithmus mit $\Theta(n(t + n^{1/t}))$ -Laufzeit, wobei n die Inputlänge und t ein beliebig wählbarer positiver Parameter ist. Wie soll man t (in Abhängigkeit von n) wählen, so dass die Laufzeit (als Funktion von n) eine minimale Wachstumsrate hat?
7. Seien s und t binäre Strings, beide mit Länge m . Man sagt, s ist lexikographisch kleiner als t , falls es einen Index $j \in \{1, \dots, m\}$ gibt, so dass $s_i = t_i$ für $i = 1, \dots, j-1$ und $s_j < t_j$. Man möchte nun n gegebene Strings der Länge m lexikographisch sortieren. Man beweise, dass es hierfür einen Algorithmus mit linearer Laufzeit gibt (d. h. mit $O(nm)$ -Laufzeit).

Hinweis: Man gruppieren die Strings nach dem ersten Bit und sortiere jede Gruppe.
8. Man beschreibe einen Algorithmus zum Sortieren einer Liste natürlicher Zahlen a_1, \dots, a_n , in linearer Laufzeit, d. h. zur Bestimmung einer Permutation π mit $a_{\pi(i)} \leq a_{\pi(i+1)}$ ($i = 1, \dots, n-1$) in $O(\log(a_1 + 1) + \dots + \log(a_n + 1))$ -Laufzeit.

Hinweis: Zunächst sortiere man die Strings, die die Zahlen kodieren, der Länge nach. Dann wende man den Algorithmus von Aufgabe 7 an.

Bemerkung: Das diesem Algorithmus und dem von Aufgabe 7 zugrunde liegende Verfahren ist bekannt als „radix sorting“.

Literatur

Allgemeine Literatur:

- Cormen, T.H., Leiserson, C.E., Rivest, R.L., und Stein, C. [2009]: Introduction to Algorithms. 3. Aufl. MIT Press, Cambridge 2009
- Hougardy, S., und Vygen, J. [2016]: Algorithmische Mathematik. Springer, Berlin 2016
- Knuth, D.E. [1968]: The Art of Computer Programming; Vol. 1. Fundamental Algorithms. Addison-Wesley, Reading 1968 (3. Aufl. 1997)
- Mehlhorn, K., und Sanders, P. [2008]: Algorithms and Data Structures: The Basic Toolbox. Springer, Berlin 2008

Zitierte Literatur:

- Aho, A.V., Hopcroft, J.E., und Ullman, J.D. [1974]: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading 1974
- Cobham, A. [1964]: The intrinsic computational difficulty of functions. Proceedings of the 1964 Congress for Logic Methodology and Philosophy of Science (Y. Bar-Hillel, Hrsg.), North-Holland, Amsterdam 1964, pp. 24–30
- Edmonds, J. [1965]: Paths, trees, and flowers. Canadian Journal of Mathematics 17 (1965), 449–467
- Garey, M.R., Graham, R.L., und Johnson, D.S. [1976]: Some NP-complete geometric problems. Proceedings of the 8th Annual ACM Symposium on the Theory of Computing (1976), 10–22
- Han, Y. [2004]: Deterministic sorting in $O(n \log \log n)$ time and linear space. Journal of Algorithms 50 (2004), 96–105
- Stirling, J. [1730]: Methodus Differentialis. London 1730



2 Graphen

Graphen sind fundamentale kombinatorische Strukturen, die überall in diesem Buch vorkommen. In diesem Kapitel werden wir nicht nur die grundlegenden Definitionen und die Standardnotation einführen, sondern auch einige fundamentale Sätze und Algorithmen beweisen bzw. beschreiben.

Nach den einführenden Definitionen in Abschnitt 2.1 werden wir fundamentale in diesem Buch sehr oft wiederkehrende Objekte betrachten: Bäume, Kreise und Schnitte. In Abschnitt 2.2 zeigen wir einige wichtige Eigenschaften und betrachten baumartige Mengensysteme. Erste Graphenalgorithmen, nämlich zur Bestimmung von Zusammenhangskomponenten und von starken Zusammenhangskomponenten, werden in Abschnitt 2.3 vorgestellt. In Abschnitt 2.4 beweisen wir Eulers Satz über geschlossene, jede Kante genau einmal durchlaufende Spaziergänge. Schließlich werden wir in den Abschnitten 2.5 und 2.6 diejenigen Graphen betrachten, die kreuzungsfrei in der Ebene dargestellt werden können.

2.1 Grundlegende Definitionen

Ein **ungerichteter Graph** ist ein Tripel (V, E, Ψ) , wobei V eine nichtleere endliche Menge ist, E eine endliche Menge und $\Psi : E \rightarrow \{X \subseteq V : |X| = 2\}$. Ein **gerichteter Graph** oder **Digraph** ist ein Tripel (V, E, Ψ) , wobei V eine nichtleere endliche Menge ist, E eine endliche Menge und $\Psi : E \rightarrow \{(v, w) \in V \times V : v \neq w\}$. Ein **Graph** ist entweder ein ungerichteter Graph oder ein Digraph. Die Elemente von V heißen **Knoten** und diejenigen von E **Kanten**.

Zwei Kanten e und e' mit $e \neq e'$ und $\Psi(e) = \Psi(e')$ heißen **parallel**. Graphen ohne parallele Kanten heißen **einfach**. In einfachen Graphen identifiziert man eine Kante e gewöhnlich mit ihrem Bild $\Psi(e)$ und schreibt $G = (V(G), E(G))$, wobei $E(G) \subseteq \binom{X}{2} := \{X \subseteq V(G) : |X| = 2\}$ oder $E(G) \subseteq V(G) \times V(G)$. Wir werden uns oft dieser einfacheren Notation bedienen, selbst wenn parallele Kanten vorhanden sind, wobei die „Menge“ $E(G)$ dann einige „identische“ Elemente enthält. Die Anzahl der Kanten wird mit $|E(G)|$ bezeichnet und für zwei Kantenmengen E und F gilt immer $|E \cup F| = |E| + |F|$, auch wenn dabei parallele Kanten entstehen. Wir schreiben $e = \{v, w\}$ bzw. $e = (v, w)$ für jede Kante e mit $\Psi(e) = \{v, w\}$ bzw. $\Psi(e) = (v, w)$.

Man sagt: Eine Kante $e = \{v, w\}$ oder $e = (v, w)$ **verbindet** v und w , und in diesem Fall sind v und w **benachbart**. Der Knoten v ist ein **Nachbar** von w (und

umgekehrt). Es sind v und w die **Endknoten** von e . Ist v ein Endknoten einer Kante e , so sagt man: v ist **inzipient** mit e . Im gerichteten Fall sagt man: (v, w) **beginnt** in v und **endet** in w , oder auch: (v, w) geht **von** v **nach** w . Gelegentlich sprechen wir auch von einer in w **ankommenden** Kante. Zwei Kanten, die mindestens einen Endknoten gemeinsam haben, heißen **benachbart**.

Für manche der obigen graphentheoretischen Begriffe gibt es alternative Bezeichnungen. Knoten werden auch Ecken genannt und Kanten auch Linien oder Bögen, Letzteres besonders in gerichteten Graphen. In manchen Texten entspricht ein Graph unserem einfachen ungerichteten Graphen, und ein Multigraph erlaubt zusätzlich noch parallele Kanten. Manchmal erlaubt man auch Kanten, deren Endknoten übereinstimmen, sogenannte **Schleifen**. Wir werden Letztere jedoch nicht betrachten, ohne besonders darauf hinzuweisen.

Für einen Digraphen G betrachten wir gelegentlich den **zugrunde liegenden ungerichteten Graphen**, d. h. den ungerichteten Graphen G' mit derselben Knotenmenge wie G und den Kanten $\{v, w\}$ für alle Kanten (v, w) von G (d. h. $|E(G')| = |E(G)|$). Man nennt G auch eine **Orientierung** von G' .

Ein **Teilgraph** eines Graphen $G = (V(G), E(G))$ ist ein Graph $H = (V(H), E(H))$ mit $V(H) \subseteq V(G)$ und $E(H) \subseteq E(G)$. Man sagt auch: G enthält einen Teilgraphen H . Man nennt H einen **induzierten Teilgraphen** von G , wenn H ein Teilgraph von G ist und $E(H) = \{(x, y) \in E(G) : x, y \in V(H)\}$ bzw. $E(H) = \{(x, y) \in E(G) : x, y \in V(H)\}$. Dieses H ist der **von** $V(H)$ **induzierte** Teilgraph von G . Dafür schreibt man auch $H = G[V(H)]$. Ein Teilgraph H von G heißt **aufspannend**, wenn $V(H) = V(G)$. Ferner wird öfters der Begriff eines **maximalen (minimalen)** Teilgraphen H von G bezüglich einer bestimmten Eigenschaft vorkommen; dies bedeutet im ersten Fall, dass H diese Eigenschaft hat, aber kein echter Teilgraph eines Teilgraphen von G mit dieser Eigenschaft ist, und im zweiten Fall, dass H , aber kein echter Teilgraph von H , diese Eigenschaft hat.

Ist $v \in V(G)$, so bezeichnet $G - v$ den von $V(G) \setminus \{v\}$ induzierten Teilgraphen von G . Ist $e \in E(G)$, so ist $G - e$ der Teilgraph $(V(G), E(G) \setminus \{e\})$ von G . Diese Notation werden wir auch zur Entfernung einer Knoten- oder Kantenmenge X benutzen; dann schreiben wir $G - X$. Das Hinzufügen einer neuen Kante e ergibt den Graphen $G + e := (V(G), E(G) \cup \{e\})$. Sind G und H zwei Graphen, so bezeichnet $G+H$ den Graphen mit $V(G+H) = V(G) \cup V(H)$ und $E(G+H)$ gleich der disjunkten Vereinigung von $E(G)$ und $E(H)$ (es können dabei parallele Kanten entstehen). Eine Familie von Graphen heißt **knotendisjunkt** bzw. **kantendisjunkt**, falls deren Knoten- bzw. Kantenmengen paarweise disjunkt sind.

Zwei Graphen G und H sind **isomorph**, wenn es Bijektionen $\Phi_V : V(G) \rightarrow V(H)$ und $\Phi_E : E(G) \rightarrow E(H)$ gibt, so dass $\Phi_E((v, w)) = (\Phi_V(v), \Phi_V(w))$ für alle $(v, w) \in E(G)$ oder, im ungerichteten Fall, $\Phi_E(\{v, w\}) = \{\Phi_V(v), \Phi_V(w)\}$ für alle $\{v, w\} \in E(G)$. Normalerweise werden wir zwischen isomorphen Graphen nicht unterscheiden; enthält G z. B. einen zu H isomorphen Teilgraphen, so werden wir kurzerhand sagen: G enthält H .

Sei G ein ungerichteter Graph und $X \subseteq V(G)$. Eine **Kontraktion** (oder **Schrumpfung**) von X beinhaltet erstens das Weglassen der Knoten von X und der Kanten aus $G[X]$ und zweitens das Hinzufügen eines neuen Knotens x und Ersetzen einer jeden Kante $\{v, w\}$ mit $v \in X$ und $w \notin X$ durch die Kante $\{x, w\}$ (es können dabei parallele Kanten entstehen); analoges gilt für Digraphen. Diese Operation wird mit G/X bezeichnet.

Sei G ein Graph und $X, Y \subseteq V(G)$. Ist G ungerichtet, so setzt man $E(X, Y) := \{\{x, y\} \in E(G) : x \in X \setminus Y, y \in Y \setminus X\}$. Ist andererseits G gerichtet, so setzt man $E^+(X, Y) := \{(x, y) \in E(G) : x \in X \setminus Y, y \in Y \setminus X\}$. Für einen ungerichteten Graphen G und $X \subseteq V(G)$ setzt man ferner $\delta(X) := E(X, V(G) \setminus X)$. Die **Menge der Nachbarn** von X ist $\Gamma(X) := \{v \in V(G) \setminus X : E(X, \{v\}) \neq \emptyset\}$. Für einen Digraphen G und $X \subseteq V(G)$ setzt man $\delta^+(X) := E^+(X, V(G) \setminus X)$, $\delta^-(X) := \delta^+(V(G) \setminus X)$ und $\delta(X) := \delta^+(X) \cup \delta^-(X)$. Falls nötig, werden wir den betreffenden Graphen G durch ein tiefgestelltes G (z. B. $\delta_G(X)$) angeben.

Für einelementige Knotenmengen $\{v\}$ ($v \in V(G)$) (im Englischen: **singletons**) setzt man $\delta(v) := \delta(\{v\})$, $\Gamma(v) := \Gamma(\{v\})$, $\delta^+(v) := \delta^+(\{v\})$ und $\delta^-(v) := \delta^-(\{v\})$. Der **Grad** eines Knotens v ist die Anzahl $|\delta(v)|$ der mit v inzidenten Kanten. Für einen gerichteten Graphen ist der **Eingangsgrad** gleich $|\delta^-(v)|$, der **Ausgangsgrad** gleich $|\delta^+(v)|$ und der Grad gleich $|\delta^+(v)| + |\delta^-(v)|$. Ein Knoten mit Grad 0 heißt **isoliert**. Ein Graph, für den alle Knoten den Grad k haben, heißt **k -regulär**.

Für jeden Graphen G gilt $\sum_{v \in V(G)} |\delta(v)| = 2|E(G)|$. Insbesondere ist die Anzahl der Knoten mit ungeradem Grad eine gerade Zahl. Für einen Digraphen gilt $\sum_{v \in V(G)} |\delta^+(v)| = \sum_{v \in V(G)} |\delta^-(v)|$. Zum Beweis dieser Aussagen beachte man, dass jede Kante auf beiden Seiten der ersten Gleichung zweimal gezählt wird, jedoch nur einmal im Falle der zweiten Gleichung. Mit nur wenig zusätzlichem Aufwand erhalten wir die folgenden nützlichen Aussagen:

Lemma 2.1. Für einen Digraphen G und zwei Mengen $X, Y \subseteq V(G)$ gilt:

- (a) $|\delta^+(X)| + |\delta^+(Y)| = |\delta^+(X \cap Y)| + |\delta^+(X \cup Y)| + |E^+(X, Y)| + |E^+(Y, X)|$;
- (b) $|\delta^-(X)| + |\delta^-(Y)| = |\delta^-(X \cap Y)| + |\delta^-(X \cup Y)| + |E^+(X, Y)| + |E^+(Y, X)|$.

Für einen ungerichteten Graphen G und zwei Mengen $X, Y \subseteq V(G)$ gilt:

- (c) $|\delta(X)| + |\delta(Y)| = |\delta(X \cap Y)| + |\delta(X \cup Y)| + 2|E(X, Y)|$;
- (d) $|\delta(X)| + |\delta(Y)| = |\delta(X \setminus Y)| + |\delta(Y \setminus X)| + 2|E(X \cap Y, V(G) \setminus (X \cup Y))|$;
- (e) $|\Gamma(X)| + |\Gamma(Y)| \geq |\Gamma(X \cap Y)| + |\Gamma(X \cup Y)|$.

Beweis: Alle vier Aussagen lassen sich durch einfaches Abzählen beweisen. Sei $Z := V(G) \setminus (X \cup Y)$.

Zum Beweis von (a): Beachte, dass $|\delta^+(X)| + |\delta^+(Y)| = |E^+(X, Z)| + |E^+(X, Y \setminus X)| + |E^+(Y, Z)| + |E^+(Y, X \setminus Y)| = |E^+(X \cup Y, Z)| + |E^+(X \cap Y, Z)| + |E^+(X, Y \setminus X)| + |E^+(Y, X \setminus Y)| = |\delta^+(X \cup Y)| + |\delta^+(X \cap Y)| + |E^+(X, Y)| + |E^+(Y, X)|$. Es folgt (b) aus (a) mittels Umorientierung jeder Kante (ersetze (v, w) durch (w, v)). Es folgt (c) aus (a) mittels Ersetzen einer jeden Kante $\{v, w\}$ durch ein Paar entgegengesetzter gerichteter Kanten (v, w) und (w, v) . Es folgt (d) aus (c) mittels Ersetzen von Y durch $V(G) \setminus Y$. Zum Beweis von (e): Beachte, dass

$$|\Gamma(X)| + |\Gamma(Y)| = |\Gamma(X \cup Y)| + |\Gamma(X) \cap \Gamma(Y)| + |\Gamma(X) \cap Y| + |\Gamma(Y) \cap X| \geq |\Gamma(X \cup Y)| + |\Gamma(X \cap Y)|. \quad \square$$

Eine Funktion $f : 2^U \rightarrow \mathbb{R}$ (wobei U eine endliche Menge ist und 2^U die Potenzmenge von U bezeichnet) heißt

- **submodular**, falls $f(X \cap Y) + f(X \cup Y) \leq f(X) + f(Y)$ für alle $X, Y \subseteq U$;
- **supermodular**, falls $f(X \cap Y) + f(X \cup Y) \geq f(X) + f(Y)$ für alle $X, Y \subseteq U$;
- **modular**, falls $f(X \cap Y) + f(X \cup Y) = f(X) + f(Y)$ für alle $X, Y \subseteq U$.

Also folgt aus Lemma 2.1, dass $|\delta^+|$, $|\delta^-|$, $|\delta|$ und $|\Gamma|$ submodulare Funktionen sind. Dies wird später gebraucht.

Ein **vollständiger Graph** ist ein einfacher ungerichteter Graph mit der Eigenschaft, dass je zwei Knoten benachbart sind. Den vollständigen Graphen mit n Knoten bezeichnet man mit K_n . Der **Komplementgraph** eines einfachen ungerichteten Graphen G ist der Graph H mit $V(G) = V(H)$ und der Eigenschaft, dass $G + H$ vollständig ist.

Ein **Matching** in einem ungerichteten Graphen G ist eine Menge paarweise disjunkter Kanten (d. h. ihre Endknoten sind alle verschieden). Eine **Knotenüberdeckung** von G ist eine Menge $S \subseteq V(G)$ von Knoten mit der Eigenschaft, dass jede Kante von G mit mindestens einem Knoten in S inzident ist. Eine **Kantenüberdeckung** von G ist eine Menge $F \subseteq E(G)$ von Kanten mit der Eigenschaft, dass jeder Knoten von G mit mindestens einer Kante in F inzident ist. Eine **stabile Menge** in G ist eine Menge paarweise nicht benachbarter Knoten. Ein Graph mit leerer Kantenmenge heißt **leer**. Eine **Clique** in G ist eine Menge paarweise benachbarter Knoten.

Proposition 2.2. Sei G ein Graph und $X \subseteq V(G)$. Dann sind die folgenden drei Aussagen äquivalent:

- (a) X ist eine Knotenüberdeckung von G ,
- (b) $V(G) \setminus X$ ist eine stabile Menge in G ,
- (c) $V(G) \setminus X$ ist eine Clique im Komplementgraphen von G . \square

Sei \mathcal{F} eine Familie von Mengen. Dann sagt man: F ist ein **inklusionsminimales** Element von \mathcal{F} , wenn F , aber keine echte Teilmenge von F , in \mathcal{F} enthalten ist. Analog sagt man: F ist **inklusionsmaximal** in \mathcal{F} , wenn $F \in \mathcal{F}$ und F keine echte Teilmenge eines Elementes von \mathcal{F} ist.

Man beachte, dass z. B. eine inklusionsminimale Knotenüberdeckung im Allgemeinen keine Knotenüberdeckung minimaler Kardinalität (d. h. nicht **kardinalitätsminimal**) ist (siehe z. B. den Graphen in Abb. 13.1). Auch ist ein inklusionsmaximales Matching im Allgemeinen kein Matching maximaler Kardinalität (d. h. nicht **kardinalitätsmaximal**). Wie man ein Matching, eine stabile Menge oder eine Clique maximaler Kardinalität in einem ungerichteten Graphen findet, oder auch eine Knoten- oder Kantenüberdeckung minimaler Kardinalität, wird in späteren Kapiteln eine wichtige Rolle spielen.

Der **Kantengraph** eines einfachen ungerichteten Graphen G ist der Graph $(E(G), F)$ mit $F = \{\{e_1, e_2\} : e_1, e_2 \in E(G), |e_1 \cap e_2| = 1\}$. Offensichtlich entsprechen den Matchings in einem Graphen G die stabilen Mengen im Kantengraphen von G .

Sei nun G ein Graph, gerichtet oder auch ungerichtet. Eine **Kantenfolge** W in G (**von** v_1 **nach** v_{k+1}) ist eine Folge $v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1}$, wobei $k \geq 0$ und $e_i = (v_i, v_{i+1}) \in E(G)$ oder $e_i = \{v_i, v_{i+1}\} \in E(G)$ für $i = 1, \dots, k$. Gilt zusätzlich $e_i \neq e_j$ für alle $1 \leq i < j \leq k$, so heißt W ein **Spaziergang** in G . Gilt auch noch $v_1 = v_{k+1}$, so heißt der Spaziergang W **geschlossen**.

Ein **Weg** ist ein Graph $P = (\{v_1, \dots, v_{k+1}\}, \{e_1, \dots, e_k\})$ mit der Eigenschaft, dass $v_i \neq v_j$ für $1 \leq i < j \leq k + 1$ und die Folge $v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1}$ ein Spaziergang ist. Es wird P auch ein **Weg von v_1 nach v_{k+1}** oder **zu v_{k+1}** oder auch ein v_1 - v_{k+1} -Weg genannt. Die Knoten v_1 und v_{k+1} sind die **Endknoten** von P , und die Knoten v_2, \dots, v_k sind die **inneren Knoten** von P . Für $x, y \in V(P)$ bezeichnen wir mit $P_{[x,y]}$ den (eindeutig bestimmten) Teilgraphen von P , der ein x - y -Weg ist.

Für jeden Graphen G und Knoten $v, w \in V(G)$ gibt es genau dann eine von v nach w laufende Kantenfolge in G , wenn es einen v - w -Weg in G (d.h. in G als Teilgraph enthalten) gibt. Für ungerichtete Graphen wird dadurch eine Äquivalenzrelation definiert.

Ein **Kreis** ist ein Graph $(\{v_1, \dots, v_k\}, \{e_1, \dots, e_k\})$ mit der Eigenschaft, dass die Folge $v_1, e_1, v_2, \dots, v_k, e_k, v_1$ ein (geschlossener) Spaziergang mit $k \geq 2$ und $v_i \neq v_j$ für $1 \leq i < j \leq k$ ist. Ein einfacher Induktionsbeweis zeigt, dass die Kantenmenge eines geschlossenen Spaziergangs in eine Familie von Kreis-Kantenmengen zerlegt werden kann.

Ein **ungerichteter Weg** bzw. ein **ungerichteter Kreis** in einem Digraphen ist ein Teilgraph im zugrunde liegenden Graphen, der einem Weg bzw. einem Kreis entspricht.

Die **Länge** eines Weges oder Kreises ist die Anzahl seiner Kanten. Ist ein Weg oder Kreis ein Teilgraph von G , so spricht man von einem Weg oder Kreis in G . Ein aufspannender Weg in G heißt **hamiltonscher Weg** und ein aufspannender Kreis in G heißt **Hamilton-Kreis**. Enthält ein Graph einen Hamilton-Kreis, so heißt er **hamiltonscher Graph**.

Für zwei Knoten v und w bezeichnen wir mit $\text{dist}(v, w)$ oder $\text{dist}_G(v, w)$ die Länge eines kürzesten v - w -Wege (die **Distanz** von v nach w) in G . Gibt es gar keinen v - w -Weg, d.h. w ist von v aus nicht **erreichbar**, so setzen wir $\text{dist}(v, w) := \infty$. In ungerichteten Graphen gilt $\text{dist}(v, w) = \text{dist}(w, v)$ für alle $v, w \in V(G)$.

Wir werden es oft mit einer (Kosten-) oder Gewichtsfunktion $c : E(G) \rightarrow \mathbb{R}$ zu tun haben. Für $F \subseteq E(G)$ setzen wir dann $c(F) := \sum_{e \in F} c(e)$ (und $c(\emptyset) = 0$). Dadurch wird c zu einer modularen Funktion $c : 2^{E(G)} \rightarrow \mathbb{R}$ erweitert. Mit $\text{dist}_{(G,c)}(v, w)$ bezeichnen wir den minimalen Wert von $c(E(P))$ für alle v - w -Wege P in G .

2.2 Bäume, Kreise und Schnitte

Ein ungerichteter Graph G heißt **zusammenhängend** wenn es für alle $v, w \in V(G)$ einen $v-w$ -Weg gibt; sonst heißt G **unzusammenhängend**. Ein Digraph heißt zusammenhängend, falls der zugrunde liegende ungerichtete Graph zusammenhängend ist. Die maximalen zusammenhängenden Teilgraphen eines Graphen sind seine **Zusammenhangskomponenten**. Gelegentlich identifizieren wir die Zusammenhangskomponenten mit den Knotenmengen, von denen sie induziert werden. Dementsprechend heißt eine Knotenmenge X zusammenhängend, wenn der von X induzierte Teilgraph zusammenhängend ist. Ein Knoten v mit der Eigenschaft, dass $G - v$ mehr Zusammenhangskomponenten als G hat, heißt **Artikulationsknoten**. Eine Kante e mit der Eigenschaft, dass $G - e$ mehr Zusammenhangskomponenten als G hat, heißt **Brücke**.

Ein ungerichteter Graph ohne Kreise (als Teilgraph) heißt **Wald**. Ein zusammenhängender Wald heißt **Baum**. Ein Knoten vom Grad höchstens gleich 1 in einem Baum heißt **Blatt**. Ein Baum mit höchstens einem Knoten, der kein Blatt ist, heißt **Stern**.

Wir werden nun einige äquivalente Formulierungen für Bäume und ihre gerichteten Varianten, die Arboreszenzen, angeben. Dazu benötigen wir das folgende Zusammenhangskriterium:

- Proposition 2.3.** (a) *Ein ungerichteter Graph G ist genau dann zusammenhängend, wenn $\delta(X) \neq \emptyset$ für alle $\emptyset \neq X \subset V(G)$.*
 (b) *Sei G ein Digraph und $r \in V(G)$. Dann gibt es für jedes $v \in V(G)$ einen $r-v$ -Weg genau dann, wenn $\delta^+(X) \neq \emptyset$ für alle $X \subset V(G)$ mit $r \in X$.*

Beweis: (a): Gibt es eine Menge $X \subset V(G)$ mit $r \in X$, $v \in V(G) \setminus X$ und $\delta(X) = \emptyset$, so kann es keinen $r-v$ -Weg geben, also ist G unzusammenhängend. Ist andererseits G unzusammenhängend, so gibt es ein r und ein v , für die es keinen $r-v$ -Weg gibt. Sei R die Menge der von r aus erreichbaren Knoten. Dann ist $r \in R$, $v \notin R$ und $\delta(R) = \emptyset$.

(b) wird analog bewiesen. □

Satz 2.4. *Sei G ein ungerichteter Graph mit n Knoten. Dann sind die folgenden sieben Aussagen äquivalent:*

- (a) *G ist ein Baum (d. h. zusammenhängend und kreisfrei).*
- (b) *G hat $n - 1$ Kanten und ist kreisfrei.*
- (c) *G hat $n - 1$ Kanten und ist zusammenhängend.*
- (d) *G ist zusammenhängend und jede Kante ist eine Brücke.*
- (e) *G erfüllt $\delta(X) \neq \emptyset$ für alle $\emptyset \neq X \subset V(G)$, die Entfernung einer beliebigen Kante von G zerstört jedoch diese Eigenschaft.*
- (f) *G ist ein Wald mit der Eigenschaft, dass das Hinzufügen irgendeiner weiteren Kante einen Kreis erzeugt.*
- (g) *G enthält einen eindeutig bestimmten Weg zwischen je zwei seiner Knoten.*

Beweis: (a)⇒(g): Dies folgt aus der Tatsache, dass die Vereinigung zweier verschiedener Wege mit denselben Endknoten immer einen Kreis erzeugt.

(g) \Rightarrow (e) \Rightarrow (d) folgt mit Proposition 2.3(a).

(d) \Rightarrow (f) ist trivial.

(f) \Rightarrow (b) \Rightarrow (c): Dies folgt aus der Tatsache, dass für jeden Wald mit n Knoten, m Kanten und p Zusammenhangskomponenten $n = m + p$ gilt. (Der Beweis erfolgt leicht mittels Induktion über m .)

(c) \Rightarrow (a): Sei G zusammenhängend mit $n - 1$ Kanten. Gibt es Kreise in G , so zerstören wir jeden von ihnen, indem wir eine seiner Kanten entfernen. Haben wir auf diese Weise k Kanten entfernt, so ist der resultierende Graph G' kreisfrei, aber immer noch zusammenhängend; ferner hat er $m = n - 1 - k$ Kanten. Somit ist $n = m + p = n - 1 - k + 1$, woraus $k = 0$ folgt. \square

Insbesondere folgt aus (d) \Rightarrow (a), dass ein Graph genau dann zusammenhängend ist, wenn er einen **aufspannenden Baum** (einen aufspannenden Teilgraphen, der ein Baum ist) enthält.

Ein Digraph ist ein **Branching**, falls der zugrunde liegende ungerichtete Graph ein Wald ist und jeder Knoten v höchstens eine ankommende Kante hat. Ein zusammenhängendes Branching heißt **Arboreszenz**. Eine Arboreszenz mit n Knoten hat nach Satz 2.4 $n - 1$ Kanten, somit hat sie genau einen Knoten r mit $\delta^-(r) = \emptyset$. Dieser Knoten heißt die **Wurzel** der Arboreszenz. Für einen gegebenen Knoten v eines Branchings heißen diejenigen Knoten w , für die (v, w) eine Kante ist, die **Kinder** von v . Ist w ein Kind von v , so heißt v der **Vorgänger** von w . Knoten ohne Kinder heißen **Blätter**.

Satz 2.5. *Sei G ein Digraph mit n Knoten. Dann sind die folgenden sieben Aussagen äquivalent:*

- (a) *G ist eine Arboreszenz mit Wurzel r (d. h. ein zusammenhängendes Branching mit $\delta^-(r) = \emptyset$).*
- (b) *G ist ein Branching mit $n - 1$ Kanten und $\delta^-(r) = \emptyset$.*
- (c) *G hat $n - 1$ Kanten und jeder Knoten ist von r aus erreichbar.*
- (d) *Jeder Knoten ist von r aus erreichbar, aber das Entfernen einer beliebigen Kante zerstört diese Eigenschaft.*
- (e) *G erfüllt $\delta^+(X) \neq \emptyset$ für alle $X \subset V(G)$ mit $r \in X$, die Entfernung einer beliebigen Kante von G zerstört jedoch diese Eigenschaft.*
- (f) *$\delta^-(r) = \emptyset$, und für jedes $v \in V(G) \setminus \{r\}$ gibt es einen eindeutig bestimmten Spaziergang von r nach v .*
- (g) *$\delta^-(r) = \emptyset$ und $|\delta^-(v)| = 1$ für alle $v \in V(G) \setminus \{r\}$, und G ist kreisfrei.*

Beweis: (a) \Rightarrow (b) und (c) \Rightarrow (d): Diese Implikationen folgen mit Satz 2.4.

(b) \Rightarrow (c): Wir haben $|\delta^-(v)| = 1$ für alle $v \in V(G) \setminus \{r\}$. Also haben wir für jedes v einen r - v -Weg (beginne in v und folge immer der ankommenden Kante bis zu r).

(d) \Leftrightarrow (e): Dies folgt mit Proposition 2.3(b).

(d) \Rightarrow (f): Jede Kante in $\delta^-(r)$ kann entfernt werden ohne die Erreichbarkeit von r aus zu zerstören. Angenommen, für ein $v \in V(G)$ gäbe es zwei r - v -Spaziergänge P und Q . Sei $e = (v, w)$ die letzte Kante von P , die nicht zu Q gehört oder nicht die

erste in w endende Kante von Q ist (wenn keine solche Kante existiert, vertausche man P und Q). Dann ist nach dem Entfernen von e jeder Knoten weiterhin von r aus erreichbar.

(f) \Rightarrow (g): Ist jeder Knoten von r aus erreichbar und gibt es einen Knoten $v \in V(G) \setminus \{r\}$ mit $|\delta^-(v)| > 1$, dann haben wir zwei Spaziergänge von r nach v . Enthält G einen Kreis C , so sei $v \in V(C)$ und P der eindeutig bestimmte r - v -Weg. Sei x der erste zu C gehörende Knoten von P . Dann gibt es zwei Spaziergänge von r nach x , nämlich $P_{[r,x]}$ und $P_{[r,x]}$ plus C .

(g) \Rightarrow (a): Gilt $|\delta^-(v)| \leq 1$ für alle Knoten v , so ist jeder ungerichtete Kreis ein (gerichteter) Kreis. \square

Ein **Schnitt** in einem ungerichteten Graphen G ist eine Kantensubmenge vom Typ $\delta(X)$ für $\emptyset \neq X \subset V(G)$. In einem Digraphen G ist $\delta^+(X)$ ein **gerichteter Schnitt**, falls $\emptyset \neq X \subset V(G)$ und $\delta^-(X) = \emptyset$, d. h. keine Kante endet in X .

Wir sagen, dass eine Kantensubmenge $F \subseteq E(G)$ zwei Knoten s und t **trennt**, falls t von s aus in G , aber nicht in $(V(G), E(G) \setminus F)$ erreichbar ist. Ein **s - t -Schnitt** in einem ungerichteten Graphen ist ein Schnitt $\delta(X)$, wobei $X \subset V(G)$ mit $s \in X$ und $t \notin X$. Ein **s - t -Schnitt** in einem Digraphen ist eine Kantensubmenge $\delta^+(X)$ mit $s \in X$ und $t \notin X$. Ein **r -Schnitt** in einem Digraphen ist eine Kantensubmenge $\delta^+(X)$, wobei $X \subset V(G)$ mit $r \in X$.

Ein **ungerichteter Schnitt** in einem Digraphen ist eine Kantensubmenge die einem Schnitt im zugrunde liegenden ungerichteten Graphen entspricht, d. h. $\delta(X)$ für ein $\emptyset \neq X \subset V(G)$.

Lemma 2.6. (Minty [1960]) *Sei G ein Digraph und $e \in E(G)$. Angenommen, e ist schwarz gefärbt, während alle anderen Kanten rot, schwarz oder grün gefärbt sind. Dann gilt genau eine der folgenden zwei Aussagen:*

- (a) *Es gibt einen ungerichteten Kreis, der e enthält und dessen weitere Kanten nur rot oder schwarz sind, wobei die schwarzen Kanten alle gleich orientiert sind.*
- (b) *Es gibt einen ungerichteten Schnitt, der e enthält und dessen weitere Kanten nur grün oder schwarz sind, wobei die schwarzen Kanten alle gleich orientiert sind.*

Beweis: Sei $e = (x, y)$. Wir markieren nun die Knoten von G wie folgt. Zuerst markieren wir y . Ist v bereits markiert und w noch nicht, so markieren wir w , falls es eine schwarze Kante (v, w) , eine rote Kante (v, w) oder eine rote Kante (w, v) gibt. Wird w markiert, so schreiben wir $\text{pred}(w) := v$.

Nach Beendigung des Markierungsverfahrens gibt es zwei Möglichkeiten:

Fall 1: x ist markiert worden. Dann bilden die Knoten $x, \text{pred}(x), \text{pred}(\text{pred}(x)), \dots, y$ einen ungerichteten Kreis mit der Eigenschaft (a).

Fall 2: x ist nicht markiert worden. Dann sei R die Menge aller markierten Knoten. Offensichtlich hat der ungerichtete Schnitt $\delta^+(R) \cup \delta^-(R)$ die Eigenschaft (b).

Angenommen, es gäbe sowohl einen ungerichteten Kreis C wie in (a) als auch einen ungerichteten Schnitt $\delta^+(X) \cup \delta^-(X)$ wie in (b). Alle Kanten in deren (nichtleerem) Durchschnitt sind schwarz, alle haben dieselbe Orientierung bezüglich C , und alle beginnen in X oder alle enden in X . Dies ist jedoch ein Widerspruch. \square

Ein Digraph heißt **stark zusammenhängend**, falls es für alle $s, t \in V(G)$ einen Weg von s nach t und einen Weg von t nach s gibt. Die **starken Zusammenhangskomponenten** eines Digraphen sind die maximalen stark zusammenhängenden Teilgraphen.

Korollar 2.7. *In einem Digraphen G ist jede Kante entweder in einem (gerichteten) Kreis oder in einem gerichteten Schnitt enthalten. Ferner sind die folgenden drei Aussagen äquivalent:*

- (a) G ist stark zusammenhängend.
- (b) G enthält keinen gerichteten Schnitt.
- (c) G ist zusammenhängend und jede Kante von G liegt in einem Kreis.

Beweis: Die erste Aussage folgt sofort aus Mintys Lemma 2.6, indem man alle Kanten schwarz färbt. Hiermit ist auch (b) \Rightarrow (c) bewiesen.

(a) \Rightarrow (b): Dies folgt aus Proposition 2.3(b).

(c) \Rightarrow (a): Sei $r \in V(G)$ ein beliebiger Knoten. Wir beweisen nun, dass es für jedes $v \in V(G)$ einen r - v -Weg gibt. Angenommen, das Gegenteil wäre der Fall. Dann gibt es nach Proposition 2.3(b) ein $X \subset V(G)$ mit $r \in X$ und $\delta^+(X) = \emptyset$. Da G zusammenhängend ist, folgt $\delta^+(X) \cup \delta^-(X) \neq \emptyset$ (nach Proposition 2.3(a)), also sei $e \in \delta^-(X)$. Dann kann e aber nicht in einem Kreis liegen, da keine Kante aus X hinausführt. \square

Aus Korollar 2.7 und Satz 2.5 folgt, dass ein Digraph genau dann stark zusammenhängend ist, wenn er für jeden Knoten v eine aufspannende Arboreszenz mit Wurzel v enthält.

Ein Digraph heißt **azyklisch**, falls er keinen (gerichteten) Kreis enthält. Nach Korollar 2.7 folgt somit: Ein Digraph ist genau dann azyklisch, wenn jede Kante in einem gerichteten Schnitt liegt. Ferner ist ein Digraph genau dann azyklisch, wenn seine starken Zusammenhangskomponenten die einelementigen Knotenmengen sind. Die Knoten eines azyklischen Digraphen können auf schöne Art geordnet werden:

Definition 2.8. *Sei G ein Digraph. Eine **topologische Ordnung** von G ist eine Ordnung der Knoten $V(G) = \{v_1, \dots, v_n\}$, so dass für jede Kante $(v_i, v_j) \in E(G)$ $i < j$ gilt.*

Proposition 2.9. *Ein Digraph hat genau dann eine topologische Ordnung, wenn er azyklisch ist.*

Beweis: Hat ein Digraph einen Kreis, so kann er offensichtlich keine topologische Ordnung haben. Die Umkehrung beweisen wir mittels Induktion über die Anzahl der Kanten. Gibt es gar keine Kanten, so ist jede Ordnung topologisch. Sei andererfalls $e \in E(G)$. Dann folgt mit Korollar 2.7, dass e in einem gerichteten Schnitt $\delta^+(X)$ liegt. Eine topologische Ordnung von $G[X]$, gefolgt von einer topologischen Ordnung von $G - X$ (beide existieren nach der Induktionsvoraussetzung) liefert dann eine topologische Ordnung von G . \square

Kreise und Schnitte spielen auch eine wichtige Rolle in der algebraischen Graphentheorie. Jedem Graphen G ordnen wir einen Vektorraum $\mathbb{R}^{E(G)}$ zu, dessen Elemente Vektoren $(x_e)_{e \in E(G)}$ mit $|E(G)|$ reellen Komponenten sind. In Anlehnung an Berge [1985] werden wir nun kurz auf zwei besonders wichtige Untervektorräume eingehen.

Sei G ein Digraph. Jedem ungerichteten Kreis C in G ordnen wir einen Vektor $\zeta(C) \in \{-1, 0, 1\}^{E(G)}$ zu, indem wir $\zeta(C)_e = 0$ für alle $e \notin E(C)$ und $\zeta(C)_e \in \{-1, 1\}$ für alle $e \in E(C)$ setzen, so dass die Umorientierung aller Kanten e mit $\zeta(C)_e = -1$ einen gerichteten Kreis ergibt. Analog ordnen wir jedem ungerichteten Schnitt $D = \delta(X)$ in G einen Vektor $\zeta(D) \in \{-1, 0, 1\}^{E(G)}$ zu, indem wir $\zeta(D)_e = 0$ für alle $e \notin D$, $\zeta(D)_e = -1$ für alle $e \in \delta^-(X)$ und $\zeta(D)_e = 1$ für alle $e \in \delta^+(X)$ setzen. Beachte, dass diese Vektoren nur bis auf Multiplikation mit -1 eindeutig definiert sind. Die beiden Teilmengen des Vektorraumes $\mathbb{R}^{E(G)}$ jedoch, die einerseits von den Vektoren, die den ungerichteten Kreisen in G zugeordnet wurden, und andererseits von den Vektoren, die den ungerichteten Schnitten in G zugeordnet wurden, erzeugt werden, sind aber eindeutig definiert. Sie heißen **Kreisraum** bzw. **Kozyklenraum** von G .

Proposition 2.10. *Der Kreisraum und der Kozyklenraum sind orthogonal zu einander.*

Beweis: Sei C ein ungerichteter Kreis und $D = \delta(X)$ ein ungerichteter Schnitt. Wir behaupten nun, dass das Skalarprodukt von $\zeta(C)$ und $\zeta(D)$ Null ist. Da jede Umorientierung einer Kante das Skalarprodukt nicht verändert, können wir annehmen, dass D ein gerichteter Schnitt ist. Die Aussage folgt nun aus der Beobachtung, dass jeder Kreis eine Menge X genauso oft verlässt wie er in ihr ankommt. \square

Wir werden nun zeigen, dass die Summe der Dimensionen von Kreisraum und Kozyklenraum gleich der Dimension $|E(G)|$ des ganzen Raumes ist. Eine Menge ungerichteter Kreise (bzw. ungerichteter Schnitte) heißt **Kreisbasis** (bzw. **Kozyklenbasis**), falls die zugeordneten Vektoren eine Basis des Kreisraumes (bzw. Kozyklenraumes) bilden. Sei G ein Graph (gerichtet oder ungerichtet) und T ein maximaler Teilgraph ohne ungerichtete Kreise. Dann nennen wir den für jedes $e \in E(G) \setminus E(T)$ eindeutig bestimmten ungerichteten Kreis in $T + e$ den **Fundamentalkreis** von e bezüglich T . Ferner gibt es für jedes $e \in E(T)$ eine Menge $X \subseteq V(G)$ mit $\delta_G(X) \cap E(T) = \{e\}$ (betrachte eine Komponente von $T - e$). Wir nennen $\delta_G(X)$ den **Fundamentalschnitt** von e bezüglich T .

Satz 2.11. *Sei G ein Digraph und T ein maximaler Teilgraph ohne ungerichtete Kreise. Die $|E(G) \setminus E(T)|$ Fundamentalkreise bezüglich T bilden eine Kreisbasis von G und die $|E(T)|$ Fundamentalschnitte bezüglich T eine Kozyklenbasis von G .*

Beweis: Die den Fundamentalkreisen zugeordneten Vektoren sind linear unabhängig, da jeder Fundamentalkreis eine zu keinem anderen Fundamentalkreis gehörende Kante enthält. Für die Fundamentalschnitte haben wir ein analoges Resultat. Da die zwei Untervektorräume nach Proposition 2.10 orthogonal zu einander sind, kann die Summe ihrer Dimensionen nicht größer als $|E(G)| = |E(G) \setminus E(T)| + |E(T)|$ sein. \square

Die Fundamentalschnitte haben eine schöne Eigenschaft, die wir des öfteren gebrauchen werden und die wir jetzt erläutern. Sei T ein Digraph, dessen zugrunde liegender ungerichteter Graph ein Baum ist. Wir betrachten nun die Familie $\mathcal{F} := \{C_e : e \in E(T)\}$, wobei wir für jedes $e = (x, y) \in E(T)$ die y enthaltende Zusammenhangskomponente von $T - e$ mit C_e bezeichnen (also ist $\delta(C_e)$ der Fundamentalschnitt von e bezüglich T). Ist T eine Arboreszenz, so sind je zwei Elemente von \mathcal{F} entweder disjunkt oder das eine ist Teilmenge des anderen. Im Allgemeinen ist \mathcal{F} wenigstens kreuzungsfrei:

Definition 2.12. Ein **Mengensystem** ist ein Paar (U, \mathcal{F}) , wobei U eine nichtleere endliche Menge und \mathcal{F} eine Familie von Teilmengen von U ist. Es ist (U, \mathcal{F}) **kreuzungsfrei**, falls für je zwei Mengen $X, Y \in \mathcal{F}$ mindestens eine der vier Mengen $X \setminus Y, Y \setminus X, X \cap Y, U \setminus (X \cup Y)$ leer ist. Es ist (U, \mathcal{F}) **laminar**, falls für je zwei Mengen $X, Y \in \mathcal{F}$ mindestens eine der drei Mengen $X \setminus Y, Y \setminus X, X \cap Y$ leer ist.

In der Literatur werden Mengensysteme auch **Hypergraphen** genannt. Abbildung 2.1(a) zeigt die laminare Familie $\{\{a\}, \{b, c\}, \{a, b, c\}, \{a, b, c, d\}, \{f\}, \{f, g\}\}$. Eine andere gebräuchliche Bezeichnung für laminar ist **geschachtelt**.

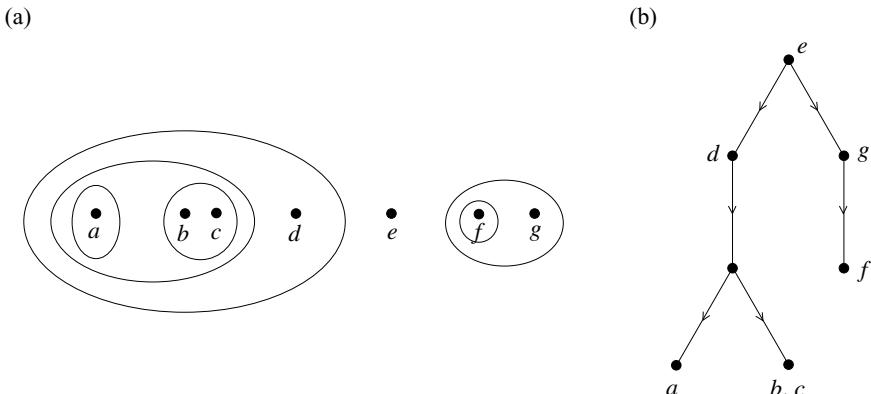


Abbildung 2.1.

Ob ein Mengensystem (U, \mathcal{F}) laminar ist, hängt nicht von U ab, also sagt man oft der Einfachheit halber, dass \mathcal{F} eine laminare Familie ist. Ob ein Mengensystem kreuzungsfrei ist, kann aber durchaus von der Grundmenge U abhängen. Enthält U ein Element, das in keiner der Mengen aus \mathcal{F} enthalten ist, so ist \mathcal{F} genau dann kreuzungsfrei, wenn \mathcal{F} laminar ist. Sei $r \in U$ beliebig. Aus der Definition folgt sofort, dass ein Mengensystem (U, \mathcal{F}) genau dann kreuzungsfrei ist, wenn

$$\mathcal{F}' := \{X \in \mathcal{F} : r \notin X\} \cup \{U \setminus X : X \in \mathcal{F}, r \in X\}$$

laminar ist. Deswegen werden kreuzungsfreie Familien manchmal ähnlich wie laminare Familien abgebildet: Z. B. zeigt Abb. 2.2(a) die kreuzungsfreie Familie $\{\{b, c, d, e, f\}, \{c\}, \{a, b, c\}, \{e\}, \{a, b, c, d, f\}, \{e, f\}\}$, wobei ein Rechteck der Menge entspricht, die alle außerhalb liegenden Elemente enthält.

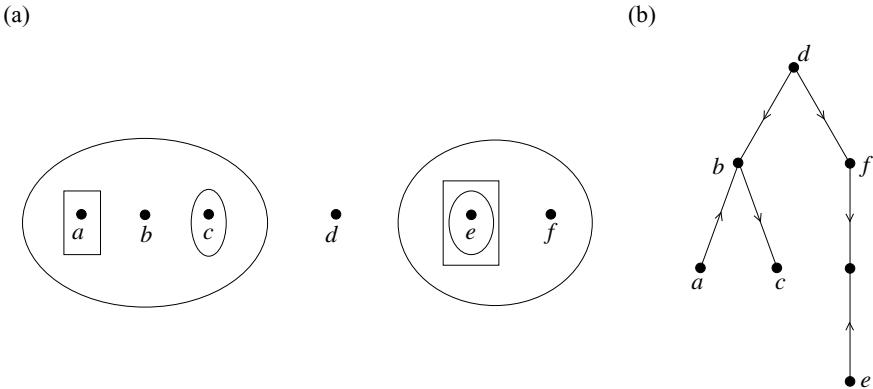


Abbildung 2.2.

Während gerichtete Bäume ganz natürlich zu kreuzungsfreien Familien führen, gilt die Umkehrung in einem gewissen Sinne auch: Jede kreuzungsfreie Familie kann auf folgende Weise mittels eines Baumes repräsentiert werden:

Definition 2.13. Sei T ein Digraph, dessen zugrunde liegender ungerichteter Graph ein Baum ist. Sei ferner U eine endliche Menge und $\varphi : U \rightarrow V(T)$. Wir setzen $\mathcal{F} := \{S_e : e \in E(T)\}$, wobei wir für jedes $e = (x, y)$ $S_e := \{s \in U : \varphi(s)$ ist in derselben Zusammenhangskomponente von $T - e$ wie $y\}$ definieren. Dann nennt man (T, φ) eine **Baumdarstellung** von (U, \mathcal{F}) .

Siehe Beispiele in Abb. 2.1(b) und 2.2(b).

Proposition 2.14. Sei (U, \mathcal{F}) ein Mengensystem mit einer Baumdarstellung (T, φ) . Dann ist (U, \mathcal{F}) kreuzungsfrei. Ist T eine Arboreszenz, so ist (U, \mathcal{F}) laminar. Ferner hat jede kreuzungsfreie Familie eine Baumdarstellung, und für laminare Familien kann man für T eine Arboreszenz wählen.

Beweis: Ist (T, φ) eine Baumdarstellung von (U, \mathcal{F}) und $e = (v, w), f = (x, y) \in E(T)$, dann haben wir einen ungerichteten v - x -Weg P in T (ohne Berücksichtigung der Orientierungen). Es gibt vier Fälle: Sind $w, y \notin V(P)$, so folgt $S_e \cap S_f = \emptyset$ (da T kreisfrei ist). Ist $w \notin V(P)$ und $y \in V(P)$, so folgt $S_e \subseteq S_f$. Ist $y \notin V(P)$ und $w \in V(P)$, so folgt $S_f \subseteq S_e$. Sind $w, y \in V(P)$, so folgt $S_e \cup S_f = U$. Somit ist (U, \mathcal{F}) kreuzungsfrei. Ist T eine Arboreszenz, so ist der letzte Fall unmöglich, da sonst in mindestens einem Knoten von P zwei Kanten ankämen. Also ist \mathcal{F} laminar.

Nun beweisen wir die Umkehrung. Zunächst sei \mathcal{F} eine laminare Familie. Wir setzen $V(T) := \mathcal{F} \cup \{r\}$ und $E(T) :=$

$$\begin{aligned} & \{(X, Y) \in \mathcal{F} \times \mathcal{F} : X \supset Y \neq \emptyset \text{ und es gibt kein } Z \in \mathcal{F} \text{ mit } X \supset Z \supset Y\} \\ & \cup \{(r, X) : X = \emptyset \in \mathcal{F} \text{ oder } X \text{ ist ein inklusionsmaximales Element von } \mathcal{F}\}. \end{aligned}$$

Ferner setzen wir $\varphi(x) := X$, wobei X die minimale x enthaltende Menge in \mathcal{F} ist, und $\varphi(x) := r$, falls es keine x enthaltende Menge in \mathcal{F} gibt. Offensichtlich ist T eine Arboreszenz mit Wurzel r und (T, φ) eine Baumdarstellung von \mathcal{F} .

Nun sei \mathcal{F}' eine kreuzungsfreie Familie von Teilmengen von U . Ferner sei $r \in U$. Wie wir oben gesehen haben, ist

$$\mathcal{F}' := \{X \in \mathcal{F} : r \notin X\} \cup \{U \setminus X : X \in \mathcal{F}, r \in X\}$$

laminar, also sei (T, φ) eine Baumdarstellung von (U, \mathcal{F}') . Für jede Kante $e \in E(T)$ gibt es drei Fälle: Ist $S_e \in \mathcal{F}$ und $U \setminus S_e \in \mathcal{F}$, so ersetzen wir die Kante $e = (x, y)$ durch zwei Kanten (x, z) und (y, z) , wobei z ein neuer Knoten ist. Ist $S_e \notin \mathcal{F}$ und $U \setminus S_e \in \mathcal{F}$, so ersetzen wir die Kante $e = (x, y)$ durch (y, x) . Ist $S_e \in \mathcal{F}$ und $U \setminus S_e \notin \mathcal{F}$, so ändern wir nichts. Sei T' der resultierende Graph. Dann ist (T', φ) eine Baumdarstellung von (U, \mathcal{F}) . \square

Das obige Resultat ist in Edmonds und Giles [1977] erwähnt worden, war aber wahrscheinlich bereits früher bekannt.

Korollar 2.15. *Eine laminare Familie paarweise verschiedener Teilmengen von U hat höchstens $2|U|$ Elemente. Eine kreuzungsfreie Familie paarweise verschiedener Teilmengen von U hat höchstens $4|U| - 2$ Elemente.*

Beweis: Zunächst betrachten wir eine laminare Familie \mathcal{F} paarweise verschiedener nichtleerer echter Teilmengen von U . Wir beweisen, dass $|\mathcal{F}| \leq 2|U| - 2$ gilt. Sei (T, φ) eine Baumdarstellung, wobei T ein Arboreszenz ist, deren Anzahl von Knoten so klein wie möglich ist. Dann haben wir für jedes $w \in V(T)$: Entweder gilt $|\delta^+(w)| \geq 2$, oder es gibt ein $x \in U$ mit $\varphi(x) = w$, oder beides gilt. (Für die Wurzel folgt dies aus $U \notin \mathcal{F}$, für die Blätter aus $\emptyset \notin \mathcal{F}$ und für alle anderen Knoten aus der Minimalität von T .)

Es kann höchstens $|U|$ Knoten w mit $\varphi(x) = w$ für ein $x \in U$ geben und höchstens $\left\lfloor \frac{|E(T)|}{2} \right\rfloor$ Knoten w mit $|\delta^+(w)| \geq 2$. Somit haben wir $|E(T)| + 1 = |V(T)| \leq |U| + \frac{|E(T)|}{2}$ und es folgt $|\mathcal{F}| = |E(T)| \leq 2|U| - 2$.

Nun sei (U, \mathcal{F}) eine kreuzungsfreie Familie mit $\emptyset, U \notin \mathcal{F}$ und $r \in U$. Da $\mathcal{F}' := \{X \in \mathcal{F} : r \notin X\} \cup \{U \setminus X : X \in \mathcal{F}, r \in X\}$ laminar ist, folgt $|\mathcal{F}'| \leq 2|U| - 2$. Also haben wir $|\mathcal{F}| \leq 2|\mathcal{F}'| \leq 4|U| - 4$. Der Beweis ist mit der Berücksichtigung von \emptyset und U als mögliche Elemente von \mathcal{F} abgeschlossen. \square

2.3 Zusammenhang

Der Begriff des Zusammenhangs spielt eine sehr wichtige Rolle in der Graphentheorie. Für viele Probleme genügt es, zusammenhängende Graphen zu betrachten, da man das Problem für jede einzelne Zusammenhangskomponente separat lösen kann. Somit ist die Bestimmung der Zusammenhangskomponenten eines Graphen eine fundamentale Aufgabe. Der folgende einfache Algorithmus findet einen Weg

von einem festen Knoten s zu allen von s aus erreichbaren Knoten. Er funktioniert sowohl für gerichtete als auch für ungerichtete Graphen. Im ungerichteten Fall baut er einen maximalen s enthaltenden Baum auf und im gerichteten Fall eine maximale Arboreszenz mit Wurzel s .

GRAPH-SCANNING-ALGORITHMUS

Input: Ein Graph G (gerichtet oder ungerichtet) und ein Knoten s .

Output: Die Menge R der von s aus erreichbaren Knoten und eine Menge $T \subseteq E(G)$, für die (R, T) eine Arboreszenz mit Wurzel s oder ein Baum ist.

- ① Setze $R := \{s\}$, $Q := \{s\}$ und $T := \emptyset$.
- ② If $Q = \emptyset$ then stop,
else wähle ein $v \in Q$.
- ③ Wähle ein $w \in V(G) \setminus R$ mit $e = (v, w) \in E(G)$ oder $e = \{v, w\} \in E(G)$.
If es gibt keinen solchen Knoten w then setze $Q := Q \setminus \{v\}$ und go to ②.
- ④ Setze $R := R \cup \{w\}$, $Q := Q \cup \{w\}$ und $T := T \cup \{e\}$. Go to ②.

Proposition 2.16. Der GRAPH-SCANNING-ALGORITHMUS arbeitet korrekt.

Beweis: Zu jeder Zeit ist (R, T) ein Baum oder eine Arboreszenz mit Wurzel s . Angenommen, bei Terminierung liegt ein von s aus erreichbarer Knoten $w \in V(G) \setminus R$ vor. Sei P ein $s-w$ -Weg und $\{x, y\}$ oder (x, y) eine Kante von P mit $x \in R$ und $y \notin R$. Da x der Knotenmenge R hinzugefügt wurde, ist x auch irgendwann während des Ablaufes des Algorithmus der Knotenmenge Q hinzugefügt worden. Der Algorithmus terminiert nicht bevor er x aus Q entfernt hat. Dies erfolgt jedoch nur dann in ③, wenn es keine Kante $\{x, y\}$ oder (x, y) mit $y \notin R$ gibt. \square

Da dies der erste Graphenalgorithmus in diesem Buch ist, werden wir einige Implementierungsdetails besprechen. Die erste Frage betrifft die Darstellung des gegebenen Graphen. Diese kann auf verschiedene Arten erfolgen, z.B. kann man sich eine Matrix vorstellen, deren Zeilen bzw. Spalten den Knoten bzw. Kanten des Graphen entsprechen. Die **Inzidenzmatrix** eines ungerichteten Graphen G ist die Matrix $A = (a_{v,e})_{v \in V(G), e \in E(G)}$ mit

$$a_{v,e} = \begin{cases} 1 & \text{für } v \in e \\ 0 & \text{für } v \notin e. \end{cases}$$

Die **Inzidenzmatrix** eines Digraphen G ist die Matrix $A = (a_{v,e})_{v \in V(G), e \in E(G)}$ mit

$$a_{v,(x,y)} = \begin{cases} -1 & \text{für } v = x \\ 1 & \text{für } v = y \\ 0 & \text{für } v \notin \{x, y\}. \end{cases}$$

Natürlich ist diese Darstellung nicht sehr effizient, da jede Spalte nur zwei nicht-verschwindende Komponenten enthält. Der zur Speicherung einer Inzidenzmatrix benötigte Platz ist offensichtlich $O(nm)$, wobei $n := |V(G)|$ und $m := |E(G)|$.

Eine bessere Darstellung wird durch eine Matrix gegeben, deren Zeilen und auch Spalten mit der Knotenmenge indiziert sind. Die **Adjazenzmatrix** eines einfachen Graphen G ist die 0-1-Matrix $A = (a_{v,w})_{v,w \in V(G)}$ mit $a_{v,w} = 1$ genau dann, wenn $\{v, w\} \in E(G)$ oder $(v, w) \in E(G)$. Für Graphen mit parallelen Kanten können wir $a_{v,w}$ als die Anzahl der Kanten von v nach w definieren. Eine Adjazenzmatrix benötigt $O(n^2)$ Speicherplatz für einfache Graphen.

Die Adjazenzmatrix ist für **dichte** Graphen gut geeignet, d. h. solche mit $\Theta(n^2)$ oder mehr Kanten. Für **dünne** Graphen, z. B. mit nur $O(n)$ Kanten, kann man aber viel besser verfahren. Neben der Speicherung der Anzahl der Knoten, brauchen wir nur noch eine Liste der Kanten zu speichern, indem wir für jede Kante ihre Endknoten notieren. Bezeichnen wir jeden Knoten mit einer Zahl zwischen 1 und n , so benötigen wir $O(\log n)$ Speicherplatz pro Kante. Also benötigen wir $O(m \log n)$ Platz insgesamt.

Die Speicherung der Kanten in einer beliebigen Reihenfolge ist jedoch nicht besonders praktisch. Fast alle Graphenalgorithmen müssen die mit einem gegebenen Knoten inzidenten Kanten auffinden. Also sollte man auch für jeden Knoten eine Liste der mit ihm inzidenten Kanten haben. Für gerichtete Graphen sind zwei Listen nötig, eine für die dort endenden und eine für die dort beginnenden Kanten. Diese Datenstruktur heißt **Adjazenzliste** und ist für Graphen die übliche. Für den direkten Zugriff auf diese Liste(n) für jeden Knoten benutzt man Zeiger auf den Listenanfang. Diese Zeiger benötigen zusätzlich $O(n \log m)$ Speicherplatz. Somit benötigt eine Adjazenzliste insgesamt $O(n \log m + m \log n)$ Speicherplatz.

Wann immer in diesem Buch ein Graph Teil des Inputs eines Algorithmus ist, gehen wir davon aus, dass er mittels einer Adjazenzliste gegeben ist.

Wie für die elementaren Operationen mit Zahlen (siehe Abschnitt 1.2), nehmen wir an, dass die elementaren Operationen für Knoten und Kanten nur konstante Zeit in Anspruch nehmen. Dazu zählen wir auch das Scannen einer Kante, die Identifizierung ihrer Endknoten und das Aufrufen des Anfangs der Adjazenzliste eines Knotens. Die Laufzeit wird mittels der Parameter n und m gegeben, z. B. heißt ein Algorithmus mit $O(m + n)$ -Laufzeit linear.

Die Buchstaben n bzw. m werden immer die Anzahl der Knoten bzw. Kanten eines Graphen bezeichnen. Für viele Graphenalgorithmen können wir o. B. d. A. annehmen, dass der vorliegende Graph einfach und zusammenhängend ist, somit haben wir $n - 1 \leq m < n^2$. Oft brauchen wir von mehreren parallelen Kanten nur eine zu betrachten, und verschiedene Zusammenhangskomponenten können meist separat bearbeitet werden. Die Vorbearbeitung kann vorab in linearer Zeit erledigt werden, siehe unten und auch Aufgabe 17.

Wir sind nun in der Lage, die Laufzeit des GRAPH-SCANNING-ALGORITHMUS zu betrachten:

Proposition 2.17. *Der GRAPH-SCANNING-ALGORITHMUS kann so implementiert werden, dass er mit $O(m)$ -Laufzeit läuft. Die Zusammenhangskomponenten eines ungerichteten Graphen können in linearer Zeit bestimmt werden.*

Beweis: Wir nehmen an, dass G mittels einer Adjazenzliste gegeben ist. Implementiere Q mittels einer einfachen Liste, so dass ② konstante Zeit benötigt. Für jeden zu Q hinzugefügten Knoten x führen wir einen Zeiger $current(x)$ ein, der die aktuelle Kante in der Liste aller Kanten in $\delta(x)$ oder $\delta^+(x)$ aufzeigt (diese Liste ist Teil des Inputs). Am Anfang wird $current(x)$ auf das erste Element der Liste gesetzt. In ③ bewegt sich der Zeiger vorwärts. Bei Erreichen des Endes der Liste wird x aus Q entfernt und niemals wieder hinzugefügt. Also ist die Gesamlaufzeit proportional zur Summe der Anzahl der von s aus erreichbaren Knoten und der Anzahl der Kanten, d. h. $O(m)$.

Zur Bestimmung der Zusammenhangskomponenten eines Graphen wenden wir den Algorithmus einmal an und prüfen dann, ob $R = V(G)$. Falls ja, ist der Graph zusammenhängend. Sonst ist R eine Zusammenhangskomponente, und wir wenden den Algorithmus auf (G, s') an, wobei $s' \in V(G) \setminus R$ ein beliebiger Knoten ist (und iterieren, bis alle Knoten gescannt, d. h. zu R hinzugefügt worden sind). Wir betonen, dass keine Kante mehr als zweimal gescannt wird, d. h. die Gesamlaufzeit bleibt linear. \square

Es stellt sich eine interessante Frage: In welcher Reihenfolge werden die Knoten in ③ gewählt? Offensichtlich können wir hierzu nicht viel sagen, wenn wir nicht angegeben haben, wie ein $v \in Q$ in ② gewählt werden soll. Hier sind zwei Methoden üblich: die beiden Suchverfahren DEPTH-FIRST-SEARCH (DFS) (Tiefensuche) und BREADTH-FIRST-SEARCH (BFS) (Breitensuche). In DFS wird das zuletzt zu Q hinzugefügte $v \in Q$ gewählt. Mit anderen Worten, es wird Q als ein LIFO-Stack (last-in-first-out) implementiert. In BFS wird das erste zu Q hinzugefügte $v \in Q$ gewählt. Hier wird Q als eine FIFO-Warteschlange (first-in-first-out) implementiert.

Ein dem DFS ähnlicher Algorithmus ist bereits vor 1900 von Trémaux und Tarry beschrieben worden, siehe König [1936]. Wahrscheinlich wurde BFS zum ersten Mal von Moore [1959] erwähnt. Ein mittels DFS bzw. BFS erstellter Baum (R, T) (oder Arboreszenz im gerichteten Fall) heißt **DFS-Baum** bzw. **BFS-Baum**. BFS-Bäume haben die folgende wichtige Eigenschaft:

Proposition 2.18. *Ein BFS-Baum enthält einen kürzesten Weg von s zu jedem von s aus erreichbaren Knoten. Die Werte $\text{dist}_G(s, v)$ können für alle $v \in V(G)$ in linearer Zeit bestimmt werden.*

Beweis: Wir wenden BFS auf (G, s) an und fügen zwei Forderungen hinzu: Am Anfang (in ① des GRAPH-SCANNING-ALGORITHMUS) wird $l(s) := 0$ gesetzt und in ④ $l(w) := l(v) + 1$. Offensichtlich folgt zu jedem Zeitpunkt des Algorithmus $l(v) = \text{dist}_{(R, T)}(s, v)$ für alle $v \in R$. Ferner gilt: Ist v der aktuell gescannte Knoten (der in ② gewählt wurde), so gibt es zu diesem Zeitpunkt keinen Knoten $w \in R$ mit $l(w) > l(v) + 1$ (da die Knoten in einer Reihenfolge mit nicht abnehmenden l -Werten gescannt werden).

Angenommen, bei Terminierung des Algorithmus liegt ein Knoten $w \in V(G)$ mit $\text{dist}_G(s, w) < \text{dist}_{(R, T)}(s, w)$ vor. Unter allen Knoten mit dieser Eigenschaft habe w minimale Entfernung von s in G . Sei P ein kürzester s - w -

Weg in G und sei $e = (v, w)$ oder $e = \{v, w\}$ die letzte Kante in P . So mit ist $\text{dist}_G(s, v) = \text{dist}_{(R, T)}(s, v)$, aber e liegt nicht in T . Ferner gilt $l(w) = \text{dist}_{(R, T)}(s, w) > \text{dist}_G(s, w) = \text{dist}_G(s, v) + 1 = \text{dist}_{(R, T)}(s, v) + 1 = l(v) + 1$. Mit dieser Ungleichung und dem obigen Zwischenresultat folgt, dass w nicht in R lag, als v aus Q entfernt wurde. Wegen der Kante e haben wir hiermit aber einen Widerspruch zu ③. \square

Dieses Resultat wird auch aus der Korrektheit von DIJKSTRAS ALGORITHMUS für das KÜRZESTE-WEGE-PROBLEM folgen. Dijkstras Algorithmus kann man als eine Verallgemeinerung von BFS auf den Fall nichtnegativ gewichteter Kanten betrachten (siehe Abschnitt 7.1).

Wir werden nun zeigen, wie man die starken Zusammenhangskomponenten eines Digraphen bestimmt. Natürlich können diese einfach dadurch berechnet werden, indem man DFS (oder BFS) n mal anwendet. Es ist aber möglich, die starken Zusammenhangskomponenten mit nur zweimaliger Bearbeitung jeder Kante zu bestimmen:

ALGORITHMUS FÜR STARKE ZUSAMMENHANGSKOMPONENTEN

Input: Ein Digraph G .

Output: Eine Funktion $\text{comp} : V(G) \rightarrow \mathbb{N}$ zur Angabe der Zugehörigkeit zu den starken Zusammenhangskomponenten.

-
- ① Setze $R := \emptyset$. Setze $N := 0$.
 - ② **For** alle $v \in V(G)$ **do:** **If** $v \notin R$ **then** **VISIT1**(v).
 - ③ Setze $R := \emptyset$. Setze $K := 0$.
 - ④ **For** $i := |V(G)|$ **down to** 1 **do:**
 If $\psi^{-1}(i) \notin R$ **then** setze $K := K + 1$ und **VISIT2**($\psi^{-1}(i)$).
-

VISIT1(v)

- ① Setze $R := R \cup \{v\}$.
 - ② **For** alle w mit $(v, w) \in E(G)$ **do:**
 If $w \notin R$ **then** **VISIT1**(w).
 - ③ Setze $N := N + 1$, $\psi(v) := N$ und $\psi^{-1}(N) := v$.
-

VISIT2(v)

- ① Setze $R := R \cup \{v\}$.
 - ② **For** alle w mit $(w, v) \in E(G)$ **do:**
 If $w \notin R$ **then** **VISIT2**(w).
 - ③ Setze $\text{comp}(v) := K$.
-

Abbildung 2.3 zeigt ein Beispiel: Die erste Anwendung von DFS scannt die Knoten in der Reihenfolge a, g, b, d, e, f und liefert die Arboreszenz in der Mitte der Abbildung; die Zahlen sind die ψ -Labels. Der einzige von a aus nicht erreichbare Knoten ist c und er bekommt das Label $\psi(c) = 7$ mit dem höchsten Wert. Die zweite DFS-Anwendung beginnt mit c , kann aber keinen anderen Knoten über eine umorientierte Kante erreichen. Also fährt sie mit Knoten a fort, da $\psi(a) = 6$. Jetzt sind b, g und f erreichbar. Schließlich wird e von d aus erreicht. Die starken Zusammenhangskomponenten sind $\{c\}$, $\{a, b, f, g\}$ und $\{d, e\}$.

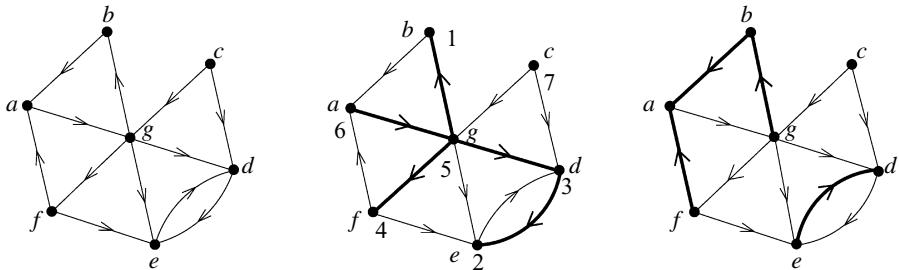


Abbildung 2.3.

Zusammenfassend braucht man also eine DFS-Anwendung zur Bestimmung einer passenden Reihenfolge, während in einer weiteren DFS-Anwendung der umorientierte Graph betrachtet wird und die Knoten in abnehmender Reihenfolge bezüglich dieser Nummerierung abgearbeitet werden. Jede Zusammenhangskomponente des zweiten DFS-Waldes ist eine **Anti-Arboreszenz**, d. h. ein aus einer Arboreszenz durch Umorientierung ihrer Kanten hervorgehender Graph. Wir zeigen nun, dass diese Anti-Arboreszenzen die starken Zusammenhangskomponenten identifizieren.

Satz 2.19. *Der ALGORITHMUS FÜR STARKE ZUSAMMENHANGSKOMPONENTEN identifiziert die starken Zusammenhangskomponenten korrekt in linearer Zeit.*

Beweis: Die Laufzeit ist offensichtlich $O(n+m)$. Natürlich sind Knoten derselben starken Zusammenhangskomponente immer in derselben Komponente eines DFS-Waldes und bekommen deshalb denselben $comp$ -Wert. Wir müssen nun zeigen, dass zwei Knoten u und v mit $comp(u) = comp(v)$ tatsächlich in derselben starken Zusammenhangskomponente liegen. Sei $r(u)$ bzw. $r(v)$ der mit dem höchsten ψ -Label von u bzw. v aus erreichbare Knoten. Da $comp(u) = comp(v)$, d. h. u und v liegen in derselben Anti-Arboreszenz des zweiten DFS-Waldes, so folgt, dass $r := r(u) = r(v)$ die Wurzel dieser Anti-Arboreszenz ist. Somit ist r sowohl von u als auch von v aus erreichbar.

Da r von u aus erreichbar ist und $\psi(r) \geq \psi(u)$, wurde r nicht nach u in der ersten DFS-Anwendung zu R hinzugefügt. Somit enthält der erste DFS-Wald einen r - u -Weg. Mit anderen Worten, u ist von r aus erreichbar. Analog ist v von r aus erreichbar. Insgesamt ist also u von v aus erreichbar und auch umgekehrt, womit bewiesen ist, dass u und v tatsächlich zu derselben starken Zusammenhangskomponente gehören. \square

Es ist interessant, dass dieser Algorithmus auch ein anderes Problem löst, nämlich eine topologische Ordnung für einen azyklischen Digraphen zu finden. Beachte, dass nach der Kontraktion der starken Zusammenhangskomponenten eines Digraphen ein azyklischer Digraph vorliegt. Nach Proposition 2.9 hat dieser azyklische Digraph eine topologische Ordnung. In der Tat wird eine solche Ordnung durch die von dem ALGORITHMUS FÜR STARKE ZUSAMMENHANGSKOMPONENTEN berechneten Zahlen $comp(v)$ gegeben:

Satz 2.20. *Der ALGORITHMUS FÜR STARKE ZUSAMMENHANGSKOMPONENTEN bestimmt eine topologische Ordnung für den durch Kontraktion aller starken Zusammenhangskomponenten eines Digraphen G hervorgehenden Digraph. Insbesondere können wir für einen gegebenen Digraphen in linearer Zeit entweder eine topologische Ordnung finden oder entscheiden, dass es keine solche gibt.*

Beweis: Seien X und Y zwei starke Zusammenhangskomponenten eines Digraphen G und angenommen, der ALGORITHMUS FÜR STARKE ZUSAMMENHANGSKOMPONENTEN berechnet $comp(x) = k_1$ für $x \in X$ und $comp(y) = k_2$ für $y \in Y$ mit $k_1 < k_2$. Wir behaupten, dass $E_G^+(Y, X) = \emptyset$.

Angenommen, es gäbe eine Kante $(y, x) \in E(G)$ mit $y \in Y$ und $x \in X$. Während der zweiten DFS-Anwendung werden alle Knoten in X zu R hinzugefügt, bevor der erste Knoten aus Y hinzugefügt wird. Insbesondere haben wir $x \in R$ und $y \notin R$, wenn die Kante (y, x) während der zweiten DFS-Anwendung gescannt wird. Dies bedeutet aber, dass y zu R hinzugefügt wird bevor K erhöht wird, im Widerspruch zu $comp(y) \neq comp(x)$.

Also bestimmen die von dem ALGORITHMUS FÜR STARKE ZUSAMMENHANGSKOMPONENTEN berechneten $comp$ -Werte eine topologische Ordnung für den durch Kontraktion aller starken Zusammenhangskomponenten hervorgehenden Digraphen. Die zweite Aussage des Satzes folgt nun mit Proposition 2.9 und der Bemerkung, dass ein Digraph genau dann azyklisch ist, wenn seine starken Zusammenhangskomponenten die einelementigen Knotenmengen sind. \square

Ein Algorithmus mit linearer Laufzeit für die Bestimmung der starken Zusammenhangskomponenten wurde zuerst von Karzanov [1970] und Tarjan [1972] präsentiert. Das Problem, eine topologische Ordnung zu finden (oder zu entscheiden, dass es keine gibt), wurde bereits früher von Kahn [1962] und Knuth [1968] gelöst. Sowohl BFS als auch DFS treten als Subroutinen in vielen anderen kombinatorischen Algorithmen auf. Dazu werden wir einige Beispiele in späteren Kapiteln kennenlernen.

Manchmal ist man an mehrfachem Zusammenhang interessiert. Sei $k \geq 2$. Ein ungerichteter Graph mit mehr als k Knoten und der Eigenschaft, dass er nach dem Entfernen von $k - 1$ beliebigen Knoten immer noch zusammenhängend ist, heißt **k -fach zusammenhängend**. Ein Graph mit mindestens zwei Knoten heißt **k -fach kantenzusammenhängend**, falls er nach dem Entfernen von $k - 1$ beliebigen Kanten immer noch zusammenhängend ist. Somit haben wir, dass ein zusammenhängender Graph mit mindestens drei Knoten genau dann 2-fach

zusammenhängend (bzw. 2-fach kantenzusammenhängend) ist, wenn er keinen Artikulationsknoten (bzw. keine Brücke) hat.

Das größte k bzw. l , so dass ein Graph G k -fach zusammenhängend bzw. l -fach kantenzusammenhängend ist, heißt der **Knotenzusammenhang** bzw. der **Kanten-zusammenhang** von G . Hier sagen wir, dass ein Graph einfach zusammenhängend (bzw. einfach kantenzusammenhängend) ist, wenn er zusammenhängend ist. Für einen unzusammenhängenden Graphen sind der Knoten- und Kantenzusammenhang beide Null.

Die **Blöcke** eines ungerichteten Graphen sind die maximalen zusammenhängenden Teilgraphen ohne Artikulationsknoten. Es ist somit jeder Block entweder ein maximaler 2-fach zusammenhängender Teilgraph oder eine Brücke oder ein isolierter Knoten. Zwei Blöcke haben höchstens einen gemeinsamen Knoten, und ferner ist ein zu mehr als einem Block gehörender Knoten ein Artikulationsknoten. Die Blöcke eines ungerichteten Graphen können in linearer Zeit mit einem Algorithmus bestimmt werden, der dem ALGORITHMUS FÜR STARKE ZUSAMMENHANGSKOMPONENTEN recht ähnlich ist, siehe Aufgabe 21. Hier werden wir nun einen schönen Struktursatz für 2-fach zusammenhängende Graphen beweisen. Beginnend mit einem einzigen Knoten, bauen wir Graphen auf durch schrittweises Hinzufügen von sogenannten Ohren:

Definition 2.21. Sei G ein Graph (gerichtet oder ungerichtet). Eine **Ohrenzerlegung** von G ist eine Folge r, P_1, \dots, P_k mit $G = (\{r\}, \emptyset) + P_1 + \dots + P_k$, so dass für alle $i \in \{1, \dots, k\}$ gilt: Jedes P_i ist entweder ein Weg, der genau zwei seiner Knoten, nämlich seine Endknoten, in $\{r\} \cup V(P_1) \cup \dots \cup V(P_{i-1})$ hat, oder ein Kreis, der genau einen seiner Knoten aus $\{r\} \cup V(P_1) \cup \dots \cup V(P_{i-1})$ hat.

Die P_1, \dots, P_k heißen **Ohren**. Ist $k \geq 1$ und P_1 ein Kreis mit Mindestlänge drei und sind P_2, \dots, P_k Wege, so heißt die Ohrenzerlegung **echt**.

Satz 2.22. (Whitney [1932]) Ein ungerichteter Graph ist genau dann 2-fach zusammenhängend, wenn er eine echte Ohrenzerlegung hat.

Beweis: Offensichtlich ist ein Kreis mit Mindestlänge drei 2-fach zusammenhängend. Ferner gilt: Ist G 2-fach zusammenhängend, so auch $G + P$, wobei P ein x - y -Weg ist, $x, y \in V(G)$ und $x \neq y$, denn das Entfernen eines Knotens zerstört nicht den Zusammenhang. Daraus folgern wir, dass ein Graph mit einer echten Ohrenzerlegung 2-fach zusammenhängend ist.

Zum Beweis der Umkehrung, sei G ein 2-fach zusammenhängender Graph. Sei ferner G' der maximale einfache Teilgraph von G ; offensichtlich ist G' auch 2-fach zusammenhängend. Somit kann G' kein Baum sein, d. h. G' enthält einen Kreis. Da G' einfach ist, enthält G' und damit auch G einen Kreis mit Mindestlänge drei. Also sei H ein maximaler Teilgraph von G mit echter Ohrenzerlegung; ein solches H gibt es nach obiger Überlegung.

Angenommen, H wäre nicht aufspannend. Da G zusammenhängend ist, wissen wir dann, dass es eine Kante $e = \{x, y\} \in E(G)$ mit $x \in V(H)$ und $y \notin V(H)$ gibt. Sei z ein Knoten in $V(H) \setminus \{x\}$. Da $G - x$ zusammenhängend ist, gibt es einen Weg P von y nach z in $G - x$. Sei z' der erste zu $V(H)$ gehörende Knoten auf

diesem von y aus durchlaufenen Weg. Dann kann $P_{[y,z']} + e$ als Ohr hinzugefügt werden, im Widerspruch zur Maximalität von H .

Damit ist H aufspannend. Da jede Kante von $E(G) \setminus E(H)$ als Ohr hinzugefügt werden kann, folgern wir, dass $H = G$. \square

Aufgabe 22 enthält ähnliche Charakterisierungen 2-fach zusammenhängender Graphen und stark zusammenhängender Digraphen.

2.4 Eulersche und bipartite Graphen

Die Graphentheorie hat ihren Ursprung in Eulers Arbeit über das Problem, ob man bei einem Spaziergang jede der sieben Brücken von Königsberg genau einmal überqueren kann. Er hat gezeigt, dass dies nicht möglich ist, indem er einen Graphen definierte, einen alle Kanten enthaltenden Spaziergang forderte und beobachtete, dass mehr als zwei Knoten ungeraden Grad haben.

Definition 2.23. Ein **eulerscher Spaziergang** in einem Graphen G ist ein geschlossener jede Kante enthaltender Spaziergang. Ein ungerichteter Graph G heißt **eulersch**, falls jeder Knoten geraden Grad hat. Ein Digraph G heißt **eulersch**, falls $|\delta^-(v)| = |\delta^+(v)|$ für jedes $v \in V(G)$.

Obwohl Euler nur den Notwendigkeitsbeweis führte und auch nicht den Fall eines geschlossenen Spaziergangs explizit betrachtete, wird das folgende berühmte Resultat ihm zugeschrieben:

Satz 2.24. (Euler [1736], Hierholzer [1873]) Ein zusammenhängender (gerichteter oder auch ungerichteter) Graph hat genau dann einen eulerschen Spaziergang, wenn er eulersch ist.

Beweis: Die Gradbedingungen sind notwendig, da ein Knoten, der k -mal in einem eulerschen Spaziergang vorkommt (oder $k+1$ -mal falls er der erste und auch der letzte Knoten ist), den Eingangsgrad k und den Ausgangsgrad k hat, oder den Grad $2k$ im ungerichteten Fall.

Sie sind aber auch hinreichend: Sei $W = v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1}$ ein längster Spaziergang in G , d. h. ein Spaziergang mit maximaler Kantenanzahl. Dann enthält W insbesondere alle in v_{k+1} beginnenden Kanten, und mit den Gradbedingungen folgt $v_{k+1} = v_1$. Somit ist W ein geschlossener Spaziergang. Wir nehmen nun an, dass W nicht alle Kanten enthält. Da G zusammenhängend ist, folgt somit, dass es eine nicht in W vorkommende Kante $e \in E(G)$ gibt, für die aber wenigstens einer ihrer Endknoten, etwa v_i , in W vorkommt. Zusammen mit e bildet nun $v_i, e_i, v_{i+1}, \dots, e_k, v_{k+1} = v_1, e_1, v_2, \dots, e_{i-1}, v_i$ einen Spaziergang der länger als W ist. \square

Der folgende Algorithmus akzeptiert nur zusammenhängende eulersche Graphen als Input. Beachte, dass man in linearer Zeit prüfen kann, ob ein gegebener Graph zusammenhängend (Proposition 2.17) und eulersch ist (trivial). Als erstes

wählt der Algorithmus einen Anfangsknoten und macht dann mit einem rekursiven Verfahren weiter. Zunächst werden wir ihn für ungerichtete Graphen angeben:

EULERS ALGORITHMUS

Input: Ein ungerichteter zusammenhängender eulerscher Graph G .

Output: Ein eulerscher Spaziergang W in G .

-
- ① Wähle ein beliebiges $v_1 \in V(G)$. **Return** $W := \text{EULER}(v_1)$.

$\text{EULER}(v_1)$

- ① Setze $W := v_1$ und $x := v_1$.
 - ② **If** $\delta(x) = \emptyset$ **then go to** ④.
Else sei $e \in \delta(x)$, etwa $e = \{x, y\}$.
 - ③ Setze $W := W, e, y$ und $x := y$. Setze $E(G) := E(G) \setminus \{e\}$ und **go to** ②.
 - ④ Sei $v_1, e_1, v_2, e_2, \dots, v_k, e_k, v_{k+1}$ die Folge W .
For $i := 2$ **to** k **do**: Setze $W_i := \text{EULER}(v_i)$.
 - ⑤ **Return** $W := v_1, e_1, W_2, e_2, \dots, W_k, e_k, v_{k+1}$.
-

Für Digraphen ersetze man ② durch:

- ② **If** $\delta^+(x) = \emptyset$ **then go to** ④.
Else sei $e \in \delta^+(x)$, etwa $e = (x, y)$.

Wir können beide Versionen (die ungerichtete wie auch die gerichtete) gleichzeitig analysieren:

Satz 2.25. EULERS ALGORITHMUS arbeitet korrekt. Die Laufzeit ist $O(m)$, wobei $m = |E(G)|$.

Beweis: Wir beweisen, dass der Schritt $\text{EULER}(v_1)$ (für einen gegebenen ungerichteten zusammenhängenden eulerschen Graphen G und ein $v_1 \in V(G)$) einen eulerschen Spaziergang W in der v_1 enthaltenden Zusammenhangskomponente G_1 von G liefert. Hierzu benutzen wir Induktion über $|E(G)|$, wobei der Fall $E(G) = \emptyset$ trivial ist.

Wegen der Gradbedingungen haben wir $v_{k+1} = x = v_1$, wenn ④ erreicht wird. In diesem Schritt des Algorithmus ist W also ein geschlossener Spaziergang. Sei G' der Graph G in diesem Schritt. Dann ist G' auch eulersch.

Für jede Kante $e \in E(G_1) \cap E(G')$ gibt es ein minimales $i \in \{2, \dots, k\}$ mit der Eigenschaft, dass e und v_i in derselben Zusammenhangskomponente von G' liegen (beachte, dass $v_1 = v_{k+1}$ ein isolierter Knoten in G' ist). Nach der Induktionsvoraussetzung folgt dann, e gehört zu W_i . Also ist der in ⑤ erstellte geschlossene Spaziergang W tatsächlich ein eulerscher Spaziergang in G_1 .

Die Laufzeit ist linear, weil jede Kante sofort nach ihrer Bearbeitung gelöscht wird. \square

EULERS ALGORITHMUS wird einige Male in späteren Kapiteln als Subroutine benutzt.

Manchmal möchte man einen gegebenen Graphen durch Hinzufügen oder Kontraktion von Kanten eulersch machen. Sei G ein ungerichteter Graph und F eine Familie nichtgeordneter Knotenpaare aus $V(G)$ (Kanten oder auch keine Kanten). Es heißt F eine **ungerade Verbindung**, falls $(V(G), E(G) \cup F)$ eulersch ist. Ferner heißt F eine **ungerade Überdeckung**, falls der aus G durch Kontraktion der Knotenmenge jeder Zusammenhangskomponente von $(V(G), F)$ hervorgehende Graph eulersch ist.

Die beiden Begriffe sind in folgendem Sinne äquivalent.

Satz 2.26. (Aoshima und Iri [1977]) *Für jeden ungerichteten Graphen gilt:*

- (a) *Jede ungerade Verbindung ist eine ungerade Überdeckung.*
- (b) *Jede inklusionsminimale ungerade Überdeckung ist eine ungerade Verbindung.*

Beweis: Sei G ein ungerichteter Graph.

Zum Beweis von (a) sei F eine ungerade Verbindung. Durch Kontraktion der Zusammenhangskomponenten von $(V(G), F)$ in G erhalten wir einen Graphen G' . Jede Zusammenhangskomponente von $(V(G), F)$ enthält eine gerade Anzahl von Knoten mit ungeradem Grad (bezüglich F , also auch bezüglich G , da F eine ungerade Verbindung ist). Also hat der resultierende Graph G' nur Knoten mit geradem Grad. Somit ist F eine ungerade Überdeckung.

Zum Beweis von (b) sei F eine inklusionsminimale ungerade Überdeckung. Wegen der Minimalität ist $(V(G), F)$ ein Wald. Wir müssen zeigen, dass $|\delta_F(v)| \equiv |\delta_G(v)| \pmod{2}$ für jedes $v \in V(G)$. Sei also $v \in V(G)$. Seien C_1, \dots, C_k diejenigen Zusammenhangskomponenten von $(V(G), F) - v$, welche einen Knoten w mit $\{v, w\} \in F$ enthalten. Da F ein Wald ist, folgt $k = |\delta_F(v)|$.

Da F eine ungerade Überdeckung ist, ergibt die Kontraktion von $X := V(C_1) \cup \dots \cup V(C_k) \cup \{v\}$ in G einen Knoten mit geradem Grad, d. h. $|\delta_G(X)|$ ist gerade. Andererseits folgt aus der Minimalität von F , dass $F \setminus \{v, w\}$ keine ungerade Überdeckung ist (für beliebiges w mit $\{v, w\} \in F$), also ist $|\delta_G(V(C_i))|$ ungerade für $i = 1, \dots, k$. Aus

$$\begin{aligned} \sum_{i=1}^k |\delta_G(V(C_i))| &= |\delta_G(X)| + |\delta_G(v)| \\ &\quad - 2|E_G(\{v\}, V(G) \setminus X)| + 2 \sum_{1 \leq i < j \leq k} |E_G(C_i, C_j)| \end{aligned}$$

folgt, dass k dieselbe Parität wie $|\delta_G(v)|$ hat. \square

In Abschnitt 12.2 werden wir uns wieder mit dem Problem, einen Graphen eulersch zu machen, befassen.

Eine **Bipartition** eines ungerichteten Graphen G besteht aus disjunkten Mengen A und B , deren Vereinigung $V(G)$ ist, so dass jede Kante einen Endknoten in A

und einen Endknoten in B hat. Ein Graph heißt **bipartit**, wenn er eine Bipartition besitzt. Der einfache bipartite Graph G mit $V(G) = A \cup B$, $|A| = n$, $|B| = m$ und $E(G) = \{\{a, b\} : a \in A, b \in B\}$ wird mit $K_{n,m}$ bezeichnet (der vollständige bipartite Graph). Wenn wir $G = (A \cup B, E(G))$ schreiben, meinen wir, dass $A \cup B$ eine Bipartition von G ist.

Proposition 2.27. (König [1916]) *Ein ungerichteter Graph ist genau dann bipartit, wenn er keinen ungeraden Kreis (Kreis ungerader Länge) enthält. Es gibt einen Algorithmus mit linearer Laufzeit, der für einen gegebenen ungerichteten Graphen G entweder eine Bipartition oder einen ungeraden Kreis findet.*

Beweis: Angenommen, G ist bipartit mit der Bipartition $V(G) = A \cup B$ und der geschlossene Spaziergang $v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1}$ definiert einen Kreis in G . Wir können o. B. d. A. annehmen, dass $v_1 \in A$. Dann gilt aber $v_2 \in B$, $v_3 \in A$, usw. Daraus folgern wir, dass $v_i \in A$ genau dann, wenn i ungerade ist. Aber $v_{k+1} = v_1 \in A$, also ist k gerade.

Zum Beweis, dass die Bedingung hinreichend ist, können wir annehmen, dass G zusammenhängend ist, da ein Graph genau dann bipartit ist, wenn jede Zusammenhangskomponente bipartit ist (und die Zusammenhangskomponenten können nach Proposition 2.17 in linearer Zeit bestimmt werden). Wir wählen nun einen beliebigen Knoten $s \in V(G)$ und wenden das BFS-Verfahren auf (G, s) an, um die Entfernung von s nach v für alle $v \in V(G)$ zu bestimmen (siehe Proposition 2.18). Sei T der resultierende BFS-Baum. Wir definieren $A := \{v \in V(G) : \text{dist}_G(s, v) \text{ ist gerade}\}$ und $B := V(G) \setminus A$.

Falls es eine Kante $e = \{x, y\}$ in $G[A]$ oder $G[B]$ gibt, so bildet der x - y -Weg in T zusammen mit e einen ungeraden Kreis in G . Falls es keine solche Kante gibt, so liegt eine Bipartition vor. \square

2.5 Planarität

Man zeichnet Graphen sehr oft in der Ebene. Ein Graph heißt planar, falls er in der Ebene ohne Kantenüberquerungen gezeichnet werden kann. Um diesen Begriff zu präzisieren, benötigen wir die folgenden topologischen Definitionen:

Definition 2.28. Eine **einfache Jordankurve** ist das Bild einer stetigen injektiven Funktion $\varphi : [0, 1] \rightarrow \mathbb{R}^2$; ihre **Endpunkte** sind $\varphi(0)$ und $\varphi(1)$. Eine **geschlossene Jordankurve** ist das Bild einer stetigen Funktion $\varphi : [0, 1] \rightarrow \mathbb{R}^2$ mit $\varphi(0) = \varphi(1)$ und $\varphi(\tau) \neq \varphi(\tau')$ für $0 \leq \tau < \tau' < 1$. Ein **polygonaler Streckenzug** ist eine einfache Jordankurve, die aus der Vereinigung endlich vieler Intervalle (geradliniger Segmente) besteht. Ein **Polygon** ist eine geschlossene Jordankurve, die aus der Vereinigung endlich vieler Intervalle besteht.

Sei $R = \mathbb{R}^2 \setminus J$, wobei J aus der Vereinigung endlich vieler Intervalle besteht. Die **zusammenhängenden Gebiete** von R werden als Äquivalenzklassen von Punkten in R definiert, wobei zwei Punkte äquivalent sind, wenn sie durch einen in R verlaufenden polygonalen Streckenzug verbunden werden können.

Definition 2.29. Eine planare Einbettung eines Graphen G besteht aus einer injektiven Abbildung $\psi : V(G) \rightarrow \mathbb{R}^2$ und einem polygonalen Streckenzug J_e für jedes $e = \{x, y\} \in E(G)$ mit den Endpunkten $\psi(x)$ und $\psi(y)$, so dass für jedes $e = \{x, y\} \in E(G)$

$$(J_e \setminus \{\psi(x), \psi(y)\}) \cap \left(\{\psi(v) : v \in V(G)\} \cup \bigcup_{e' \in E(G) \setminus \{e\}} J_{e'} \right) = \emptyset$$

gilt. Ein Graph heißt **planar**, wenn er eine planare Einbettung hat.

Sei G ein (planarer) Graph mit einer festen planaren Einbettung $\Phi = (\psi, (J_e)_{e \in E(G)})$. Entfernt man die Punkte und polygonalen Streckenzüge aus der Ebene, dann spaltet sich der Rest

$$R := \mathbb{R}^2 \setminus \left(\{\psi(v) : v \in V(G)\} \cup \bigcup_{e \in E(G)} J_e \right)$$

in offene zusammenhängende Gebiete auf, die sogenannten **Gebiete** von Φ .

Es ist z. B. K_4 offensichtlich planar, es wird sich aber herausstellen, dass K_5 nicht planar ist. Nach Aufgabe 29 macht die Beschränkung auf polygonale Streckenzüge statt beliebiger Jordankurven keinen wesentlichen Unterschied. Später werden wir zeigen, dass es für einfache Graphen in der Tat genügt, nur geradlinige Segmente zu betrachten.

Es ist unser Ziel, planare Graphen zu charakterisieren. In Anlehnung an Thomassen [1981] beweisen wir zunächst das folgende topologische Resultat, eine Version des Jordanschen Kurvensatzes:

Satz 2.30. Ist J ein Polygon, so spaltet sich $\mathbb{R}^2 \setminus J$ in genau zwei zusammenhängende Gebiete auf, die beide von J berandet werden. Ist J ein polygonaler Streckenzug, so hat $\mathbb{R}^2 \setminus J$ nur ein zusammenhängendes Gebiet.

Beweis: Sei J ein Polygon, $p \in \mathbb{R}^2 \setminus J$ und $q \in J$. Dann gibt es einen p und q verbindenden polygonalen Streckenzug in $(\mathbb{R}^2 \setminus J) \cup \{q\}$: Von p ausgehend folgt man der geraden Linie nach q bis in die Nähe von J , dann geht man weiter innerhalb der näheren Umgebung von J . (Hier benutzen wir das elementare topologische Resultat, dass disjunkte kompakte Mengen, insbesondere nicht benachbarte Intervalle von J , einen positiven Abstand zu einander haben.) Daraus folgern wir, dass p in demselben zusammenhängenden Gebiet von $\mathbb{R}^2 \setminus J$ liegt wie manche Punkte, die beliebig nahe zu q sind.

Es ist J die Vereinigung endlich vieler Intervalle; eines oder zwei von ihnen enthalten q . Sei $\epsilon > 0$ so gewählt, dass die Kugel mit Mittelpunkt q und Radius ϵ mit keinem der anderen Intervalle von J einen nichtleeren Durchschnitt hat; dann hat diese Kugel offensichtlich mit höchstens zwei der zusammenhängenden Gebiete einen nichtleeren Durchschnitt. Da $p \in \mathbb{R}^2 \setminus J$ und $q \in J$ beliebig gewählt wurden, folgt somit, dass es höchstens zwei Gebiete gibt und dass beide Gebiete von J berandet werden.

Obiges gilt auch, falls J ein polygonaler Streckenzug und q ein Endpunkt von J ist, woraus folgt, dass $\mathbb{R}^2 \setminus J$ in diesem Fall nur ein zusammenhängendes Gebiet hat.

Betrachten wir nun wieder den Fall, dass J ein Polygon ist. Es bleibt zu zeigen, dass $\mathbb{R}^2 \setminus J$ mehr als ein Gebiet hat. Sei l_α der von p ausgehende Strahl mit Winkel α , wobei $p \in \mathbb{R}^2 \setminus J$ und α beliebig sind. Es ist $J \cap l_\alpha$ eine Menge von Punkten oder abgeschlossenen Intervallen. Sei $cr(p, l_\alpha)$ die Anzahl derjenigen dieser Punkte oder Intervalle, in denen das Polygon J von einer zu der anderen Seite von l_α wechselt (d. h. wie oft J mittels eines Punktes oder Intervalls l_α überquert, z. B. zweimal in Abb. 2.4).

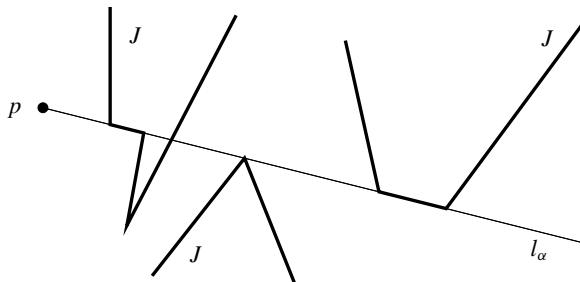


Abbildung 2.4.

Beachte: für einen beliebigen Winkel α sind

$$\left| \lim_{\epsilon \rightarrow 0, \epsilon > 0} cr(p, l_{\alpha+\epsilon}) - cr(p, l_\alpha) \right| \quad \text{und} \quad \left| \lim_{\epsilon \rightarrow 0, \epsilon < 0} cr(p, l_{\alpha+\epsilon}) - cr(p, l_\alpha) \right|$$

gerade ganze Zahlen, nämlich zweimal die Anzahl derjenigen Punkte und Intervalle von $J \cap l_\alpha$, in denen das Polygon J nicht von einer zu der anderen Seite von l_α wechselt, d. h. es sind für jeden dieser Punkte und jedes dieser Intervalle die beiden sich anschließenden J -Intervalle des Polygons auf derselben Seite von l_α (beide entweder links oder rechts von l_α). Demnach ist $g(p, \alpha) := (cr(p, l_\alpha) \bmod 2)$ eine stetige Funktion von α , also ist sie eine konstante Funktion von α , die wir mit $g(p)$ bezeichnen. Offensichtlich ist $g(p)$ konstant für Punkte p auf jeder Geraden, die J nicht schneidet, also ist sie in jedem Gebiet konstant. Es ist jedoch $g(p) \neq g(q)$ für Punkte p und q mit der Eigenschaft, dass das p und q verbindende geradlinige Segment des Polygon J genau einmal schneidet. Also gibt es in der Tat zwei Gebiete. \square

Genau eines der Gebiete, das **äußere Gebiet**, ist unbeschränkt.

Proposition 2.31. *Sei G ein 2-fach zusammenhängender Graph mit einer planaren Einbettung Φ . Dann wird jedes Gebiet von einem Kreis berandet und jede Kante ist auf dem Rand von genau zwei Gebieten. Ferner ist die Anzahl der Gebiete gleich $|E(G)| - |V(G)| + 2$.*

Beweis: Nach Satz 2.30 sind beide Aussagen richtig, falls G ein Kreis ist. Für allgemeine 2-fach zusammenhängende Graphen führen wir den Beweis mittels Induktion über die Kantenanzahl unter Benutzung von Satz 2.22. Betrachte eine echte Ohrenzerlegung von G und sei P das letzte Ohr, etwa ein Weg mit den Endknoten x und y . Sei G' der Graph vor dem Hinzufügen des letzten Ohres und Φ' die Beschränkung von Φ auf G' .

Sei $\Phi = (\psi, (J_e)_{e \in E(G)})$. Sei ferner F' das $\bigcup_{e \in E(P)} J_e \setminus \{\psi(x), \psi(y)\}$ enthaltende Gebiet von Φ' . Nach der Induktionsvoraussetzung ist F' von einem Kreis C berandet. Die Punkte x und y liegen in C , also ist C die Vereinigung zweier x - y -Wege Q_1 und Q_2 in G' . Nun wenden wir Satz 2.30 auf die beiden Kreise $Q_1 + P$ und $Q_2 + P$ an. Damit folgt

$$F' \cup \{\psi(x), \psi(y)\} = F_1 \dot{\cup} F_2 \dot{\cup} \bigcup_{e \in E(P)} J_e,$$

wobei F_1 und F_2 zwei von den Kreisen $Q_1 + P$ bzw. $Q_2 + P$ berandete Gebiete von G sind. Somit hat G ein Gebiet mehr als G' . Mit $|E(G) \setminus E(G')| = |V(G) \setminus V(G')| + 1$ folgt der Induktionsschritt. \square

Dieser Beweis stammt von Tutte. Aus ihm folgt auch leicht, dass die Kreise, welche die beschränkten Gebiete beranden, eine Kreisbasis bilden (siehe Aufgabe 30). Die letzte Aussage von Proposition 2.31 ist die bekannte Eulersche Formel; sie gilt für allgemeine zusammenhängende Graphen:

Satz 2.32. (Euler [1758], Legendre [1794]) *Für einen planaren zusammenhängenden Graphen G mit beliebiger Einbettung ist die Anzahl der Gebiete gleich $|E(G)| - |V(G)| + 2$.*

Beweis: Mit Proposition 2.31 haben wir die Aussage bereits für 2-fach zusammenhängende Graphen bewiesen. Ferner ist die Aussage für $|V(G)| = 1$ trivial, und für $|E(G)| = 1$ folgt sie aus Satz 2.30. Haben wir $|V(G)| = 2$ und $|E(G)| \geq 2$, so können wir eine Kante e unterteilen und damit sowohl die Knotenzahl als auch die Kantenanzahl um eins erhöhen und den Graphen 2-fach zusammenhängend machen. Dann wenden wir Proposition 2.31 an.

Also können wir jetzt annehmen, dass G einen Artikulationsknoten x hat. Nun wenden wir Induktion über die Knotenzahl an. Sei Φ eine Einbettung von G . Seien C_1, \dots, C_k die Zusammenhangskomponenten von $G - x$ und sei Φ_i die Beschränkung von Φ auf $G_i := G[V(C_i) \cup \{x\}]$ für $i = 1, \dots, k$.

Die Menge der inneren (beschränkten) Gebiete von Φ ist die disjunkte Vereinigung der Mengen der inneren Gebiete von Φ_i , $i = 1, \dots, k$. Durch Anwendung der Induktionsvoraussetzung auf (G_i, Φ_i) , $i = 1, \dots, k$, folgt, dass die Gesamtanzahl der inneren Gebiete von (G, Φ) gleich

$$\sum_{i=1}^k (|E(G_i)| - |V(G_i)| + 1) = |E(G)| - \sum_{i=1}^k |V(G_i) \setminus \{x\}| = |E(G)| - |V(G)| + 1$$

ist. Durch Hinzunahme des äußeren Gebietes folgt der Beweis. \square

Insbesondere ist die Anzahl der Gebiete unabhängig von der Einbettung. Der durchschnittliche Grad eines einfachen planaren Graphen ist kleiner als 6:

Korollar 2.33. Sei G ein 2-fach zusammenhängender einfacher planarer Graph, dessen kürzester Kreis die Länge k hat (man sagt auch, dass G die **Taille** k hat). Dann hat G höchstens $(n-2)\frac{k}{k-2}$ Kanten. Jeder einfache planare Graph mit $n \geq 3$ Knoten hat höchstens $3n - 6$ Kanten.

Beweis: Zunächst nehmen wir an, dass G 2-fach zusammenhängend ist. Sei Φ eine Einbettung von G und r die Anzahl der Gebiete. Mit der Eulerschen Formel (Satz 2.32) folgt $r = |E(G)| - |V(G)| + 2$. Nach Proposition 2.31 wird jedes Gebiet von einem Kreis berandet, d. h. von mindestens k Kanten, und jede Kante ist auf dem Rand von genau zwei Gebieten. Also ist $kr \leq 2|E(G)|$. Diese beiden Resultate ergeben $|E(G)| - |V(G)| + 2 \leq \frac{2}{k}|E(G)|$, woraus $|E(G)| \leq (n-2)\frac{k}{k-2}$ folgt.

Ist G nicht 2-fach zusammenhängend, so fügen wir Kanten zwischen nicht benachbarten Knoten hinzu, um G 2-fach zusammenhängend zu machen ohne die Planarität zu zerstören. Nach dem ersten Teil haben wir höchstens $(n-2)\frac{3}{3-2}$ Kanten, inklusive der neuen. \square

Nun zeigen wir, dass manche Graphen nicht planar sind:

Korollar 2.34. Weder K_5 noch $K_{3,3}$ ist planar.

Beweis: Dies folgt sofort aus Korollar 2.33: K_5 hat fünf Knoten und $10 > 3 \cdot 5 - 6$ Kanten; $K_{3,3}$ ist 2-fach zusammenhängend, hat Taille 4 (da $K_{3,3}$ bipartit ist) und $9 > (6-2)\frac{4}{4-2}$ Kanten. \square



Abbildung 2.5.

Abbildung 2.5 zeigt diese zwei Graphen; sie sind die kleinsten nicht planaren Graphen. Wir werden beweisen, dass jeder nicht planare Graph in einem gewissen Sinne K_5 oder $K_{3,3}$ enthält. Um dies zu präzisieren, benötigen wir den folgenden Begriff:

Definition 2.35. Seien G und H zwei ungerichtete Graphen. Es ist G ein **Minor** von H , wenn es einen Teilgraphen H' von H und eine Partition $V(H') = V_1 \cup \dots \cup V_k$ der Knotenmenge von H' gibt, wobei die V_1, \dots, V_k zusammenhängende Teilmengen mit der Eigenschaft sind, dass die Kontraktion aller V_i einen zu G isomorphen Graphen liefert.

Mit anderen Worten, G ist ein Minor von H , falls G aus H mittels einer Folge von Operationen der folgenden Art entsteht: Das Entfernen eines Knotens oder einer Kante, oder die Kontraktion einer Kante. Da keine dieser Operationen die Planarität zerstört, ist der Minor eines planaren Graphen wieder planar. Damit kann ein Graph, der K_5 oder $K_{3,3}$ als Minor enthält, nicht planar sein. Der Satz von Kuratowski besagt, dass die Umkehrung auch gilt. Wir werden zuerst 3-fach zusammenhängende Graphen betrachten und beginnen mit dem folgenden Lemma (welches den Kern des sogenannten Rad-Satzes von Tutte bildet):

Lemma 2.36. (Tutte [1961], Thomassen [1980]) *Sei G ein 3-fach zusammenhängender Graph mit mindestens fünf Knoten. Dann gibt es eine Kante e , so dass G/e auch 3-fach zusammenhängend ist.*

Beweis: Angenommen, es gäbe keine solche Kante. Dann gibt es für jede Kante $e = \{v, w\}$ einen Knoten x , so dass $G - \{v, w, x\}$ unzusammenhängend ist, d. h. eine Zusammenhangskomponente C mit $|V(C)| < |V(G)| - 3$ hat. Wähle e , x und C so, dass $|V(C)|$ minimal ist.

Der Knoten x hat einen Nachbar y in C , da C sonst eine Zusammenhangskomponente von $G - \{v, w\}$ wäre (aber G ist 3-fach zusammenhängend). Aus unserer Annahme folgt, dass $G/\{x, y\}$ nicht 3-fach zusammenhängend ist, d. h. es gibt einen Knoten z , so dass $G - \{x, y, z\}$ unzusammenhängend ist. Da $\{v, w\} \in E(G)$, gibt es eine Zusammenhangskomponente D von $G - \{x, y, z\}$, die weder v noch w enthält.

Es enthält D aber einen Nachbarn d von y , da D sonst eine Zusammenhangskomponente von $G - \{x, z\}$ wäre (wiederum im Widerspruch zum 3-fachen Zusammenhang von G). Also ist $d \in V(D) \cap V(C)$ und somit ist D ein Teilgraph von C . Da $y \in V(C) \setminus V(D)$, haben wir einen Widerspruch zur Minimalität von $|V(C)|$. \square

Satz 2.37. (Kuratowski [1930], Wagner [1937]) *Ein 3-fach zusammenhängender Graph ist genau dann planar, wenn er weder K_5 noch $K_{3,3}$ als Minor enthält.*

Beweis: Da die Notwendigkeit nach obigem klar ist, brauchen wir nur noch zu zeigen, dass die Bedingung hinreichend ist. Da K_4 offensichtlich planar ist, benutzen wir Induktion über die Knotenzahl: Sei G ein 3-fach zusammenhängender Graph mit mindestens fünf Knoten, der aber kein K_5 oder $K_{3,3}$ als Minor hat.

Nach Lemma 2.36 gibt es eine Kante $e = \{v, w\}$, so dass G/e 3-fach zusammenhängend ist. Sei $\Phi = (\psi, (J_{e'})_{e' \in E(G/e)})$ eine planare Einbettung von G/e ; eine solche gibt es nach der Induktionsvoraussetzung. Sei x der durch die Kontraktion der Kante e entstehende Knoten in G/e . Betrachte nun $(G/e) - x$ mit der Beschränkung von Φ als planare Einbettung. Da $(G/e) - x$ 2-fach zusammenhängend ist, wird jedes Gebiet nach Proposition 2.31 von einem Kreis berandet. Insbesondere wird das den Punkt $\psi(x)$ enthaltende Gebiet von einem Kreis C berandet.

Seien $y_1, \dots, y_k \in V(C)$ die Nachbarn von v , die verschieden von w und in zyklischer Reihenfolge nummeriert sind. Wir partitionieren C nun in paarweise kantendisjunkte Wege P_i , $i = 1, \dots, k$, so dass P_i ein y_i - y_{i+1} -Weg ist ($y_{k+1} := y_1$).

Angenommen, es gibt einen Index $i \in \{1, \dots, k\}$ mit $\Gamma(w) \subseteq \{v\} \cup V(P_i)$ ($\Gamma(w)$ ist die Menge der Nachbarn von w). Dann kann man leicht eine planare Einbettung von G konstruieren, indem man Φ entsprechend ändert.

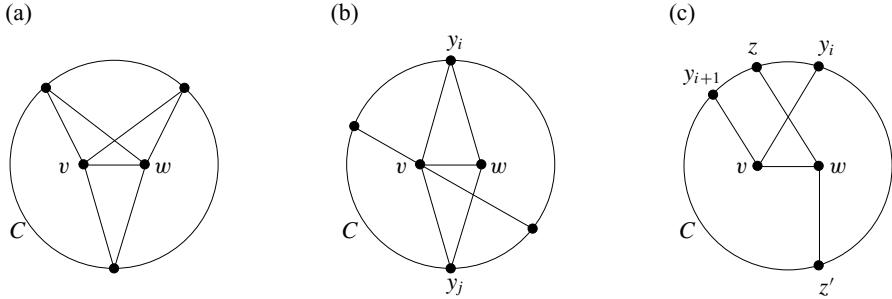


Abbildung 2.6.

Wir werden zeigen, dass alle anderen Fälle unmöglich sind. Erstens: Hat w drei Nachbarn unter den y_1, \dots, y_k , so gibt es einen K_5 -Minor (siehe Abb. 2.6(a)).

Als Nächstes: Gilt $\Gamma(w) = \{v, y_i, y_j\}$ für irgendwelche $i < j$, so folgt $i+1 < j$ und $(i, j) \neq (1, k)$ (sonst würden y_i und y_j beide auf P_i oder P_j liegen); siehe Abb. 2.6(b). Andernfalls gibt es einen Nachbarn z von w in $V(P_i) \setminus \{y_i, y_{i+1}\}$ für irgendein i und einen weiteren Nachbarn $z' \notin V(P_i)$ (siehe Abb. 2.6(c)). In beiden Fällen haben wir vier Knoten y, z, y', z' auf C in dieser zyklischen Reihenfolge, mit $y, y' \in \Gamma(v)$ und $z, z' \in \Gamma(w)$. Dies bedeutet aber, dass es einen $K_{3,3}$ -Minor gibt. \square

Aus diesem Beweis folgt ziemlich direkt, dass jeder 3-fach zusammenhängende einfache planare Graph eine planare Einbettung hat mit der Eigenschaft, dass jede Kante als geradlinige Strecke und jedes Gebiet, außer dem äußeren, als konvexe Region eingebettet wird (siehe Aufgabe 33(a)). Der allgemeine Fall des Satzes von Kuratowski kann auf den 3-fach zusammenhängenden Fall zurückgeführt werden, indem man planare Einbettungen der maximalen 3-fach zusammenhängenden Teilgraphen zusammenklebt, oder auch mittels folgendem Lemma:

Lemma 2.38. (Thomassen [1980]) *Sei G ein Graph mit mindestens fünf Knoten, der weder 3-fach zusammenhängend ist noch K_5 oder $K_{3,3}$ als Minor enthält. Dann gibt es zwei nicht benachbarte Knoten $v, w \in V(G)$, so dass $G + e$, wobei $e = \{v, w\}$ eine neue Kante ist, ebenso weder K_5 noch $K_{3,3}$ als Minor enthält.*

Beweis: Wir verwenden Induktion über $|V(G)|$. Sei G wie oben. O. B. d. A. können wir annehmen, dass G einfach ist. Ist G unzusammenhängend, so brauchen wir nur eine Kante e hinzu zu fügen, die zwei verschiedene Zusammenhangskomponenten verbindet. Also werden wir nun annehmen, dass G zusammenhängend ist. Da G nicht 3-fach zusammenhängend ist, gibt es eine Menge $X = \{x, y\}$ zweier Knoten, so dass $G - X$ unzusammenhängend ist. (Ist G nicht einmal 2-fach

zusammenhängend, so können wir für x einen Artikulationsknoten und für y einen Nachbarn von x wählen.) Sei C eine Zusammenhangskomponente von $G - X$, $G_1 := G[V(C) \cup X]$ und $G_2 := G - V(C)$. Zunächst beweisen wir folgendes Resultat:

Behauptung: Seien $v, w \in V(G_1)$ zwei Knoten, so dass das Hinzufügen einer Kante $e = \{v, w\}$ zu G einen $K_{3,3}$ - oder K_5 -Minor erzeugt. Dann enthält mindestens einer der Graphen $G_1 + e + f$ und $G_2 + f$ einen K_5 - oder $K_{3,3}$ -Minor, wobei f eine neue die Knoten x und y verbindende Kante ist.

Um diese Behauptung zu beweisen, seien $v, w \in V(G_1)$ und $e = \{v, w\}$. Angenommen, es gibt paarweise disjunkte zusammenhängende Knotenmengen Z_1, \dots, Z_t von $G + e$, so dass man nach deren Kontraktion einen K_5 - ($t = 5$) oder $K_{3,3}$ - ($t = 6$) Teilgraphen hat.

Beachte, dass es unmöglich ist, dass $Z_i \subseteq V(G_1) \setminus X$ und $Z_j \subseteq V(G_2) \setminus X$ für irgendwelche $i, j \in \{1, \dots, t\}$: Wir hätten sonst, dass die Familie derjenigen Z_k mit $Z_k \cap X \neq \emptyset$ (es gibt höchstens zwei solche) Z_i und Z_j trennen würde, im Widerspruch zu der Tatsache, dass K_5 und $K_{3,3}$ beide 3-fach zusammenhängend sind.

Also gibt es zwei Fälle: Falls keines der Z_1, \dots, Z_t eine Teilmenge von $V(G_2) \setminus X$ ist, dann enthält $G_1 + e + f$ auch einen K_5 - oder $K_{3,3}$ -Minor: Man braucht nur $Z_i \cap V(G_1)$ ($i = 1, \dots, t$) zu betrachten.

Analog folgt: Falls keines der Z_1, \dots, Z_t eine Teilmenge von $V(G_1) \setminus X$ ist, dann enthält $G_2 + f$ einen K_5 - oder $K_{3,3}$ -Minor: Betrachte $Z_i \cap V(G_2)$ ($i = 1, \dots, t$).

Damit ist die Behauptung bewiesen. Nun betrachten wir zunächst den Fall, dass G einen Artikulationsknoten x enthält und y ein Nachbar von x ist. Wir wählen einen zweiten Nachbarn z von x , so dass y und z in verschiedenen Zusammenhangskomponenten von $G - x$ liegen. O. B. d. A. können wir annehmen, dass $z \in V(G_1)$. Angenommen, das Hinzufügen von $e = \{y, z\}$ erzeugt einen K_5 - oder $K_{3,3}$ -Minor. Nach obiger Behauptung enthält mindestens einer der Graphen $G_1 + e$ und G_2 einen K_5 - oder $K_{3,3}$ -Minor (eine Kante $\{x, y\}$ ist bereits vorhanden). Damit folgt aber, dass G_1 oder G_2 , also auch G , einen K_5 - oder $K_{3,3}$ -Minor enthält, im Widerspruch zu unserer Annahme.

Also können wir annehmen, dass G 2-fach zusammenhängend ist. Beachte: Es wurden $x, y \in V(G)$ so gewählt, dass $G - \{x, y\}$ unzusammenhängend ist. Ist $\{x, y\} \notin E(G)$, so fügen wir einfach eine Kante $f = \{x, y\}$ hinzu. Erzeugt dies einen K_5 - oder $K_{3,3}$ -Minor, so folgt aus der Behauptung, dass $G_1 + f$ oder $G_2 + f$ einen solchen Minor enthält. Da es sowohl in G_1 als auch in G_2 einen x - y -Weg gibt (sonst würden wir einen Artikulationsknoten von G haben), folgt, dass es einen K_5 - oder $K_{3,3}$ -Minor in G gibt, womit wir wiederum einen Widerspruch haben.

Also können wir annehmen, dass $f = \{x, y\} \in E(G)$. Nun nehmen wir an, dass mindestens einer der Graphen G_i ($i \in \{1, 2\}$) nicht planar ist. Somit hat dieser Graph G_i mindestens fünf Knoten. Da er aber keinen K_5 - oder $K_{3,3}$ -Minor enthält (dieser würde dann auch ein Minor von G sein), so folgt mit Satz 2.37, dass G_i

nicht 3-fach zusammenhängend ist. Also können wir die Induktionsvoraussetzung auf G_i anwenden. Nach der Behauptung folgt: Erzeugt das Hinzufügen einer Kante in G_i keinen K_5 - oder $K_{3,3}$ -Minor in G_i , so auch nicht in G .

Also können wir annehmen, dass G_1 und G_2 beide planar sind; seien Φ_1 und Φ_2 planare Einbettungen. Sei F_i ein Gebiet von Φ_i mit f auf seinem Rand und z_i ein weiterer Knoten auf dem Rand von F_i , $z_i \notin \{x, y\}$ ($i = 1, 2$). Wir behaupten nun, dass das Hinzufügen einer Kante $\{z_1, z_2\}$ (siehe Abb. 2.7) keinen K_5 - oder $K_{3,3}$ -Minor erzeugt.

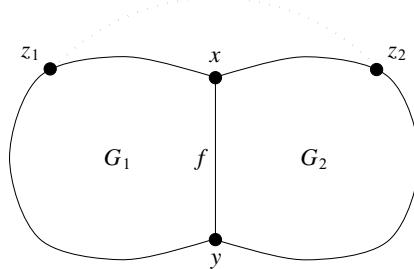


Abbildung 2.7.

Angenommen, das Gegenteil gelte, dass also das Hinzufügen von $\{z_1, z_2\}$ und die Kontraktion einiger paarweise disjunkter zusammenhängender Knotenmengen Z_1, \dots, Z_t einen K_5 - ($t = 5$) oder $K_{3,3}$ - ($t = 6$) Teilgraphen erzeugen würden.

Zunächst nehmen wir an, dass höchstens eines der Z_i eine Teilmenge von $V(G_1) \setminus \{x, y\}$ ist. Dann enthält der Graph G'_2 , welcher aus G_2 durch das Hinzufügen eines Knotens w und der Kanten von w nach x, y und z_2 hervorgeht, auch einen K_5 - oder $K_{3,3}$ -Minor. (Hier entspricht w der kontrahierten Menge $Z_i \subseteq V(G_1) \setminus \{x, y\}$.) Dies ist jedoch ein Widerspruch, da es eine planare Einbettung von G'_2 gibt: Man braucht nur Φ_2 dahingehend zu ergänzen, dass man w in F_2 platziert.

Also können wir annehmen, dass $Z_1, Z_2 \subseteq V(G_1) \setminus \{x, y\}$. Analog können wir annehmen, dass $Z_3, Z_4 \subseteq V(G_2) \setminus \{x, y\}$. O. B. d. A. können wir annehmen, dass $z_1 \notin Z_1$ und $z_2 \notin Z_3$. Dann kann es kein K_5 geben, da Z_1 und Z_3 nicht benachbart sind. Ferner sind Z_5 und Z_6 die einzige möglichen gemeinsamen Nachbarn von Z_1 und Z_3 . Da zwei Knoten von $K_{3,3}$ aber entweder benachbart sind oder drei gemeinsame Nachbarn haben, ist ein $K_{3,3}$ -Minor auch unmöglich. \square

Der Satz von Kuratowski folgt nun aus Satz 2.37 und Lemma 2.38:

Satz 2.39. (Kuratowski [1930], Wagner [1937]) *Ein ungerichteter Graph ist genau dann planar, wenn er weder K_5 noch $K_{3,3}$ als Minor enthält.* \square

Tatsächlich hat Kuratowski eine stärkere Version bewiesen (siehe Aufgabe 34). Der Beweis kann ohne viel Mühe als ein polynomieller Algorithmus formuliert werden (siehe Aufgabe 33(b)). Es gibt sogar einen Algorithmus mit linearer Laufzeit:

Satz 2.40. (Hopcroft und Tarjan [1974]) Es gibt einen Algorithmus mit linearer Laufzeit, um eine planare Einbettung für einen gegebenen Graphen zu finden oder zu entscheiden, dass er nicht planar ist.

2.6 Planare Dualität

Wir werden nun einen wichtigen Dualitätsbegriff einführen. In diesem Abschnitt können Graphen sogenannte Schleifen enthalten, d. h. Kanten mit übereinstimmenden Endknoten. In einer planaren Einbettung werden Schleifen natürlich durch Polygone anstatt durch polygonale Streckenzüge dargestellt.

Beachte, dass die Eulersche Formel (Satz 2.32) auch für Graphen mit Schleifen gilt: Dies folgt aus der Tatsache, dass die Unterteilung einer Schleife e (d. h. man ersetzt $e = \{v, v\}$ durch zwei parallele Kanten $\{v, w\}, \{w, v\}$, wobei w ein neuer Knoten ist) und die Justierung der Einbettung (d. h. man ersetzt das Polygon J_e durch zwei polygonale Streckenzüge, deren Vereinigung J_e ist) sowohl die Anzahl der Knoten als auch die Anzahl der Kanten um eins erhöht, die Anzahl der Gebiete aber unverändert lässt.

Definition 2.41. Sei G ein gerichteteter oder ungerichteter Graph, möglicherweise mit Schleifen, und sei $\Phi = (\psi, (J_e)_{e \in E(G)})$ eine planare Einbettung von G . Das **planare Dual** G^* von G ist der Graph, dessen Knoten die Gebiete von Φ sind und dessen Kantenmenge gleich $\{e^* : e \in E(G)\}$ ist, wobei e^* diejenigen Gebiete von G verbindet, auf deren Rand J_e liegt (gibt es nur ein solches Gebiet, so ist e^* eine Schleife). Im gerichteten Fall, etwa für $e = (v, w)$, orientieren wir $e^* = (F_1, F_2)$ so, dass F_1 das „rechts liegende“ Gebiet ist, wenn man J_e von $\psi(v)$ aus nach $\psi(w)$ durchläuft.

Es ist G^* wieder planar. Es gibt sogar eine planare Einbettung $(\psi^*, (J_{e^*})_{e^* \in E(G^*)})$ von G^* , so dass $\psi^*(F) \in F$ für alle Gebiete F von Φ gilt und ferner für jedes $e \in E(G)$ Folgendes gilt:

$$J_{e^*} \cap \left(\{\psi(v) : v \in V(G)\} \cup \bigcup_{f \in E(G) \setminus \{e\}} J_f \right) = \emptyset,$$

$|J_{e^*} \cap J_e| = 1$, und für jede Schleife e^* enthält das durch J_{e^*} berandete Gebiet genau einen Endknoten von e . Eine solche Einbettung heißt eine **Standardeinbettung** von G^* .

Das planare Dual eines Graphen hängt durchaus von der Einbettung ab: Betrachte die beiden verschiedenen Einbettungen desselben Graphen in Abb. 2.8. Die resultierenden planaren Duale sind nicht isomorph, da das zweite einen Knoten des Grades vier hat (dieser entspricht dem äußeren Gebiet) während das erste 3-regulär ist.

Proposition 2.42. Sei G ein ungerichteter zusammenhängender planarer Graph mit einer festen Einbettung. Sei G^* das planare Dual von G mit einer Standard-einbettung. Dann gilt $(G^*)^* = G$.

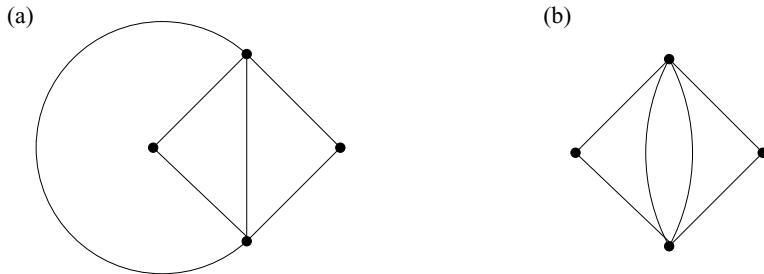


Abbildung 2.8.

Beweis: Sei $(\psi, (J_e)_{e \in E(G)})$ eine feste Einbettung von G und $(\psi^*, (J_{e^*})_{e^* \in E(G^*)})$ eine Standardeinbettung von G^* . Sei ferner F ein Gebiet von G^* . Der Rand von F enthält J_{e^*} für mindestens eine Kante e^* , also enthält F den ψ -Bildpunkt eines der Endknoten von e . Damit enthält jedes Gebiet von G^* mindestens einen Knoten von G .

Wenden wir nun die Eulersche Formel (Satz 2.32) auf G^* und auf G an, so folgt, dass die Anzahl der Gebiete von G^* gleich $|E(G^*)| - |V(G^*)| + 2 = |E(G)| - (|E(G)| - |V(G)| + 2) + 2 = |V(G)|$ ist. Somit enthält jedes Gebiet von G^* genau einen Knoten von G . Daraus folgt, dass das planare Dual von G^* zu G isomorph ist. \square

Die Bedingung, dass G hier zusammenhängend sein muss, ist wesentlich: Beachte, dass G^* immer zusammenhängend ist, auch wenn G unzusammenhängend ist.

Satz 2.43. *Sei G ein zusammenhängender planarer ungerichteter Graph mit einer beliebigen Einbettung. Die Kantenmenge eines Kreises in G entspricht einem inklusionsminimalen Schnitt in G^* , und ein inklusionsminimaler Schnitt in G entspricht der Kantenmenge eines Kreises in G^* .*

Beweis: Sei $\Phi = (\psi, (J_e)_{e \in E(G)})$ eine feste planare Einbettung von G . Sei ferner C ein Kreis in G . Nach Satz 2.30 spaltet sich $\mathbb{R}^2 \setminus \bigcup_{e \in E(C)} J_e$ in genau zwei zusammenhängende Regionen auf. Sei A bzw. B die Menge der Gebiete von Φ in der inneren bzw. äußereren Region. Dann haben wir $V(G^*) = A \cup B$ und $E_{G^*}(A, B) = \{e^* : e \in E(C)\}$. Da sowohl A als auch B eine zusammenhängende Knotenmenge in G^* bildet, liegt in der Tat ein inklusionsminimaler Schnitt vor.

Zur Umkehrung, sei $\delta_G(A)$ ein inklusionsminimaler Schnitt in G . Sei ferner $\Phi^* = (\psi^*, (J_e)_{e \in E(G^*)})$ eine Standardeinbettung von G^* . Sei $a \in A$ und $b \in V(G) \setminus A$. Beachte, dass es keinen $\psi(a)$ und $\psi(b)$ verbindenden polygonalen Streckenzug in

$$R := \mathbb{R}^2 \setminus \left(\{\psi^*(v) : v \in V(G^*)\} \cup \bigcup_{e \in \delta_G(A)} J_{e^*} \right)$$

gibt: Die Folge der von einem solchen polygonalen Streckenzug durchquerten Gebiete von G^* würde eine Kantenfolge von a nach b in G definieren, die keine Kante von $\delta_G(A)$ enthält.

Also besteht R aus mindestens zwei zusammenhängenden Regionen. Offensichtlich enthalten dann die Ränder dieser beiden Regionen je einen Kreis. Folglich enthält $F := \{e^* : e \in \delta_G(A)\}$ die Kantenmenge eines Kreises C in G^* . Wir haben nun $\{e^* : e \in E(C)\} \subseteq \{e^* : e \in F\} = \delta_G(A)$ und mit dem ersten Teil folgt, dass $\{e^* : e \in E(C)\}$ ein inklusionsminimaler Schnitt in $(G^*)^* = G$ ist (siehe Proposition 2.42). Daraus folgern wir, dass $\{e^* : e \in E(C)\} = \delta_G(A)$. \square

Insbesondere ist e^* genau dann eine Schleife, wenn e eine Brücke ist, und umgekehrt: e^* ist genau dann eine Brücke, wenn e eine Schleife ist. Für Digraphen ergibt der obige Beweis das

Korollar 2.44. *Sei G ein zusammenhängender planarer Digraph mit einer festen planaren Einbettung. Dann entspricht die Kantenmenge eines Kreises in G einem inklusionsminimalen gerichteten Schnitt in G^* und umgekehrt.* \square

Eine weitere interessante Folge aus Satz 2.43 ist:

Korollar 2.45. *Sei G ein zusammenhängender ungerichteter Graph mit einer beliebigen planaren Einbettung. Dann ist G bipartit genau dann, wenn G^* eulersch ist und umgekehrt: G ist eulersch genau dann, wenn G^* bipartit ist.*

Beweis: Beachte, dass ein zusammenhängender Graph genau dann eulersch ist, wenn jeder inklusionsminimale Schnitt gerade Kardinalität hat. Nach Satz 2.43 ist G bipartit, wenn G^* eulersch ist, und G ist eulersch, wenn G^* bipartit ist. Nach Proposition 2.42 gilt auch die Umkehrung. \square

Ein **abstraktes Dual** von G ist ein Graph G' , für den es eine Bijektion $\chi : E(G) \rightarrow E(G')$ gibt mit der Eigenschaft: F ist die Kantenmenge eines Kreises genau dann, wenn $\chi(F)$ ein inklusionsminimaler Schnitt in G' ist, und umgekehrt. Nach Satz 2.43 ist jedes planare Dual auch ein abstraktes Dual. Die Umkehrung gilt jedoch nicht. Whitney [1933] hat aber bewiesen, dass ein Graph genau dann ein abstraktes Dual hat, wenn er planar ist (siehe Aufgabe 40). Wir werden uns später wieder bei der Betrachtung von Matroiden in Abschnitt 13.3 mit dieser Dualitätsrelation befassen.

Aufgaben

1. Sei G ein einfacher ungerichteter Graph mit n Knoten, der zu seinem Komplementgraphen isomorph ist. Man zeige, dass $n \bmod 4 \in \{0, 1\}$ ist.
2. Sei G ein ungerichteter Graph. Für $X \subseteq V(G)$ sei $f(X) := |E(G[X])|$. Man zeige, dass $f : 2^{V(G)} \rightarrow \mathbb{Z}$ supermodular ist.
3. Man beweise, dass jeder einfache ungerichtete Graph G mit $|\delta(v)| \geq \frac{1}{2}|V(G)|$ für alle $v \in V(G)$ hamiltonsch ist.

Hinweis: Man betrachte einen längsten Weg in G und die Nachbarn seiner Endknoten.

(Dirac [1952])

4. Man beweise, dass ein einfacher ungerichteter Graph G mit $|E(G)| > \binom{|V(G)|-1}{2}$ zusammenhängend ist.
5. Sei G ein einfacher ungerichteter Graph. Man zeige, dass G oder sein Komplementgraph zusammenhängend ist.
6. Man beweise: Jeder einfache ungerichtete Graph mit mindestens zwei Knoten enthält zwei Knoten desselben Grades. Man beweise ferner, dass jeder Baum (außer dem einknotigen Baum) mindestens zwei Blätter enthält.
7. Man zeige, dass ein Baum T mit k Blättern höchstens $k-2$ Knoten mindestens dritten Grades enthält.
8. Man beweise, dass jeder Baum T einen Knoten v mit der Eigenschaft hat, dass keine Zusammenhangskomponente von $T - v$ mehr als $\frac{|V(T)|}{2}$ Knoten enthält. Kann man einen solchen Knoten in linearer Laufzeit finden?
9. Sei G ein zusammenhängender ungerichteter Graph und $(V(G), F)$ ein Wald in G . Man beweise, dass es einen aufspannenden Baum $(V(G), T)$ mit $F \subseteq T \subseteq E(G)$ gibt.
10. Seien (V, F_1) und (V, F_2) zwei Wälder mit $|F_1| < |F_2|$. Man beweise, dass es eine Kante $e \in F_2 \setminus F_1$ gibt, so dass $(V, F_1 \cup \{e\})$ ein Wald ist.
11. Seien (V, F_1) und (V, F_2) zwei Branchings mit $2|F_1| < |F_2|$. Man beweise, dass es eine Kante $e \in F_2 \setminus F_1$ gibt, so dass $(V, F_1 \cup \{e\})$ ein Branching ist.
12. Man beweise, dass ein Schnitt in einem ungerichteten Graphen die disjunkte Vereinigung inklusionsminimaler Schnitte ist.
13. Sei G ein ungerichteter Graph, C ein Kreis und D ein Schnitt. Man zeige, dass $|E(C) \cap D|$ eine gerade Zahl ist.
14. Man zeige, dass ein ungerichteter Graph einen Schnitt hat, der mindestens die Hälfte aller Kanten enthält.
15. Sei (U, \mathcal{F}) ein kreuzungsfreies Mengensystem mit $|U| \geq 2$. Man beweise, dass \mathcal{F} höchstens $4|U| - 4$ verschiedene Elemente enthält.
16. Sei G ein zusammenhängender ungerichteter Graph. Man zeige, dass es eine Orientierung G' von G und eine aufspannende Arboreszenz T von G' gibt, so dass die Menge der Fundamentalkreise bezüglich T genau die Menge der gerichteten Kreise in G' ist.
Hinweis: Man betrachte einen DFS-Baum.
(Camion [1968])
17. Man beschreibe einen Algorithmus mit linearer Laufzeit für das Problem: Ist eine Adjazenzliste für einen Graphen G gegeben, so berechne man eine Adjazenzliste für den maximalen einfachen Teilgraphen von G . Dabei nehme man nicht an, dass parallele Kanten im Input hintereinander erscheinen.
18. Man zeige für einen (gerichteten oder ungerichteten) Graphen, dass es einen Algorithmus mit linearer Laufzeit gibt, um einen Kreis zu finden oder zu entscheiden, dass es keinen gibt.
19. Man beschreibe einen einfachen linearen Algorithmus zur Bildung einer topologischen Ordnung in einem gegebenen azyklischen Digraphen. (Dazu benutze man nicht den ALGORITHMUS FÜR STARKE ZUSAMMENHANGSKOMPONENTEN.)

20. Sei G ein zusammenhängender ungerichteter Graph, $s \in V(G)$ und T ein aus der DFS-Anwendung auf (G, s) resultierender DFS-Baum. Der Knoten s heißt die Wurzel von T . Es ist x ein Vorfahre von y in T , falls x auf dem (eindeutig definierten) s - y -Weg in T liegt. Es ist x der Vorgänger von y , falls die Kante $\{x, y\}$ auf dem s - y -Weg in T liegt. Es ist y ein Kind (bzw. Nachfolger) von x , falls x der Vorgänger (bzw. ein Vorfahre) von y ist. Man beachte, dass nach dieser Definition jeder Knoten ein Vorfahre und auch ein Nachfolger von sich selbst ist. Jeder Knoten mit Ausnahme von s hat genau einen Vorgänger. Man beweise:

- (a) Für jede Kante $\{v, w\} \in E(G)$ ist v ein Vorfahre oder ein Nachfolger von w in T .
- (b) Ein Knoten v ist ein Artikulationsknoten von G genau dann, wenn
 - entweder $v = s$ und $|\delta_T(v)| > 1$,
 - oder $v \neq s$ und es gibt ein Kind w von v , so dass keine Kante in G einen echten Vorfahren von v (d. h. außer v) mit einem Nachfolger von w verbindet.

* 21. Anhand von Aufgabe 20 konstruiere man einen Algorithmus mit linearer Laufzeit zur Bestimmung der Blöcke eines ungerichteten Graphen. Es wird sich als nützlich herausstellen, die Zahlen $\alpha(x) :=$

$$\min\{f(w) : w = x \text{ oder } \{w, y\} \in E(G) \setminus T \text{ für einen Nachfolger } y \text{ von } x\}$$

während der DFS-Anwendung rekursiv zu bestimmen. Hier ist (R, T) der DFS-Baum (mit Wurzel s) und die f -Werte ergeben die Reihenfolge des Hinzufügens der Knoten zu R (siehe den GRAPH-SCANNING-ALGORITHMUS). Gilt $\alpha(x) \geq f(w)$ für einen Knoten $x \in R \setminus \{s\}$, wobei w der Vorgänger von x ist, so ist w entweder die Wurzel oder ein Artikulationsknoten.

22. Man beweise:

- (a) Ein ungerichteter Graph ist genau dann 2-fach kantenzusammenhängend, wenn er mindestens zwei Knoten und eine Ohrenzerlegung hat.
- (b) Ein Digraph ist genau dann stark zusammenhängend, wenn er eine Ohrenzerlegung hat.
- (c) Die Kanten eines ungerichteten Graphen G mit mindestens zwei Knoten können genau dann so orientiert werden, dass der resultierende Digraph stark zusammenhängend ist, wenn G 2-fach kantenzusammenhängend ist. (Robbins [1939])

23. Ein Turnier ist ein Digraph, dessen zugrunde liegender ungerichteter Graph ein (einfacher) vollständiger Graph ist. Man beweise, dass jedes Turnier einen hamiltonschen Weg enthält (Rédei [1934]). Man beweise ferner, dass jedes stark zusammenhängende Turnier hamiltonsch ist (Camion [1959]).

24. Man zeige, dass es für einen ungerichteten Graphen G eine Orientierung G' gibt mit der Eigenschaft, dass $||\delta_{G'}^+(v)| - |\delta_{G'}^-(v)|| \leq 1$ für alle $v \in V(G')$.

25. Man beweise: Ist ein zusammenhängender ungerichteter einfacher Graph eulersch, so ist sein Kantengraph hamiltonsch. Wie steht es mit der Umkehrung?

26. Man beweise erstens, dass ein zusammenhängender bipartiter Graph eine eindeutig bestimmte Bipartition hat. Man beweise zweitens, dass ein nicht bipartiter ungerichteter Graph einen ungeraden Kreis als induzierten Teilgraphen enthält. Man beweise drittens, dass ein ungerichteter Graph G genau dann bipartit ist, wenn es eine aus Schnitten bestehende Partition von $E(G)$ gibt.
27. Man beweise, dass ein stark zusammenhängender Digraph mit nicht bipartitem zugrunde liegendem ungerichtetem Graphen einen (gerichteten) Kreis ungerader Länge enthält.
- * 28. Sei G ein ungerichteter Graph. Eine **Baumzerlegung** von G ist ein Paar (T, φ) , wobei T ein Baum ist und $\varphi : V(T) \rightarrow 2^{V(G)}$ die folgenden Bedingungen erfüllt:
- für jedes $e \in E(G)$ gibt es ein $t \in V(T)$ mit $e \subseteq \varphi(t)$;
 - für jedes $v \in V(G)$ ist die Menge $\{t \in V(T) : v \in \varphi(t)\}$ zusammenhängend in T .

Man definiert die Weite von (T, φ) als $\max_{t \in V(T)} |\varphi(t)| - 1$. Die **Baumweite** eines Graphen G ist die minimale Weite einer Baumzerlegung von G . Dieser Begriff stammt von Robertson und Seymour [1986].

Man zeige, dass die einfachen Graphen mit Baumweite höchstens 1 genau die Wälder sind. Ferner beweise man, dass die folgenden drei Aussagen für einen ungerichteten Graphen G äquivalent sind:

- (a) G hat Baumweite höchstens 2;
- (b) G enthält keinen K_4 -Minor;
- (c) G kann aus einem leeren Graphen durch schrittweises Hinzufügen von Brücken und Verdoppelung und Unterteilung von Kanten erstellt werden. (Verdoppelung einer Kante $e = \{v, w\} \in E(G)$ bedeutet das Hinzufügen einer weiteren Kante mit den Endknoten v und w ; Unterteilung einer Kante $e = \{v, w\} \in E(G)$ bedeutet das Hinzufügen eines Knotens x und das Ersetzen von e durch zwei Kanten $\{v, x\}, \{x, w\}$.)

Bemerkung: In Anlehnung an die in (c) beschriebene Konstruktion heißen solche Graphen seriengleich.

29. Man zeige: Hat ein Graph G eine planare Einbettung für welche die eingebetteten Kanten beliebige Jordankurven sind, so hat er auch eine planare Einbettung bestehend allein aus polygonalen Streckenzügen.
30. Sei G ein 2-fach zusammenhängender Graph mit einer planaren Einbettung. Man zeige, dass die Menge der als Ränder der beschränkten Gebiete auftretenden Kreise eine Kreisbasis von G bildet.
31. Kann man die Eulersche Formel (Satz 2.32) auf unzusammenhängende Graphen erweitern?
32. Man zeige, dass es genau fünf platonische Graphen gibt (entsprechend den platonischen Körpern; siehe Aufgabe 11, Kapitel 4), d. h. 3-fach zusammenhängende planare reguläre Graphen, dessen Gebiete alle mit der gleichen Anzahl von Kanten umrandet werden.
- Hinweis:* Man benutze die Eulersche Formel (Satz 2.32).
33. Aus dem Beweis des Satzes von Kuratowski (Satz 2.39) leite man ab:

- (a) Jeder 3-fach zusammenhängende einfache planare Graph hat eine planare Einbettung mit der Eigenschaft, dass jede Kante geradlinig und jedes Gebiet, bis auf das äußere, konvex eingebettet wird.
- (b) Es gibt einen polynomiellen Algorithmus zur Prüfung, ob ein gegebener Graph planar ist.
- * 34. Sei G ein Graph und $e = \{v, w\} \in E(G)$ eine Kante. Gilt für einen Graphen H , dass $V(H) = V(G) \cup \{x\}$ und $E(H) = (E(G) \setminus \{e\}) \cup \{\{v, x\}, \{x, w\}\}$, so sagt man, dass H aus G durch die Unterteilung von e hervorgeht. Einen Graphen, welcher aus G durch die schrittweise Unterteilung von Kanten hervorgeht, nennt man eine **Unterteilung** von G .
- (a) Trivialerweise folgt: Enthält H eine Unterteilung von G , so ist G ein Minor von H . Man zeige, dass die Umkehrung nicht gilt.
- (b) Man beweise, dass ein Graph mit einem $K_{3,3}$ - oder K_5 -Minor auch eine Unterteilung von $K_{3,3}$ oder K_5 enthält.
- Hinweis:* Man untersuche, was bei der Kontraktion einer Kante geschieht.
- (c) Man folgere daraus, dass ein Graph genau dann planar ist, wenn kein Teilgraph eine Unterteilung von $K_{3,3}$ oder K_5 ist.
(Kuratowski [1930])
35. Man beweise, dass aus jeder der beiden folgenden Aussagen die andere folgt:
- (a) Für jede unendliche Folge G_1, G_2, \dots von Graphen gibt es zwei Indizes $i < j$, so dass G_i ein Minor von G_j ist.
- (b) Sei \mathcal{G} eine Klasse von Graphen mit der Eigenschaft: Für jeden Graphen $G \in \mathcal{G}$ und jeden Minor H von G ist $H \in \mathcal{G}$ (d.h. Mitgliedschaft in \mathcal{G} ist eine erbliche Grapheneigenschaft). Dann gibt es eine endliche Menge \mathcal{X} von Graphen, so dass \mathcal{G} aus allen kein Element von \mathcal{X} als Minor enthaltenden Graphen besteht.
- Bemerkung:* Diese Aussagen sind von Robertson und Seymour [2004] bewiesen worden. Sie bilden eines der Hauptresultate in der Reihe von Arbeiten dieser Autoren über Minoren von Graphen. Satz 2.39 und Aufgabe 28 liefern Beispiele von Charakterisierungen mittels verbotener Minoren analog (b).
36. Sei G ein planarer Graph mit einer Einbettung Φ und C ein Kreis in G , der irgendein Gebiet von Φ berandet. Man beweise, dass es eine Einbettung Φ' von G gibt, so dass C das äußere Gebiet berandet.
37. (a) Sei G ein unzusammenhängender Graph mit einer beliebigen planaren Einbettung und G^* das planare Dual mit einer Standardeinbettung. Man beweise, dass $(G^*)^*$ aus G gewonnen werden kann durch die schrittweise Anwendung der folgenden Operation bis der resultierende Graph zusammenhängend ist: Man wähle zwei Knoten x und y , die zwei verschiedenen Zusammenhangskomponenten angehören, aber beide auf dem Rand des selben Gebietes liegen, und kontrahiere $\{x, y\}$.
- (b) Man verallgemeinere Korollar 2.45 auf beliebige planare Graphen.
Hinweis: Man benutze (a) und Satz 2.26.
38. Sei G ein zusammenhängender Digraph mit einer festen planaren Einbettung und G^* das planare Dual mit einer Standardeinbettung. Wie sind G und $(G^*)^*$ miteinander verwandt?

39. Man beweise: Ist ein planarer Digraph azyklisch bzw. stark zusammenhängend, so ist sein planares Dual stark zusammenhängend bzw. azyklisch. Wie steht es mit der Umkehrung?
40. (a) Man zeige: Hat G ein abstraktes Dual und ist H ein Minor von G , so hat H auch ein abstraktes Dual.
- * (b) Man zeige, dass weder K_5 noch $K_{3,3}$ ein abstraktes Dual hat.
- (c) Man folgere hieraus, dass ein Graph genau dann planar ist, wenn er ein abstraktes Dual hat.
 (Whitney [1933])

Literatur

Allgemeine Literatur:

- Berge, C. [1985]: Graphs. 2. Aufl. Elsevier, Amsterdam 1985
 Bollobás, B. [1998]: Modern Graph Theory. Springer, New York 1998
 Bondy, J.A. [1995]: Basic graph theory: paths and circuits. In: Handbook of Combinatorics; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, Hrsg.), Elsevier, Amsterdam 1995
 Bondy, J.A., und Murty, U.S.R. [2008]: Graph Theory. Springer, New York 2008
 Diestel, R. [2017]: Graphentheorie. 5. Aufl. Springer, Berlin 2017
 Wilson, R.J. [2010]: Introduction to Graph Theory. 5. Aufl. Addison-Wesley, Reading 2010

Zitierte Literatur:

- Aoshima, K., und Iri, M. [1977]: Comments on F. Hadlock's paper: finding a maximum cut of a planar graph in polynomial time. SIAM Journal on Computing 6 (1977), 86–87
 Camion, P. [1959]: Chemins et circuits hamiltoniens des graphes complets. Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris) 249 (1959), 2151–2152
 Camion, P. [1968]: Modulaires unimodulaires. Journal of Combinatorial Theory A 4 (1968), 301–362
 Dirac, G.A. [1952]: Some theorems on abstract graphs. Proceedings of the London Mathematical Society 2 (1952), 69–81
 Edmonds, J., und Giles, R. [1977]: A min-max relation for submodular functions on graphs. In: Studies in Integer Programming; Annals of Discrete Mathematics 1 (P.L. Hammer, E.L. Johnson, B.H. Korte, G.L. Nemhauser, Hrsg.), North-Holland, Amsterdam 1977, pp. 185–204
 Euler, L. [1736]: Solutio problematis ad geometriam situs pertinentis. Commentarii Academiae Petropolitanae 8 (1736), 128–140
 Euler, L. [1758]: Demonstratio nonnullarum insignium proprietatum quibus solida hedris planis inclusa sunt praedita. Novi Commentarii Academiae Petropolitanae 4 (1758), 140–160
 Hierholzer, C. [1873]: Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. Mathematische Annalen 6 (1873), 30–32
 Hopcroft, J.E., und Tarjan, R.E. [1974]: Efficient planarity testing. Journal of the ACM 21 (1974), 549–568
 Kahn, A.B. [1962]: Topological sorting of large networks. Communications of the ACM 5 (1962), 558–562

- Karzanov, A.V. [1970]: An efficient algorithm for finding all the bi-components of a graph. In: Trudy 3-ї Zimneї Shkoly po Matematicheskому Programmirovaniyu i Smezhnym Voprosam (Drogobych, 1970), Nr. 2, Moskau Institute for Construction Engineering (MISI) Press, Moskau, 1970, pp. 343–347 [auf Russisch]
- Knuth, D.E. [1968]: The Art of Computer Programming; Vol. 1. Fundamental Algorithms. Addison-Wesley, Reading 1968 (3. Aufl. 1997)
- König, D. [1916]: Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Mathematische Annalen* 77 (1916), 453–465
- König, D. [1936]: Theorie der endlichen und unendlichen Graphen. Teubner, Leipzig 1936; Nachdruck: Chelsea Publishing Co., New York 1950
- Kuratowski, K. [1930]: Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae* 15 (1930), 271–283
- Legendre, A.M. [1794]: Éléments de Géométrie. Firmin Didot, Paris 1794
- Minty, G.J. [1960]: Monotone networks. *Proceedings of the Royal Society of London A* 257 (1960), 194–212
- Moore, E.F. [1959]: The shortest path through a maze. *Proceedings of the International Symposium on the Theory of Switching; Part II*. Harvard University Press 1959, pp. 285–292
- Rédei, L. [1934]: Ein kombinatorischer Satz. *Acta Litt. Szeged* 7 (1934), 39–43
- Robbins, H.E. [1939]: A theorem on graphs with an application to a problem of traffic control. *American Mathematical Monthly* 46 (1939), 281–283
- Robertson, N., und Seymour, P.D. [1986]: Graph minors II: algorithmic aspects of tree-width. *Journal of Algorithms* 7 (1986), 309–322
- Robertson, N., und Seymour, P.D. [2004]: Graph minors XX: Wagner's conjecture. *Journal of Combinatorial Theory B* 92 (2004), 325–357
- Tarjan, R.E. [1972]: Depth first search and linear graph algorithms. *SIAM Journal on Computing* 1 (1972), 146–160
- Thomassen, C. [1980]: Planarity and duality of finite and infinite graphs. *Journal of Combinatorial Theory B* 29 (1980), 244–271
- Thomassen, C. [1981]: Kuratowski's theorem. *Journal of Graph Theory* 5 (1981), 225–241
- Tutte, W.T. [1961]: A theory of 3-connected graphs. *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen A* 64 (1961), 441–455
- Wagner, K. [1937]: Über eine Eigenschaft der ebenen Komplexe. *Mathematische Annalen* 114 (1937), 570–590
- Whitney, H. [1932]: Non-separable and planar graphs. *Transactions of the American Mathematical Society* 34 (1932), 339–362
- Whitney, H. [1933]: Planar graphs. *Fundamenta Mathematicae* 21 (1933), 73–84



3 Lineare Optimierung

In diesem Kapitel werden wir die wichtigsten Definitionen und Resultate der linearen Optimierung zusammenstellen. Es ist zwar in sich vollständig, kann jedoch nicht als eine umfassende Einführung in die lineare Optimierung betrachtet werden. Dem mit der linearen Optimierung nicht vertrauten Leser seien die am Ende des Kapitels angegebenen Texte empfohlen.

Das allgemeine lineare Optimierungsproblem lautet:

LINEARE OPTIMIERUNG

Instanz: Eine Matrix $A \in \mathbb{R}^{m \times n}$ und Spaltenvektoren $b \in \mathbb{R}^m, c \in \mathbb{R}^n$.

Aufgabe: Bestimme einen Spaltenvektor $x \in \mathbb{R}^n$ mit $Ax \leq b$ und $c^\top x$ maximal;
entscheide, dass $\{x \in \mathbb{R}^n : Ax \leq b\}$ leer ist;
oder entscheide, dass es für alle $\alpha \in \mathbb{R}$ ein $x \in \mathbb{R}^n$ mit $Ax \leq b$ und $c^\top x > \alpha$ gibt.

Hier bedeutet $c^\top x$ das Skalarprodukt von Vektoren. Die Notation $x \leq y$ für Vektoren x und y gleicher Dimension bedeutet, dass die Ungleichung für jede Komponente gilt. Werden keine Dimensionen angegeben, so wird davon ausgegangen, dass alle Matrizen und Vektoren kompatibel sind. Auch werden wir bei Spaltenvektoren oft das Transpositionszeichen weglassen und z. B. cx für das Skalarprodukt schreiben. Mit 0 bezeichnen wir nicht nur die Zahl Null, sondern auch Vektoren und Matrizen, in denen sämtliche Elemente gleich Null sind. (Vektorlänge und Matrixgröße werden stets aus dem Kontext ersichtlich sein).

Eine Instanz dieses Problems nennt man ein **lineares Programm (LP)**. Oft schreibt man ein lineares Programm auch in der Form $\max\{cx : Ax \leq b\}$. Eine **zulässige Lösung** eines LP $\max\{cx : Ax \leq b\}$ ist ein Vektor x mit $Ax \leq b$. Eine zulässige Lösung, für welche das Maximum angenommen wird, heißt **optimale Lösung**.

Wie bereits aus der Problemformulierung hervorgeht, gibt es genau zwei Fälle, in denen ein LP keine Lösungen hat: Das Problem kann **unzulässig** (d. h. $P := \{x \in \mathbb{R}^n : Ax \leq b\} = \emptyset$) oder **unbeschränkt** (d. h. für jedes $\alpha \in \mathbb{R}$ gibt es ein $x \in P$ mit $cx > \alpha$) sein. Ist ein LP weder unzulässig noch unbeschränkt, so besitzt es eine optimale Lösung:

Proposition 3.1. Sei $P = \{x \in \mathbb{R}^n : Ax \leq b\} \neq \emptyset$ und $c \in \mathbb{R}^n$ mit $\delta := \sup\{c^\top x : x \in P\} < \infty$. Dann gibt es einen Vektor $z \in P$ mit $c^\top z = \delta$.

Beweis: Sei U eine Matrix, deren Spalten eine orthonormale Basis des Kerns von A bilden, d.h. $U^\top U = I$, $AU = 0$ und $\text{rank}(A') = n$, wobei $A' := \begin{pmatrix} A \\ U^\top \end{pmatrix}$. Sei $b' := \begin{pmatrix} b \\ 0 \end{pmatrix}$.

Wir werden zeigen, dass es für jedes $y \in P$ ein Teilsystem $A''x \leq b''$ von $A'x \leq b'$ gibt, mit A'' nichtsingulär, $y' := (A'')^{-1}b'' \in P$ und $c^\top y' \geq c^\top y$. Da es nur eine endliche Anzahl solcher Teilsysteme gibt, nimmt eines dieser y' den maximalen Zielfunktionswert $c^\top y' = \delta$ an, womit die Aussage folgt.

Sei also $y \in P$ und sei $k(y)$ der Rang von A'' für das maximale Teilsystem $A''x \leq b''$ von $A'x \leq b'$ mit $A''y = b''$. Ist $k(y) < n$, so zeigen wir, dass wir ein $y' \in P$ mit $c^\top y' \geq c^\top y$ und $k(y') > k(y)$ bestimmen können. Nach höchstens n Schritten haben wir dann einen Vektor y' mit $k(y') = n$, wie erwünscht.

Zunächst betrachten wir den Fall $U^\top y \neq 0$ und setzen $y' := y - UU^\top y$. Da $y + \lambda UU^\top c \in P$ für alle $\lambda \in \mathbb{R}$, haben wir $\sup\{c^\top(y + \lambda UU^\top c) : \lambda \in \mathbb{R}\} \leq \delta < \infty$ und somit $c^\top U = 0$ und $c^\top y' = c^\top y$. Ferner gilt $Ay' = Ay - AUU^\top y = Ay$ und $U^\top y' = U^\top y - U^\top UU^\top y = 0$.

Ist andererseits $U^\top y = 0$, so sei $v \neq 0$ mit $A''v = 0$. Sei ferner $a_i x \leq \beta_i$ die i -te Zeile von $Ax \leq b$. Wir setzen $\mu := \min \left\{ \frac{\beta_i - a_i y}{a_i v} : a_i v > 0 \right\}$ und $\kappa := \max \left\{ \frac{\beta_i - a_i y}{a_i v} : a_i v < 0 \right\}$, wobei $\min \emptyset = \infty$ und $\max \emptyset = -\infty$. Es folgt, dass $\kappa \leq 0 \leq \mu$ und dass mindestens eine der beiden Größen κ und μ endlich ist (da $A'v \neq 0$ aber $U^\top v = 0$).

Für $\lambda \in \mathbb{R}$ mit $\kappa \leq \lambda \leq \mu$ haben wir $A''(y + \lambda v) = A''y + \lambda A''v = A''y = b''$ und $A(y + \lambda v) = Ay + \lambda Av \leq b$, d.h. $y + \lambda v \in P$. Da $\sup\{c^\top x : x \in P\} < \infty$, folgt $\mu < \infty$, falls $c^\top v > 0$, und $\kappa > -\infty$, falls $c^\top v < 0$.

Ferner haben wir: Ist $c^\top v \geq 0$ und $\mu < \infty$, so folgt $a_i(y + \mu v) = \beta_i$ für irgendein i . Analog haben wir: Ist $c^\top v \leq 0$ und $\kappa > -\infty$, so folgt $a_i(y + \kappa v) = \beta_i$ für irgendein i .

Also haben wir in allen Fällen einen Vektor $y' \in P$ mit $c^\top y' \geq c^\top y$ und $k(y') \geq k(y) + 1$ bestimmt. \square

Dies erlaubt uns, $\max\{c^\top x : Ax \leq b\}$ anstatt $\sup\{c^\top x : Ax \leq b\}$ zu schreiben.

Viele kombinatorische Optimierungsprobleme können als LP formuliert werden.

Um dies zu erreichen, kodieren wir die zulässigen Lösungen als Vektoren im \mathbb{R}^n für ein passendes n . Im Abschnitt 3.5 zeigen wir, dass man eine lineare Zielfunktion über einer endlichen Menge S von Vektoren optimieren kann, indem man ein LP löst. Obwohl die Menge der zulässigen Lösungen dieses LP nicht nur die Vektoren in S enthält, sondern auch sämtliche Konvexitätskombinationen dieser Vektoren, kann man beweisen, dass es unter den optimalen Lösungen immer ein Element aus S gibt.

Abschnitt 3.1 beinhaltet eine Zusammenstellung einiger grundlegender Begriffe und Resultate über Polyeder, d.h. der Mengen $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ der zulässigen Lösungen von LPs. In den Abschnitten 3.2 und 3.3 betrachten wir den SIMPLEXALGORITHMUS, den wir dann auch zur Herleitung des Dualitätssatzes und damit zusammenhängender Resultate benutzen werden (Abschnitt 3.4). Der Begriff der LP-Dualität ist außerordentlich wichtig und erscheint explizit oder

implizit in fast allen Teilen der kombinatorischen Optimierung; wir werden uns oft auf die Ergebnisse der Abschnitte 3.4 und 3.5 beziehen.

3.1 Polyeder

In der linearen Optimierung betrachtet man die Maximierung oder Minimierung einer linearen Zielfunktion mit endlich vielen Variablen, bezüglich endlich vieler linearer Ungleichungen als Nebenbedingungen. Also besteht die Menge der zulässigen Lösungen aus dem Schnitt endlich vieler Halbräume. Eine solche Menge heißt Polyeder:

Definition 3.2. Ein **Polyeder** im \mathbb{R}^n ist eine Menge vom Typ $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, wobei $A \in \mathbb{R}^{m \times n}$ eine Matrix und $b \in \mathbb{R}^m$ ein Vektor ist. Sind A und b rational, so nennt man P ein **rationales Polyeder**. Ein beschränktes Polyeder wird auch **Polytop** genannt.

Für den Rang einer Matrix A schreiben wir $\text{rank}(A)$. Die **Dimension** $\dim X$ einer nichtleeren Menge $X \subseteq \mathbb{R}^n$ ist die Zahl

$$n - \max\{\text{rank}(A) : A \text{ ist eine } n \times n\text{-Matrix mit } Ax = Ay \text{ für alle } x, y \in X\}.$$

Ein Polyeder $P \subseteq \mathbb{R}^n$ heißt **volldimensional**, falls $\dim P = n$.

Eine äquivalente Definition ist, dass ein Polyeder genau dann volldimensional ist, wenn es einen inneren Punkt hat. In diesem Kapitel macht es größtenteils keinen Unterschied, ob wir im rationalen oder im reellen Raum arbeiten. Wir benötigen die folgende Standardterminologie:

Definition 3.3. Sei $P := \{x : Ax \leq b\}$ ein nichtleeres Polyeder und $c \neq 0$ ein Vektor, für den $\delta := \max\{cx : x \in P\}$ endlich ist. Dann nennt man $\{x : cx = \delta\}$ eine **stützende Hyperebene** von P . Eine **Seitenfläche** von P ist entweder ganz P oder der Schnitt von P mit einer stützenden Hyperebene von P . Ein Punkt x , für den $\{x\}$ eine Seitenfläche ist, heißt **Ecke** von P oder auch **Basislösung** des Systems $Ax \leq b$.

Proposition 3.4. Sei $P = \{x : Ax \leq b\}$ ein Polyeder und $F \subseteq P$. Dann sind die folgenden drei Aussagen äquivalent:

- (a) F ist eine Seitenfläche von P .
- (b) Es gibt einen Vektor c , für welchen $\delta := \max\{cx : x \in P\}$ endlich ist und $F = \{x \in P : cx = \delta\}$.
- (c) $F = \{x \in P : A'x = b'\} \neq \emptyset$ für ein geeignetes Teilsystem $A'x \leq b'$ von $Ax \leq b$.

Beweis: (a) und (b) sind offensichtlich äquivalent.

(c) \Rightarrow (b): Sei $F = \{x \in P : A'x = b'\}$ nicht leer, c die Summe der Zeilen von A' und δ die Summe der Komponenten von b' . Dann ist offensichtlich $cx \leq \delta$ für alle $x \in P$ und $F = \{x \in P : cx = \delta\}$.

(b) \Rightarrow (c): Sei c ein Vektor, $\delta := \max\{cx : x \in P\}$ endlich und $F = \{x \in P : cx = \delta\}$. Ferner sei $A'x \leq b'$ das maximale Teilsystem von $Ax \leq b$ mit $A'x = b'$ für alle $x \in F$ und $A''x \leq b''$ der Rest des Systems $Ax \leq b$.

Als erstes bemerken wir, dass es für jede Ungleichung $a_i''x \leq \beta_i''$ von $A''x \leq b''$ ($i = 1, \dots, k$) einen Punkt $x_i \in F$ gibt, für den $a_i''x_i < \beta_i''$. Sei $x^* := \frac{1}{k} \sum_{i=1}^k x_i$ der Schwerpunkt dieser Punkte (ist $k = 0$, so wählen wir einen beliebigen Punkt $x^* \in F$); dann folgt $x^* \in F$ und $a_i''x^* < \beta_i''$ für alle i .

Wir müssen beweisen, dass $A'y = b'$ für kein $y \in P \setminus F$ gilt. Es sei also $y \in P \setminus F$. Dann folgt $cy < \delta$. Nun nehmen wir $z := x^* + \epsilon(x^* - y)$ für ein genügend kleines $\epsilon > 0$; insbesondere sei ϵ kleiner als $\frac{\beta_i'' - a_i''x^*}{a_i''(x^* - y)}$ für alle $i \in \{1, \dots, k\}$ mit $a_i''x^* > a_i''y$.

Wir haben $cz > \delta$, somit ist $z \notin P$. Also gibt es eine Ungleichung $ax \leq \beta$ aus $Ax \leq b$ mit $az > \beta$. Damit folgt $ax^* > ay$. Die Ungleichung $ax \leq \beta$ kann nicht im System $A''x \leq b''$ sein, da wir sonst $az = ax^* + \epsilon a(x^* - y) < ax^* + \frac{\beta - ax^*}{a(x^* - y)} a(x^* - y) = \beta$ hätten (nach Wahl von ϵ). Also ist die Ungleichung $ax \leq \beta$ aus $A'x \leq b'$. Aber dann gilt $ay = a(x^* + \frac{1}{\epsilon}(x^* - z)) < \beta$, womit der Beweis abgeschlossen ist. \square

Es folgt das triviale aber wichtige Korollar:

Korollar 3.5. *Sei P ein nichtleeres Polyeder und c ein Vektor. Ist $\max\{cx : x \in P\}$ beschränkt, so bildet die Menge derjenigen Punkte, für die das Maximum angenommen wird, eine Seitenfläche von P .* \square

Die Relation „ist eine Seitenfläche von“ ist transitiv:

Korollar 3.6. *Sei P ein Polyeder und F eine Seitenfläche von P . Dann ist F selbst ein Polyeder. Ferner ist eine Menge $F' \subseteq F$ genau dann eine Seitenfläche von P , wenn sie eine Seitenfläche von F ist.* \square

Die inklusionsmaximalen Seitenflächen außer P selbst sind besonders wichtig:

Definition 3.7. *Sei P ein Polyeder. Eine **Facette** von P ist eine inklusionsmaximale Seitenfläche ungleich P . Eine Ungleichung $cx \leq \delta$ heißt **facettenbestimmend** für P , wenn $cx \leq \delta$ für alle $x \in P$ gilt und $\{x \in P : cx = \delta\}$ eine Facette von P ist.*

Proposition 3.8. *Sei $P \subseteq \{x \in \mathbb{R}^n : Ax = b\}$ ein nichtleeres Polyeder der Dimension $n - \text{rank}(A)$. Sei ferner $A'x \leq b'$ ein minimales Ungleichungssystem mit $P = \{x : Ax = b, A'x \leq b'\}$. Dann ist jede Ungleichung $A'x \leq b'$ facettenbestimmend für P und jede Facette von P wird durch eine Ungleichung aus $A'x \leq b'$ bestimmt.*

Beweis: Ist $P = \{x \in \mathbb{R}^n : Ax = b\}$, so gibt es keine Facetten und die Aussage ist trivial. Also sei $A'x \leq b'$ ein minimales Ungleichungssystem mit $P = \{x : Ax = b, A'x \leq b'\}$, sei $a'x \leq \beta'$ eine dieser Ungleichungen und $A''x \leq b''$ der Rest des Systems $A'x \leq b'$. Sei ferner y ein Vektor mit $Ay = b, A''y \leq b''$ und $a'y > \beta'$

(ein solcher Vektor y existiert, da die Ungleichung $a'x \leq \beta'$ nicht redundant ist) und sei $x \in P$ mit $A'x < b'$ (solch ein Vektor existiert, weil $\dim P = n - \text{rank}(A)$).

Setze $z := x + \frac{\beta' - a'x}{a'y - a'x}(y - x)$. Dann haben wir $a'z = \beta'$, $A''z < b''$, und auch $z \in P$, da $0 < \frac{\beta' - a'x}{a'y - a'x} < 1$. Also ist $F := \{x \in P : a'x = \beta'\} \neq 0$ und $F \neq P$ (da $x \in P \setminus F$). Damit folgt, dass F eine Facette von P ist.

Mit Proposition 3.4 folgt, dass jede Facette durch eine Ungleichung aus $A'x \leq b'$ bestimmt wird. \square

Eine weitere wichtige Klasse von Seitenflächen (außer den Facetten) ist diejenige der minimalen Seitenflächen (d. h. sie enthalten keine anderen Seitenflächen). Hier haben wir:

Proposition 3.9. (Hoffman und Kruskal [1956]) *Sei $P = \{x : Ax \leq b\}$ ein Polyeder. Eine nichtleere Teilmenge $F \subseteq P$ ist genau dann eine minimale Seitenfläche von P , wenn $F = \{x : A'x = b'\}$ für ein geeignetes Teilsystem $A'x \leq b'$ von $Ax \leq b$.*

Beweis: Ist F eine minimale Seitenfläche von P , so folgt nach Proposition 3.4, dass es ein Teilsystem $A'x \leq b'$ von $Ax \leq b$ gibt, mit $F = \{x \in P : A'x = b'\}$. Man wähle $A'x \leq b'$ maximal. Sei $A''x \leq b''$ ein minimales Teilsystem von $Ax \leq b$ mit $F = \{x : A'x = b', A''x \leq b''\}$. Dann behaupten wir, dass $A''x \leq b''$ keine Ungleichungen enthält.

Angenommen, das Gegenteil gelte, dass also $A''x \leq b''$ eine Ungleichung aus $A''x \leq b''$ ist. Da diese für die Bestimmung von F nicht redundant ist, folgt aus Proposition 3.8, dass $F' := \{x : A'x = b', A''x \leq b'', a''x = \beta''\}$ eine Facette von F ist. Nach Korollar 3.6 ist F' auch eine Seitenfläche von P , im Widerspruch zu der Annahme, dass F eine minimale Seitenfläche von P ist.

Nun sei $\emptyset \neq F = \{x : A'x = b'\} \subseteq P$ für ein passendes Teilsystem $A'x \leq b'$ von $Ax \leq b$. Offensichtlich hat F keine Seitenflächen außer sich selbst. Nach Proposition 3.4 ist F eine Seitenfläche von P . Dann folgt nach Korollar 3.6, dass F eine minimale Seitenfläche von P ist. \square

Aus Korollar 3.5 und Proposition 3.9 folgt, dass LINEARE OPTIMIERUNG in endlicher Zeit gelöst werden kann, indem man das lineare Gleichungssystem $A'x = b'$ für jedes Teilsystem $A'x \leq b'$ von $Ax \leq b$ löst. Ein intelligenterer Weg wird durch den SIMPLEXALGORITHMUS ermöglicht. Diesen beschreiben wir im folgenden Abschnitt.

Aus Proposition 3.9 folgt ferner:

Korollar 3.10. *Sei $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ ein Polyeder. Dann haben alle minimalen Seitenflächen von P die Dimension $n - \text{rank}(A)$. Die minimalen Seitenflächen eines Polytops sind seine Ecken.* \square

Aus diesem Grunde heißen Polyeder $\{x \in \mathbb{R}^n : Ax \leq b\}$ mit $\text{rank}(A) = n$ spitz; ihre minimalen Seitenflächen sind Punkte.

Wir schließen diesen Abschnitt mit einigen Bemerkungen über polyedrische Kegel.

Definition 3.11. Ein (konvexer) **Kegel** ist eine Menge $C \subseteq \mathbb{R}^n$ mit der Eigenschaft: Für alle $x, y \in C$ und alle $\lambda, \mu \geq 0$ ist $\lambda x + \mu y \in C$. Man sagt, dass ein **Kegel** C von $x_1, \dots, x_k \in C$ **erzeugt** wird, wenn es für jedes $x \in C$ Zahlen $\lambda_1, \dots, \lambda_k \geq 0$ gibt, mit $x = \sum_{i=1}^k \lambda_i x_i$. Ein Kegel heißt **endlich erzeugt**, wenn er von einer endlichen Menge von Vektoren erzeugt wird. Ein **polyedrischer Kegel** ist ein Polyeder vom Typ $\{x : Ax \leq 0\}$.

Es ist unmittelbar klar, dass polyedrische Kegel tatsächlich Kegel sind. Wir werden nun zeigen, dass polyedrische Kegel endlich erzeugt sind. Identitätsmatrizen werden immer mit I bezeichnet.

Lemma 3.12. (Minkowski [1896]) Sei $C = \{x \in \mathbb{R}^n : Ax \leq 0\}$ ein polyedrischer Kegel. Dann wird C von einer Teilmenge der Lösungsmenge der Systeme $My = b'$ erzeugt, wobei M aus n linear unabhängigen Zeilen der Matrix $\begin{pmatrix} A \\ I \end{pmatrix}$ besteht und $b' = \pm e_j$ für einen Einheitsvektor e_j .

Beweis: Sei A eine $m \times n$ -Matrix und betrachte die Systeme $My = b'$, wobei M aus n linear unabhängigen Zeilen der Matrix $\begin{pmatrix} A \\ I \end{pmatrix}$ besteht und $b' = \pm e_j$ für einen Einheitsvektor e_j . Es seien y_1, \dots, y_t diejenigen Lösungen der obigen Gleichungssysteme, die in C liegen. Wir behaupten nun, dass C von den y_1, \dots, y_t erzeugt wird.

Zunächst nehmen wir an, dass $C = \{x : Ax = 0\}$, d.h. C sei ein linearer Unterraum. Dann können wir $C = \{x : A'x = 0\}$ schreiben, wobei A' aus einer maximalen Menge linear unabhängiger Zeilen von A besteht. Ferner bestehe I' aus einer passenden Menge von Zeilen von I , so dass $\begin{pmatrix} A' \\ I' \end{pmatrix}$ eine nichtsinguläre quadratische Matrix ist. Dann folgt, dass C von den Lösungen des Systems

$$\begin{pmatrix} A' \\ I' \end{pmatrix} x = \begin{pmatrix} 0 \\ b \end{pmatrix} \quad \text{für alle } b = \pm e_j, j = 1, \dots, \dim C$$

erzeugt wird.

Für den allgemeinen Fall verwenden wir Induktion über die Dimension von C . Ist C kein linearer Unterraum, so gibt es einen Vektor $z \in C$ mit $-z \notin C$. Somit gibt es eine Zeile a von A mit $az < 0$.

Sei nun A' eine Matrix bestehend aus einer beliebigen inklusionsmaximalen Zeilenmenge der Matrix A mit den folgenden zwei Eigenschaften: (i) die Zeilen von $\begin{pmatrix} A' \\ a \end{pmatrix}$ sind linear unabhängig, (ii) es gibt einen Vektor $z \in C$ mit $A'z = 0$ und $az < 0$.

Sei y ein Vektor mit $A'y = 0$ und $ay = -1$. Wir zeigen nun, dass $y \in C$.

Es habe z die Eigenschaft (ii), d.h. $z \in C$, $A'z = 0$ und $az < 0$. Sei B die Menge der Zeilen b von A mit $by > 0$. Dann ist jedes $b \in B$ linear unabhängig von a und A' , denn sonst gäbe es einen Vektor c und eine Zahl δ mit $b = cA' + \delta a$. Dann hätten wir aber $0 \geq bz = cA'z + \delta az = \delta az$ und somit $\delta \geq 0$, im Widerspruch zu $0 < by = cA'y + \delta ay = -\delta$.

Angenommen, die Zeilenmenge B wäre nicht leer. Sei $\mu := \max\left\{\frac{bz}{by} : b \in B\right\}$. Dann gilt $\mu \leq 0$. Somit folgt $z' := z - \mu y \in C$, $A'z' = A'z - \mu A'y = 0$,

$az' = az - \mu ay < 0$, und es gibt ein $b' \in B$ mit $b'z' = 0$. Dies widerspricht jedoch der Maximalität von A' , also ist $B = \emptyset$, d. h. $y \in C$.

Hieraus folgt, dass es einen Index $s \in \{1, \dots, t\}$ gibt, so dass $A'y_s = 0$ und $ay_s = -1$.

Gegeben sei nun ein beliebiges $z \in C$. Es seien a_1, \dots, a_m die Zeilen von A und $\mu := \min \left\{ \frac{a_i z}{a_i y_s} : i = 1, \dots, m, a_i y_s < 0 \right\}$. Dann gilt $\mu \geq 0$. Sei k ein Index für den das Minimum angenommen wird und setze $z' := z - \mu y_s$. Nach Definition von μ haben wir $a_j z' = a_j z - \frac{a_k z}{a_k y_s} a_j y_s$ für $j = 1, \dots, m$, also ist $z' \in C' := \{x \in C : a_k x = 0\}$. Es ist C' ein Kegel, dessen Dimension um eins kleiner ist als die Dimension von C (da $a_k y_s < 0$ und $y_s \in C$). Nach Induktion folgt, dass C' von einer Teilmenge der y_1, \dots, y_t erzeugt wird, demnach ist $z' = \sum_{i=1}^t \lambda_i y_i$ für passende $\lambda_1, \dots, \lambda_t \geq 0$. Setzen wir $\lambda'_s := \lambda_s + \mu$ (beachte, dass $\mu \geq 0$) und $\lambda'_i := \lambda_i$ ($i \neq s$), so erhalten wir $z = z' + \mu y_s = \sum_{i=1}^t \lambda'_i y_i$. \square

Also ist jeder polyedrische Kegel endlich erzeugt. Am Ende von Abschnitt 3.4 werden wir die Umkehrung dieser Aussage beweisen.

3.2 Der Simplexalgorithmus

Der älteste und bekannteste Algorithmus für die LINEARE OPTIMIERUNG ist Dantzsigs [1951] Simplexalgorithmus. Zunächst werden wir annehmen, dass das Polyeder eine Ecke hat und dass irgendeine Ecke als Input vorliegt. Später werden wir dann zeigen, wie allgemeine LPs mit diesem Verfahren gelöst werden können.

Ist J eine Teilmenge der Zeilenindizes, so sei A_J die Untermatrix von A bestehend aus den durch J indizierten Zeilen und b_J der Teilvektor von b bestehend aus den durch J indizierten Komponenten. Der Einfachheit halber schreiben wir $a_i := A_{\{i\}}$ und $\beta_i := b_{\{i\}}$.

SIMPLEXALGORITHMUS

Input: Eine Matrix $A \in \mathbb{R}^{m \times n}$ und Spaltenvektoren $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.
Eine Ecke x von $P := \{x \in \mathbb{R}^n : Ax \leq b\}$.

Output: Eine Ecke x von P , in der $\max\{cx : x \in P\}$ angenommen wird,
oder ein Vektor $w \in \mathbb{R}^n$ mit $Aw \leq 0$ und $cw > 0$
(d. h. das LP ist unbeschränkt).

- ① Wähle eine Menge J von n Zeilenindizes, so dass A_J nichtsingulär und $A_J x = b_J$ ist.
- ② Berechne $c(A_J)^{-1}$ und füge Nullen hinzu, bis ein Vektor y mit $c = yA$ vorliegt, für den alle Komponenten außerhalb J verschwinden.
If $y \geq 0$ **then stop.** **Return** x und y .
- ③ Wähle den kleinsten Index i mit $y_i < 0$.
Sei w die Spalte von $-(A_J)^{-1}$ mit dem Index i . Es gilt also $A_{J \setminus \{i\}} w = 0$ und $a_i w = -1$.
If $Aw \leq 0$ **then stop.**
Return w .

- ④ Sei $\lambda := \min \left\{ \frac{\beta_j - a_j x}{a_j w} : j \in \{1, \dots, m\}, a_j w > 0 \right\}$ und j der kleinste Zeilenindex, für den dieses Minimum angenommen wird.
- ⑤ Setze $J := (J \setminus \{i\}) \cup \{j\}$ und $x := x + \lambda w$.
Go to ②.
-

Schritt ① basiert auf Proposition 3.9 und kann mit GAUSS-ELIMINATION (Abschnitt 4.3) durchgeführt werden. Die Auswahlregeln für i und j in ③ und ④ (oft auch Pivotregel genannt) gehen auf Bland [1977] zurück. Würde man ein beliebiges i mit $y_i < 0$ wählen und ein beliebiges j , für welches das Minimum in ④ angenommen wird, so gäbe es Instanzen, für welche der Algorithmus in eine Schleife gerät. Blands Pivotregel ist nicht die einzige, welche Schleifen verhindert; Dantzig, Orden und Wolfe [1955] hatten bereits gezeigt, dass eine andere (die sogenannte lexikographische Regel) Schleifen verhindert. Bevor wir die Korrektheit des SIMPLEXALGORITHMUS beweisen, möchten wir auf das folgende Resultat (manchmal „schwacher Dualitätssatz“ genannt) hinweisen:

Proposition 3.13. *Sei x bzw. y eine zulässige Lösung des LP*

$$\max\{cx : Ax \leq b\} \quad \text{bzw.} \tag{3.1}$$

$$\min\{yb : y^\top A = c^\top, y \geq 0\}. \tag{3.2}$$

Dann gilt $cx \leq yb$.

Beweis: $cx = (yA)x = y(Ax) \leq yb$. □

Satz 3.14. (Dantzig [1951], Dantzig, Orden und Wolfe [1955], Bland [1977]) *Der SIMPLEXALGORITHMUS terminiert nach höchstens $\binom{m}{n}$ Iterationen. Terminiert er in ② mit dem Ergebnis x und y , so sind diese Vektoren optimale Lösungen der LPs (3.1) bzw. (3.2), mit $cx = yb$. Terminiert der Algorithmus andererseits mit dem Ergebnis w in ③, so ist $cw > 0$ und das LP (3.1) ist unbeschränkt.*

Beweis: Zunächst beweisen wir, dass die folgenden Bedingungen zu jedem Zeitpunkt des Algorithmus erfüllt sind:

- (a) $x \in P$;
- (b) $A_J x = b_J$;
- (c) A_J ist nichtsingulär;
- (d) $cw > 0$;
- (e) $\lambda \geq 0$.

Am Anfang gelten (a) und (b). ② und ③ gewährleisten, dass $cw = yAw = -y_i > 0$. Aus ④ und $x \in P$ folgt $\lambda \geq 0$. Es gilt (c), da $A_{J \setminus \{i\}}w = 0$ und $a_j w > 0$. Es bleibt zu zeigen, dass (a) und (b) unter ⑤ erhalten bleiben.

Wir werden zeigen: Ist $x \in P$, so auch $x + \lambda w$. Für einen Zeilenindex k haben wir zwei Fälle: Ist $a_k w \leq 0$, so folgt $a_k(x + \lambda w) \leq a_k x \leq \beta_k$ (da $\lambda \geq 0$). Andernfalls

haben wir $\lambda \leq \frac{\beta_k - a_k x}{a_k w}$, also folgt $a_k(x + \lambda w) \leq a_k x + a_k w \frac{\beta_k - a_k x}{a_k w} = \beta_k$. (In der Tat wird λ in ④ als die größte Zahl mit $x + \lambda w \in P$ gewählt.)

Um (b) zu beweisen, bemerken wir, dass wir nach ④ $A_{J \setminus \{i\}}w = 0$ und $\lambda = \frac{\beta_j - a_j x}{a_j w}$ haben. Daraus folgt $A_{J \setminus \{i\}}(x + \lambda w) = A_{J \setminus \{i\}}x = b_{J \setminus \{i\}}$ und $a_j(x + \lambda w) = a_j x + a_j w \frac{\beta_j - a_j x}{a_j w} = \beta_j$. Also gilt nach ⑤ wieder $A_J x = b_J$.

Also haben wir stets (a)–(e). Terminierte der Algorithmus in ② mit dem Ergebnis x und y , so sind x bzw. y zulässige Lösungen von (3.1) bzw. (3.2). Nach (a), (b) und (c) ist x eine Ecke von P . Ferner folgt $c x = y A x = y b$, da die Komponenten von y außerhalb J verschwinden. Die Optimalität von x und y folgt nun aus Proposition 3.13.

Terminierte der Algorithmus in ③, so ist das LP (3.1) in der Tat unbeschränkt, da dann $x + \mu w \in P$ für alle $\mu \geq 0$ und $cw > 0$ nach (d).

Abschließend zeigen wir, dass der Algorithmus tatsächlich terminiert. Sei $J^{(k)}$ die Menge J und $x^{(k)}$ der Vektor x bei der k -ten Iteration des SIMPLEXALGORITHMUS. Ist der Algorithmus nach $\binom{m}{n}$ Iterationen noch nicht zu Ende, so gibt es Iterationen $k < l$ mit $J^{(k)} = J^{(l)}$. Nach (b) und (c) ist $x^{(k)} = x^{(l)}$. Nach (d) und (e) kann $c x$ nur wachsen, und es wächst streng, falls $\lambda > 0$. Also verschwindet λ bei allen Iterationen $k, k+1, \dots, l-1$ und $x^{(k)} = x^{(k+1)} = \dots = x^{(l)}$.

Sei h der höchste, J bei einer der Iterationen $k, \dots, l-1$ verlassende Index, etwa in Iteration p . Der Index h wurde auch bei irgendeiner Iteration $q \in \{k, \dots, l-1\}$ zu J hinzugefügt. Es sei y' der Vektor y bei Iteration p und w' der Vektor w bei Iteration q . Dann folgt $y' A w' = c w' > 0$. Sei also r ein Index für den $y'_r a_r w' > 0$. Da $y'_r \neq 0$, ist der Index r aus $J^{(p)}$. Wäre $r > h$, so würde r auch aus $J^{(q)}$ und $J^{(q+1)}$ sein, folglich wäre $a_r w' = 0$. Also ist $r \leq h$. Aber wegen der Wahl von i in Iteration p folgt: $y'_r < 0$ genau dann, wenn $r = h$, und wegen der Wahl von j in Iteration q folgt: $a_r w' > 0$ genau dann, wenn $r = h$ (beachte, dass $\lambda = 0$ und $a_r x^{(q)} = a_r x^{(p)} = \beta_r$, da $r \in J^{(p)}$). Dies ist jedoch ein Widerspruch. \square

Klee und Minty [1972] und Avis und Chvátal [1978] haben Beispiele dafür gefunden, dass der SIMPLEXALGORITHMUS (mit Blands Pivotregel) 2^n Iterationen für LPs mit n Variablen und $2n$ Nebenbedingungen benötigt, womit erwiesen ist, dass er kein polynomieller Algorithmus ist. Es ist nicht bekannt, ob es eine Pivotregel gibt, die zu einem polynomiellen Algorithmus führt. Borgwardt [1982] hat jedoch gezeigt, dass die durchschnittliche Laufzeit (für zufällige Instanzen in einem bestimmten natürlichen probabilistischen Modell) durch ein Polynom beschränkt werden kann. Spielman und Teng [2004] haben eine sogenannte „smoothed analysis“ (siehe Abschnitt 17.5) eingeführt: Für jeden Input betrachten wir die erwartete Laufzeit bezüglich geringer zufälliger Störungen des Inputs. Das Maximum über all diese Erwartungswerte ist polynomiell beschränkt. Kelner und Spielman [2006] haben einen randomisierten polynomiellen Algorithmus für LINEARE OPTIMIERUNG vorgeschlagen, der dem SIMPLEXALGORITHMUS ähnlich ist. In der Praxis ist der SIMPLEXALGORITHMUS auch recht schnell, wenn er geschickt implementiert wird; siehe Abschnitt 3.3.

Wir werden nun zeigen, wie man allgemeine LPs mit dem SIMPLEXALGORITHMUS lösen kann. Genauer: Wir werden zeigen, wie man eine Anfangsecke findet. Da es Polyeder ohne Ecken gibt, werden wir ein gegebenes LP zunächst in eine geeignete Form bringen.

Sei $\max\{cx : Ax \leq b\}$ ein LP. Wir ersetzen x durch $y - z$ und schreiben es in der äquivalenten Form

$$\max \left\{ \begin{pmatrix} c & -c \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} : \begin{pmatrix} A & -A \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} \leq b, \quad y, z \geq 0 \right\}.$$

O. B. d. A. können wir annehmen, dass unser LP die Form

$$\max\{cx : A'x \leq b', \quad A''x \leq b'', \quad x \geq 0\} \quad (3.3)$$

mit $b' \geq 0$ und $b'' < 0$ hat. Nun wenden wir den SIMPLEXALGORITHMUS zunächst auf die Instanz

$$\min\{(\mathbb{I}A'')x + \mathbb{I}y : A'x \leq b', \quad A''x + y \geq b'', \quad x, y \geq 0\} \quad (3.4)$$

an, wobei \mathbb{I} einen Vektor aus lauter Einsen bezeichnet. Dies ist möglich, da $\begin{pmatrix} x \\ y \end{pmatrix} = 0$ eine Ecke ist. Das LP ist offensichtlich nicht unbeschränkt, da das Minimum mindestens $\mathbb{I}b''$ ist. Für jede zulässige Lösung x von (3.3) ist $\begin{pmatrix} x \\ b'' - A''x \end{pmatrix}$ eine optimale Lösung von (3.4) mit dem Wert $\mathbb{I}b''$. Damit folgt: Ist das Minimum von (3.4) größer als $\mathbb{I}b''$, so ist (3.3) unzulässig.

Ist das Gegenteil der Fall, so sei $\begin{pmatrix} x \\ y \end{pmatrix}$ eine optimale Ecke von (3.4) mit dem Wert $\mathbb{I}b''$. Wir behaupten, dass x eine Ecke des durch (3.3) definierten Polyeders ist. Um dies zu sehen, beachten wir zunächst, dass $A''x + y = b''$. Sei n bzw. m die Dimension von x bzw. y ; dann folgt nach Proposition 3.9, dass es eine Menge S von $n+m$ mit Gleichheit erfüllten Ungleichungen aus (3.4) gibt, so dass die diesen $n+m$ Ungleichungen entsprechende Untermatrix nichtsingulär ist.

Sei S' die Menge der Ungleichungen von $A'x \leq b'$ und von $x \geq 0$, die zu S gehören. Sei ferner S'' die Menge derjenigen Ungleichungen von $A''x \leq b''$, für welche die entsprechenden Ungleichungen von $A''x + y \geq b''$ und auch $y \geq 0$ alle zu S gehören. Offensichtlich gilt $|S'| \geq |S| - m = n$ und ferner sind die Ungleichungen aus $S' \cup S''$ linear unabhängig und x erfüllt sie mit Gleichheit. Folglich werden n linear unabhängige Ungleichungen aus (3.3) durch x mit Gleichheit erfüllt; also ist x tatsächlich eine Ecke. Wir können nun den SIMPLEXALGORITHMUS mit (3.3) und x starten.

3.3 Implementierung des Simplexalgorithmus

Die obige Beschreibung des SIMPLEXALGORITHMUS ist zwar einfach, aber sie eignet sich nicht für eine effiziente Implementierung. Wie wir sehen werden, ist es nicht notwendig, in jeder Iteration ein lineares Gleichungssystem zu lösen. Um die Hauptidee einzuführen, beginnen wir mit dem folgenden Resultat (welches wir

aber nicht weiter benötigen werden): Für LPs der Form $\max\{cx : Ax = b, x \geq 0\}$ können Ecken nicht nur durch Teilmengen der Zeilenmenge, sondern auch durch Teilmengen der Spaltenmenge dargestellt werden.

Für eine Matrix A und eine Teilmenge J der Spaltenindizes sei A^J die allein aus den Spalten von J gebildete Untermatrix von A . Allgemeiner bezeichnen wir mit A_I^J die Untermatrix von A mit Zeilen in I und Spalten in J . Gelegentlich spielt die Reihenfolge der Zeilen bzw. Spalten eine Rolle: Ist $J = (j_1, \dots, j_k)$ ein Vektor von Zeilen- bzw. Spaltenindizes, so bezeichnen wir mit A_J bzw. A^J die Matrix, deren i -te Zeile bzw. Spalte die j_i -te Zeile bzw. Spalte von A ($i = 1, \dots, k$) ist.

Proposition 3.15. *Sei $P := \{x : Ax = b, x \geq 0\}$, wobei A eine Matrix und b ein Vektor ist. Dann ist x eine Ecke von P genau dann, wenn $x \in P$ und die den positiven Komponenten von x entsprechenden Spalten von A linear unabhängig sind.*

Beweis: Sei A eine $m \times n$ -Matrix. Sei $X := \begin{pmatrix} -I & 0 \\ A & I \end{pmatrix}$ und $b' := \begin{pmatrix} 0 \\ b \end{pmatrix}$. Seien $N := \{1, \dots, n\}$ und $M := \{n+1, \dots, n+m\}$. Für eine Indexmenge $J \subseteq N \cup M$ mit $|J| = n$, sei $\bar{J} := (N \cup M) \setminus J$. Dann ist X_J^N genau dann nichtsingulär, wenn $X_{M \cap J}^{N \cap \bar{J}}$ nichtsingulär ist, und $X_{M \cap J}^{N \cap \bar{J}}$ ist wiederum nichtsingulär genau dann, wenn $X_M^{\bar{J}}$ nichtsingulär ist.

Ist x eine Ecke von P , so folgt nach Proposition 3.9, dass es eine Menge $J \subseteq N \cup M$ gibt mit $|J| = n$, X_J^N nichtsingulär und $X_J^N x = b'_J$. Somit sind die der Menge $N \cap J$ entsprechenden Komponenten von x gleich Null. Auch ist $X_{M \cap J}^{N \cap \bar{J}}$ nichtsingulär, also sind die Spalten von $A^{N \cap \bar{J}}$ linear unabhängig.

Sei nun umgekehrt $x \in P$ und seien die den positiven Komponenten von x entsprechenden Spalten von A linear unabhängig. Durch das Hinzufügen geeigneter Einheitsvektoren zu diesen Spalten erhalten wir eine nichtsinguläre Untermatrix X_M^B mit $x_i = 0$ für $i \in N \setminus B$. Dann ist $X_{\bar{B}}^N$ nichtsingulär und $X_{\bar{B}}^N x = b'_{\bar{B}}$. Nach Proposition 3.9 folgt sodann, dass x eine Ecke von P ist. \square

Korollar 3.16. *Sei $\begin{pmatrix} x \\ y \end{pmatrix} \in P := \left\{ \begin{pmatrix} x \\ y \end{pmatrix} : Ax + y = b, x \geq 0, y \geq 0 \right\}$. Dann ist $\begin{pmatrix} x \\ y \end{pmatrix}$ eine Ecke von P genau dann, wenn die den positiven Komponenten von $\begin{pmatrix} x \\ y \end{pmatrix}$ entsprechenden Spalten von $(A \ I)$ linear unabhängig sind. Ferner ist x genau dann eine Ecke von $\{x : Ax \leq b, x \geq 0\}$, wenn $\begin{pmatrix} x \\ b - Ax \end{pmatrix}$ eine Ecke von P ist.* \square

Wir werden nun das Verhalten des SIMPLEXALGORITHMUS bei Anwendung auf ein LP der Form $\max\{cx : Ax \leq b, x \geq 0\}$ analysieren.

Satz 3.17. *Sei $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ und $c \in \mathbb{R}^n$. Sei $A' := \begin{pmatrix} -I \\ A \end{pmatrix}$, $b' := \begin{pmatrix} 0 \\ b \end{pmatrix}$ und $\bar{c} := (c^\top, 0)$. Sei $B \in \{1, \dots, n+m\}^m$ mit $(A \ I)^B$ nichtsingulär. Sei $J \subseteq \{1, \dots, n+m\}$ die Menge der weiteren n Indizes. Sei $Q_B := ((A \ I)^B)^{-1}$.*

Dann haben wir die folgenden fünf Aussagen:

- Es ist A'_J ist nichtsingulär.*
- Es gilt $(b' - A'x)_J = 0$ und $(b' - A'x)_B = Q_B b$ und $c^\top x = \bar{c}^B Q_B b$, wobei $x := (A'_J)^{-1} b'_J$.*

- (c) Sei y der Vektor mit $y_B = 0$ und $y^\top A' = c^\top$. Dann gilt $y^\top = \bar{c}^B Q_B(A I) - \bar{c}$.
 (d) Sei $i \in J$. Sei w der Vektor mit $A'_i w = -1$ und $A'_{J \setminus \{i\}} w = 0$. Dann gilt $A'_B w = Q_B(A I)^i$.
 (e) Sei

$$T_B := \left(\begin{array}{c|c} Q_B(A I) & Q_B b \\ \hline \bar{c}^B Q_B(A I) - \bar{c} & c^\top x \end{array} \right).$$

Ferner gehe B' aus B mittels Ersetzen von j durch i hervor, wobei i und j aus ②–④ des SIMPLEXALGORITHMUS (angewendet auf A' , b' , c und die Indexmenge J) hervorgehen. Für gegebene B und T_B können wir dann B' und $T_{B'}$ in $O(m(n+m))$ -Zeit berechnen.

Die Matrix T_B heißt das **Simplextableau** bezüglich der **Basis** B .

Beweis: (a): Sei $N := \{1, \dots, n\}$. Da $(A I)^B$ nichtsingulär ist, ist auch $(A')_{J \setminus N}^{N \setminus J}$ nichtsingulär. Somit ist A'_J nichtsingulär.

- (b): Die erste Aussage folgt sofort aus $A'_J x = b'_J$. Somit gilt
 $b = Ax + I(b - Ax) = (A I)(b' - A'x) = (A I)^B(b' - A'x)_B$ und $c^\top x = \bar{c}(b' - A'x) = \bar{c}^B(b' - A'x)_B = \bar{c}^B Q_B b$.
 (c): Diese Aussage folgt, da $(\bar{c}^B Q_B(A I) - \bar{c})^B = \bar{c}^B Q_B(A I)^B - \bar{c}^B = 0$ und $(\bar{c}^B Q_B(A I) - \bar{c})A' = \bar{c}^B Q_B(A I)A' - c^\top(-I) = c^\top$.
 (d): Diese Aussage folgt, da $0 = (A I)A'w =$

$$(A I)^B(A'_B w) + (A I)^{J \setminus \{i\}}(A'_{J \setminus \{i\}} w) + (A I)^i(A'_i w) = (A I)^B(A'_B w) - (A I)^i.$$

(e): Nach (c) folgt: Es ist y entsprechend ② des SIMPLEXALGORITHMUS die letzte Zeile von T_B . Ist $y \geq 0$, so sind x und y optimal und wir sind am Ziel. Andernfalls ist i der erste Index mit $y_i < 0$, und diesen können wir in $O(n+m)$ -Zeit bestimmen. Hat die i -te Spalte von T_B keine positiven Elemente, so sind wir fertig (das LP ist unbeschränkt und w ist durch (d) gegeben). Andernfalls haben wir mit (b) und (d), dass λ aus ④ des SIMPLEXALGORITHMUS folgendermaßen lautet:

$$\lambda = \min \left\{ \frac{(Q_B b)_j}{(Q_B(A I)^i)_j} : j \in \{1, \dots, m\}, (Q_B(A I)^i)_j > 0 \right\},$$

und unter den Indizes, für welche dieses Minimum angenommen wird, ist j derjenige, für den die j -te Komponente von B minimal ist. Somit können wir j in $O(m)$ -Zeit berechnen, indem wir die i -te und die letzte Spalte von T_B betrachten. Damit erhalten wir B' .

Das aktualisierte Tableau $T_{B'}$ berechnen wir wie folgt: Dividiere die Elemente der j -ten Zeile durch das Element in Zeile j und Spalte i . Addiere ein geeignetes Vielfaches der j -ten Zeile zu allen anderen Zeilen, so dass die i -te Spalte außerhalb von Zeile j nur Nullen hat.

Beachte, dass diese Zeilenoperationen die Eigenschaft des Tableaus, die folgende Form zu haben, nicht zerstören:

$$\left(\begin{array}{c|c} Q(A I) & Q b \\ \hline v(A I) - \bar{c} & v b \end{array} \right)$$

für eine nichtsinguläre Matrix Q und einen Vektor v , und zusätzlich haben wir noch $Q(AI)^{B'} = I$ und $(v(AI) - \bar{c})^{B'} = 0$. Da es nur eine Wahl für Q und v gibt, nämlich $Q = Q_{B'}$ und $v = \bar{c}^{B'} Q_{B'}$, wird das aktualisierte Tableau $T_{B'}$ mit den obigen Operationen in $O(m(n+m))$ -Zeit korrekt berechnet. \square

Um den SIMPLEXALGORITHMUS anzuwerfen, betrachten wir ein LP der Form

$$\max\{cx : A'x \leq b', A''x \leq b'', x \geq 0\},$$

mit $A' \in \mathbb{R}^{m' \times n}$, $A'' \in \mathbb{R}^{m'' \times n}$, $b' \geq 0$ und $b'' < 0$. Zunächst lassen wir den SIMPLEXALGORITHMUS auf der Instanz

$$\min\{(\mathbb{1}A'')x + \mathbb{1}y : A'x \leq b', A''x + y \geq b'', x, y \geq 0\}$$

laufen, mit dem Anfangstableau

$$\left(\begin{array}{cccc|c} A' & 0 & I & 0 & b' \\ -A'' & -I & 0 & I & -b'' \\ \hline \mathbb{1}A'' & \mathbb{1} & 0 & 0 & 0 \end{array} \right), \quad (3.5)$$

entsprechend der Basislösung $x = 0$, $y = 0$. Danach iterieren wir den SIMPLEXALGORITHMUS wie in Satz 3.17(e) angegeben.

Terminiert der Algorithmus mit dem optimalen Zielfunktionswert $\mathbb{1}b$, so ändern wir das letzte Simplextableau wie folgt. Multipliziere einige der Zeilen mit -1 , damit keine der Spalten $n+m''+m'+1, \dots, n+m''+m'+m''$ (der vierte Abschnitt in (3.5)) ein Einheitsvektor ist, entferne den vierten Abschnitt des Tableaus (d. h. die Spalten $n+m''+m'+1, \dots, n+m''+m'+m''$) und ersetze die letzte Zeile durch $(-c, 0, 0, 0)$. Dann addiere geeignete Vielfache der anderen Zeilen zu der letzten Zeile, um Nullen in $m'+m''$ Stellen zu erhalten, denen Spalten entsprechen, die paarweise verschiedene Einheitsvektoren sind; Letztere werden unsere Basis bilden. Es resultiert hiermit das Simplextableau bezüglich des Original-LP und dieser Basis. Somit können wir jetzt den SIMPLEXALGORITHMUS wie in Satz 3.17(e) weiter iterieren.

In der Tat kann man oft sogar effizienter verfahren. Angenommen, wir wollen ein LP $\min\{cx : Ax \geq b, x \geq 0\}$ mit einer sehr großen Anzahl von Ungleichungen lösen, wobei Letztere implizit in einer Form gegeben sind, die uns das folgende Problem effizient lösen lässt: Für einen gegebenen Vektor $x \geq 0$ entscheide man, dass $Ax \geq b$ gilt, oder finde eine verletzte Ungleichung. Hier wenden wir den SIMPLEXALGORITHMUS auf das duale LP $\max\{yb : yA \leq c, y \geq 0\} = \max\{by : A^\top y \leq c, y \geq 0\}$ an. Sei $\bar{b} := (b^\top, 0)$. Wir setzen $Q_B := ((A^\top I)^B)^{-1}$ für eine Basis B und speichern nur den rechten Teil des Simplextableaus

$$\left(\begin{array}{c|c} Q_B & Q_B c \\ \hline \bar{b}^B Q_B & b^\top x \end{array} \right).$$

Die letzte Zeile des vollen Simplextableaus lautet $\bar{b}^B Q_B (A^\top I) - \bar{b}$. Um eine Iteration zu vollziehen, müssen wir prüfen, ob $\bar{b}^B Q_B \geq 0$ und $\bar{b}^B Q_B A^\top - b \geq 0$,

und eine negative Komponente finden, falls es eine solche gibt. Dies reduziert sich auf die Lösung des obigen Problems für $x = (\bar{b}^B Q_B)^\top$. Dann erzeugen wir die entsprechende Spalte des vollen Simplextableaus, aber nur für die aktuelle Iteration. Nach der Aktualisierung des reduzierten Tableaus können wir sie wieder löschen. Dieses Verfahren ist als die *revidierte Simplexmethode* oder auch als *Spaltenerzeugung* bekannt. Anwendungen werden wir später antreffen.

3.4 Dualität

Satz 3.14 zeigt, dass die LPs (3.1) und (3.2) verwandt sind. Dies legt die folgende Definition nahe:

Definition 3.18. Ist $\max\{cx : Ax \leq b\}$ ein LP, so definieren wir das dazu **duale LP** als das lineare Programm $\min\{yb : yA = c, y \geq 0\}$.

Das ursprüngliche LP $\max\{cx : Ax \leq b\}$ wird oft als **primales LP** bezeichnet.

Proposition 3.19. Das Dual des Duals eines LP ist das ursprüngliche LP (oder ist äquivalent mit dem ursprünglichen LP).

Beweis: Gegeben sei das LP $\max\{cx : Ax \leq b\}$. Das dazu duale LP ist $\min\{yb : yA = c, y \geq 0\}$ oder, äquivalent ausgedrückt,

$$-\max \left\{ -by : \begin{pmatrix} A^\top \\ -A^\top \\ -I \end{pmatrix} y \leq \begin{pmatrix} c \\ -c \\ 0 \end{pmatrix} \right\}.$$

(Jede Gleichungsnebenbedingung wurde in zwei Ungleichungen aufgespalten.) Also ist das Dual des Duals

$$-\min \left\{ zc - z'c : \begin{pmatrix} A & -A & -I \end{pmatrix} \begin{pmatrix} z \\ z' \\ w \end{pmatrix} = -b, z, z', w \geq 0 \right\}$$

und dieses LP ist äquivalent mit $-\min\{-cx : -Ax - w = -b, w \geq 0\}$ (wobei wir $z' - z$ durch x ersetzt haben). Nach Elimination der Schlupfvariablen w sieht man sofort, dass dieses LP mit dem Ausgangs-LP äquivalent ist. \square

Als Nächstes leiten wir den wichtigsten Satz der LP-Theorie ab, nämlich den Dualitätssatz:

Satz 3.20. (von Neumann [1947], Gale, Kuhn und Tucker [1951]) Sind die Polyeder $P := \{x : Ax \leq b\}$ und $D := \{y : yA = c, y \geq 0\}$ beide nicht leer, so gilt $\max\{cx : x \in P\} = \min\{yb : y \in D\}$.

Beweis: Ist D nicht leer, so hat es eine Ecke y . Nun wenden wir den SIMPLEXALGORITHMUS auf $\min\{yb : y \in D\}$ und y an. Nach Proposition 3.13 folgt: die Existenz eines Punktes $x \in P$ gewährleistet, dass $\min\{yb : y \in D\}$ nicht unbeschränkt ist. Damit folgt nach Satz 3.14, dass der SIMPLEXALGORITHMUS optimale Lösungen y bzw. z des LP $\min\{yb : y \in D\}$ bzw. seines Duals liefert. Nach Proposition 3.19 ist das Dual aber $\max\{cx : x \in P\}$. Damit folgt $yb = cz$, wie erwünscht. \square

Wir können jedoch noch mehr über den Zusammenhang zwischen den optimalen Lösungen des primalen und dualen LP sagen:

Korollar 3.21. Seien $\max\{cx : Ax \leq b\}$ und $\min\{yb : yA = c, y \geq 0\}$ ein primal-duales LP-Paar. Seien ferner x und y zulässige Lösungen, d.h. $Ax \leq b$, $yA = c$ und $y \geq 0$. Dann sind die folgenden drei Aussagen äquivalent:

- (a) x und y sind beide optimale Lösungen.
- (b) $cx = yb$.
- (c) $y(b - Ax) = 0$.

Beweis: Die Äquivalenz von (a) und (b) folgt sofort aus dem Dualitätssatz 3.20. Die Äquivalenz von (b) und (c) folgt aus $y(b - Ax) = yb - yAx = yb - cx$. \square

Eigenschaft (c) der optimalen Lösungen wird oft als **komplementärer Schlupf** bezeichnet. Sie kann auch folgendermaßen formuliert werden: Ein Punkt $x^* \in P = \{x : Ax \leq b\}$ ist genau dann eine optimale Lösung von $\max\{cx : x \in P\}$, wenn c eine nichtnegative Linearkombination derjenigen Zeilen von A ist, die den von x^* mit Gleichheit erfüllten Ungleichungen von $Ax \leq b$ entsprechen. Aus Eigenschaft (c) folgt ferner:

Korollar 3.22. Sei $P = \{x : Ax \leq b\}$ ein Polyeder und $\emptyset \neq Z \subseteq P$. Dann ist die Menge der Vektoren c , für die jedes $z \in Z$ eine optimale Lösung von $\max\{cx : x \in P\}$ ist, der von den Zeilen von A' erzeugte Kegel, wobei $A'x \leq b'$ das maximale Teilsystem von $Ax \leq b$ mit $A'z = b'$ für alle $z \in Z$ ist.

Beweis: Es gibt ein $z \in \text{conv}(Z)$, welches alle anderen Ungleichungen des Systems $Ax \leq b$ streng erfüllt. Sei ferner c ein Vektor mit der Eigenschaft, dass jedes Element aus Z , also auch z , eine optimale Lösung von $\max\{cx : x \in P\}$ ist. Nach Korollar 3.21 gibt es dann ein $y \geq 0$ mit $c = yA'$, d.h. c ist eine nichtnegative Linearkombination der Zeilen von A' .

Zur umgekehrten Richtung: Für eine Zeile $a'x \leq \beta'$ von $A' \leq b'$ und $z \in Z$ haben wir $a'z = \beta' = \max\{a'x : x \in P\}$. \square

Korollar 3.21 können wir auch in der folgenden Form schreiben:

Korollar 3.23. Seien $\min\{cx : Ax \geq b, x \geq 0\}$ und $\max\{yb : yA \leq c, y \geq 0\}$ ein primal-duales LP-Paar. Seien ferner x und y zulässige Lösungen, d.h. $Ax \geq b$, $yA \leq c$ und $x, y \geq 0$. Dann sind die folgenden drei Aussagen äquivalent:

- (a) x und y sind beide optimale Lösungen.
- (b) $cx = yb$.
- (c) $(c - yA)x = 0$ und $y(b - Ax) = 0$.

Beweis: Die Äquivalenz von (a) und (b) folgt durch Anwendung des Dualitätsatzes 3.20 auf $\max \{(-c)x : \begin{pmatrix} -A \\ -I \end{pmatrix}x \leq \begin{pmatrix} -b \\ 0 \end{pmatrix}\}$.

Zum Beweis der Äquivalenz von (b) und (c) bemerken wir, dass $y(b - Ax) \leq 0 \leq (c - yA)x$ für beliebige zulässige Lösungen x und y , und dass $y(b - Ax) = (c - yA)x$ genau dann, wenn $yb = cx$. \square

Die beiden Bedingungen in (c) werden manchmal als **primale** und **duale Bedingungen des komplementären Schlupfes** bezeichnet.

Der Dualitätssatz hat viele Anwendungen in der kombinatorischen Optimierung. Ein Grund für seine Bedeutung ist, dass die Optimalität einer Lösung durch Kenntnis einer zulässigen Lösung des dualen LP mit demselben Zielfunktionswert bewiesen werden kann. Wir werden nun zeigen, wie man beweist, dass ein LP unbeschränkt oder unzulässig ist:

Satz 3.24. Es gibt einen Vektor x mit $Ax \leq b$ genau dann, wenn $yb \geq 0$ für jeden Vektor $y \geq 0$ mit $yA = 0$.

Beweis: Gibt es einen Vektor x mit $Ax \leq b$, so gilt $yb \geq yAx = 0$ für jedes $y \geq 0$ mit $yA = 0$.

Betrachte das LP

$$-\min\{\mathbb{1}w : Ax - w \leq b, w \geq 0\}. \quad (3.6)$$

In Standardform lautet dieses

$$\max \left\{ \begin{pmatrix} 0 & -\mathbb{1} \end{pmatrix} \begin{pmatrix} x \\ w \end{pmatrix} : \begin{pmatrix} A & -I \\ 0 & -I \end{pmatrix} \begin{pmatrix} x \\ w \end{pmatrix} \leq \begin{pmatrix} b \\ 0 \end{pmatrix} \right\},$$

und das Dual hierzu ist das LP

$$\min \left\{ \begin{pmatrix} b & 0 \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} : \begin{pmatrix} A^\top & 0 \\ -I & -I \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ -\mathbb{1} \end{pmatrix}, y, z \geq 0 \right\},$$

oder, äquivalent ausgedrückt,

$$\min\{yb : yA = 0, 0 \leq y \leq \mathbb{1}\}. \quad (3.7)$$

Da sowohl (3.6) als auch (3.7) die Lösung $(x = 0, w = |b|, y = 0)$ hat, können wir Satz 3.20 anwenden. Somit stimmen die optimalen Zielfunktionswerte von (3.6) und (3.7) überein. Das System $Ax \leq b$ hat aber genau dann eine Lösung, wenn der optimale Zielfunktionswert von (3.6) null ist, womit der Satz bewiesen ist. \square

Es kann also die Tatsache, dass ein lineares Ungleichungssystem $Ax \leq b$ keine Lösungen hat, dadurch bewiesen werden, dass man einen Vektor $y \geq 0$ mit $yA = 0$ und $yb < 0$ vorlegt. Wir erwähnen noch zwei äquivalente Formulierungen von Satz 3.24:

Korollar 3.25. Es gibt einen Vektor $x \geq 0$ mit $Ax \leq b$ genau dann, wenn $yb \geq 0$ für jeden Vektor $y \geq 0$ mit $yA \geq 0$.

Beweis: Man wende Satz 3.24 auf das System $\begin{pmatrix} A \\ -I \end{pmatrix} x \leq \begin{pmatrix} b \\ 0 \end{pmatrix}$ an. \square

Korollar 3.26. (Farkas [1894]) Es gibt einen Vektor $x \geq 0$ mit $Ax = b$ genau dann, wenn $yb \geq 0$ für jeden Vektor y mit $yA \geq 0$.

Beweis: Man wende Korollar 3.25 auf das System $\begin{pmatrix} A \\ -A \end{pmatrix} x \leq \begin{pmatrix} b \\ -b \end{pmatrix}$, $x \geq 0$ an. \square

Korollar 3.26 ist bekannt unter dem Namen Farkas' Lemma. Andererseits folgt der Dualitätssatz 3.20 aus den obigen Resultaten, was interessant ist, da sie relativ leicht direkt bewiesen werden können (sie waren tatsächlich bereits vor dem SIMPLEXALGORITHMUS bekannt); siehe Aufgaben 11 und 12.

Wir haben gesehen, wie man beweisen kann, dass ein LP unzulässig ist. Wie kann man nun beweisen, dass ein LP unbeschränkt ist? Der nächste Satz beantwortet diese Frage.

Satz 3.27. Ist ein LP unbeschränkt, so ist das dazu duale LP unzulässig. Hat ein LP eine optimale Lösung, so hat das dazu duale LP auch eine optimale Lösung.

Beweis: Die erste Aussage folgt unmittelbar aus Proposition 3.13.

Zum Beweis der zweiten Aussage nehmen wir an, dass das (primale) LP $\max\{cx : Ax \leq b\}$ eine optimale Lösung x^* hat, dass aber das dazu duale LP $\min\{yb : yA = c, y \geq 0\}$ unzulässig ist (nach der ersten Aussage kann es nicht unbeschränkt sein). Es gibt also kein $y \geq 0$ mit $A^\top y = c$. Mit Farkas' Lemma (Korollar 3.26) bekommen wir somit einen Vektor z mit $zA^\top \geq 0$ und $zc < 0$. Dann ist aber $x^* - z$ primal zulässig, da $A(x^* - z) = Ax^* - Az \leq b$. Daraus folgt $c(x^* - z) > cx^*$, im Widerspruch zur Optimalität von x^* . \square

Es gibt somit vier Fälle für primal-duale LP-Paare: Entweder haben beide eine optimale Lösung (welche dann gleiche Zielfunktionswerte ergeben), oder eines der beiden LPs ist unzulässig und das andere unbeschränkt, oder aber es sind beide unzulässig. Wir bemerken ferner:

Korollar 3.28. Ein zulässiges LP $\max\{cx : Ax \leq b\}$ ist genau dann beschränkt, wenn c in dem von den Zeilen von A erzeugten Kegel liegt.

Beweis: Das LP ist genau dann beschränkt, wenn das duale LP zulässig ist, d.h. wenn es ein $y \geq 0$ mit $yA = c$ gibt. \square

Mit Farkas' Lemma können wir ferner beweisen, dass jeder endlich erzeugte Kegel polyedrisch ist:

Satz 3.29. (Minkowski [1896], Weyl [1935]) Ein Kegel ist genau dann polyedrisch, wenn er endlich erzeugt ist.

Beweis: Die Notwendigkeit folgt unmittelbar aus Lemma 3.12. Sei nun C ein von a_1, \dots, a_t erzeugter Kegel. Wir müssen beweisen, dass C polyedrisch ist. Sei A die Matrix mit den Zeilen a_1, \dots, a_t .

Nach Lemma 3.12 wird der Kegel $D := \{x : Ax \leq 0\}$ von irgendwelchen Vektoren b_1, \dots, b_s erzeugt. Sei B die Matrix mit den Zeilen b_1, \dots, b_s . Wir werden nun beweisen, dass $C = \{x : Bx \leq 0\}$.

Da $b_j a_i = a_i b_j \leq 0$ für alle i und j , ist $C \subseteq \{x : Bx \leq 0\}$. Angenommen, es gäbe einen Vektor $w \notin C$ mit $Bw \leq 0$. Aber $w \notin C$ bedeutet, dass es kein $v \geq 0$ gibt mit $A^\top v = w$. Nach Farkas' Lemma (Korollar 3.26) folgt dann, dass es einen Vektor y gibt mit $yw < 0$ und $Ay \geq 0$. Somit ist $-y \in D$. Da D von b_1, \dots, b_s erzeugt wird, haben wir $-y = zB$ für ein passendes $z \geq 0$. Das ergibt aber den Widerspruch $0 < -yw = zBw \leq 0$. \square

3.5 Konvexe Hüllen und Polytope

In diesem Abschnitt stellen wir einige weitere Fakten über Polytope zusammen. Insbesondere wird gezeigt, dass Polytope gerade diejenigen Mengen sind, welche die konvexe Hülle einer endlichen Anzahl von Punkten sind. Zunächst erinnern wir an einige grundlegende Definitionen:

Definition 3.30. Gegeben seien Vektoren $x_1, \dots, x_k \in \mathbb{R}^n$ und $\lambda_1, \dots, \lambda_k \geq 0$ mit $\sum_{i=1}^k \lambda_i = 1$. Dann nennen wir $x = \sum_{i=1}^k \lambda_i x_i$ eine **Konvexitätskombination** von x_1, \dots, x_k . Eine Menge $X \subseteq \mathbb{R}^n$ ist **konvex**, falls $\lambda x + (1 - \lambda)y \in X$ für alle $x, y \in X$ und $\lambda \in [0, 1]$. Die **konvexe Hülle** $\text{conv}(X)$ einer Menge X ist die Menge aller Konvexitätskombinationen von Punkten in X . Ein **Extrempunkt** einer Menge X ist ein Element $x \in X$ mit $x \notin \text{conv}(X \setminus \{x\})$.

Eine Menge X ist also genau dann konvex, wenn alle Konvexitätskombinationen von Punkten in X wieder in X liegen. Die konvexe Hülle einer Menge X ist die kleinste X enthaltende konvexe Menge. Ferner ist der Durchschnitt konvexer Mengen wieder eine konvexe Menge. Also sind Polyeder konvex. Wir beweisen nun den „Satz über endliche Basen für Polytope“, ein grundlegendes Resultat, welches durchaus plausibel erscheint, dessen direkter Beweis aber keineswegs trivial ist:

Satz 3.31. (Minkowski [1896], Steinitz [1916], Weyl [1935]) Eine Menge P ist ein Polytop genau dann, wenn sie die konvexe Hülle einer endlichen Menge von Punkten ist.

Beweis: (Schrijver [1986]) Sei $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ ein nichtleeres Polytop. Offensichtlich gilt

$$P = \left\{ x : \begin{pmatrix} x \\ 1 \end{pmatrix} \in C \right\}, \quad \text{mit} \quad C = \left\{ \begin{pmatrix} x \\ \lambda \end{pmatrix} \in \mathbb{R}^{n+1} : \lambda \geq 0, Ax - \lambda b \leq 0 \right\}.$$

Es ist C ein polyedrischer Kegel, also folgt nach Satz 3.29, dass C von endlich vielen nichtverschwindenden Vektoren, etwa $\begin{pmatrix} x_1 \\ \lambda_1 \end{pmatrix}, \dots, \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix}$, erzeugt wird. Da P

beschränkt ist, verschwindet keines der λ_i ; o. B. d. A. können wir annehmen, dass alle λ_i gleich 1 sind. Folglich ist $x \in P$ genau dann, wenn

$$\begin{pmatrix} x \\ 1 \end{pmatrix} = \mu_1 \begin{pmatrix} x_1 \\ 1 \end{pmatrix} + \cdots + \mu_k \begin{pmatrix} x_k \\ 1 \end{pmatrix}$$

für geeignete $\mu_1, \dots, \mu_k \geq 0$. Mit anderen Worten, P ist die konvexe Hülle von x_1, \dots, x_k .

Nun sei P die konvexe Hülle von $x_1, \dots, x_k \in \mathbb{R}^n$. Dann ist $x \in P$ genau dann, wenn $\begin{pmatrix} x \\ 1 \end{pmatrix} \in C$, wobei C der von $\begin{pmatrix} x_1 \\ 1 \end{pmatrix}, \dots, \begin{pmatrix} x_k \\ 1 \end{pmatrix}$ erzeugte Kegel ist. Nach Satz 3.29 ist C polyedrisch, folglich ist

$$C = \left\{ \begin{pmatrix} x \\ \lambda \end{pmatrix} : Ax + b\lambda \leq 0 \right\}.$$

Daraus folgt, dass $P = \{x \in \mathbb{R}^n : Ax + b \leq 0\}$. □

Korollar 3.32. *Ein Polytop ist die konvexe Hülle seiner Ecken.*

Beweis: Sei P ein Polytop. Nach Satz 3.31 ist die konvexe Hülle der Ecken von P ein Polytop Q . Offensichtlich ist $Q \subseteq P$. Angenommen, es gäbe einen Punkt $z \in P \setminus Q$. Dann gibt es einen Vektor c mit $cz > \max\{cx : x \in Q\}$. Die stützende Hyperebene $\{x : cx = \max\{cy : y \in P\}\}$ von P definiert eine Seitenfläche von P , die keine Ecke enthält. Nach Korollar 3.10 ist dies aber unmöglich. □

Die beiden vorhergehenden Resultate, zusammen mit dem nächsten, bilden den Ausgangspunkt der polyedrischen Kombinatorik; sie werden in diesem Buch sehr oft verwendet. Ist E eine gegebene Grundmenge und $X \subseteq E$, so wird der **Inzidenzvektor** von X (bezüglich E) definiert als der Vektor $x \in \{0, 1\}^E$ mit $x_e = 1$ für $e \in X$ und $x_e = 0$ für $e \in E \setminus X$.

Korollar 3.33. *Sei (E, \mathcal{F}) ein Mengensystem, P die konvexe Hülle der Inzidenzvektoren der Elemente von \mathcal{F} und $c : E \rightarrow \mathbb{R}$. Dann ist $\max\{cx : x \in P\} = \max\{c(X) : X \in \mathcal{F}\}$.*

Beweis: Es ist $\max\{cx : x \in P\} \geq \max\{c(X) : X \in \mathcal{F}\}$ trivial, also sei x eine optimale Lösung von $\max\{cx : x \in P\}$ (beachte, dass P nach Satz 3.31 ein Polytop ist). Aus der Definition von P folgt, dass x eine Konvexitätskombination der Inzidenzvektoren y_1, \dots, y_k von k Elementen aus \mathcal{F} ist: $x = \sum_{i=1}^k \lambda_i y_i$ für geeignete $\lambda_1, \dots, \lambda_k \geq 0$ mit $\sum_{i=1}^k \lambda_i = 1$. Da $cx = \sum_{i=1}^k \lambda_i cy_i$, gilt $cy_i \geq cx$ für mindestens ein $i \in \{1, \dots, k\}$. Dieses y_i ist der Inzidenzvektor einer Menge $Y \in \mathcal{F}$ mit $c(Y) = cy_i \geq cx$. □

Aufgaben

- Sei H ein Hypergraph, $F \subseteq V(H)$ und $x, y : F \rightarrow \mathbb{R}$. Man möchte $x, y : V(H) \setminus F \rightarrow \mathbb{R}$ finden, so dass $\sum_{e \in E(H)} (\max_{v \in e} x(v) - \min_{v \in e} x(v)) +$

$\max_{v \in e} y(v) - \min_{v \in e} y(v)$) minimal ist. Man zeige, dass dies als LP formuliert werden kann.

Bemerkung: Dieses Problem ist eine Relaxierung des Placement-Problems im VLSI-Design (VLSI bedeutet „very large scale integration“). Dort heißt H die Netzliste, und die Knoten von H entsprechen den Modulen, die auf dem Chip platziert werden müssen. Manche (diejenigen aus F) werden vorab platziert. Die Hauptschwierigkeit ist, dass sich die Module nicht überlappen dürfen; dieses Problem wird in obiger Relaxierung ignoriert.

2. Eine Menge von Vektoren x_1, \dots, x_k heißt affin unabhängig wenn es kein $\lambda \in \mathbb{R}^k \setminus \{0\}$ mit $\lambda^\top \mathbf{1} = 0$ und $\sum_{i=1}^k \lambda_i x_i = 0$ gibt. Sei $\emptyset \neq X \subseteq \mathbb{R}^n$. Man beweise, dass die maximale Kardinalität einer affin unabhängigen Menge von Elementen aus X gleich $\dim X + 1$ ist.
3. Seien $P, Q \in \mathbb{R}^n$ Polyeder. Man beweise, dass der Abschluss von $\text{conv}(P \cup Q)$ ein Polyeder ist. Man zeige mit einem Gegenbeispiel, dass es Polyeder P und Q gibt, für die $\text{conv}(P \cup Q)$ kein Polyeder ist.
4. Man zeige, dass das Problem der Berechnung der größten Kugel, die eine Teilmenge eines gegebenen Polyeders ist, als LP formuliert werden kann.
5. Sei P ein Polyeder. Man beweise, dass die Dimension einer jeden Facette von P eins weniger als die Dimension von P ist.
6. Sei F eine minimale Seitenfläche eines Polyeders $\{x : Ax \leq b\}$. Man beweise, dass dann $Ax = Ay$ für alle $x, y \in F$ gilt.
7. Es seien $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ und $u \in \mathbb{Z}^n$. Man betrachte das LP $\max\{cx : Ax \leq b, 0 \leq x \leq u\}$ und beweise: Hat dieses LP eine optimale Lösung, so hat es auch eine optimale Lösung mit höchstens m nicht-ganzzahligen Komponenten.
8. Man formuliere das duale LP der LP-Formulierung (1.1) des JOB-ZUORDNUNGSPROBLEMS. Wie löst man das primale und das duale LP, falls es nur zwei Jobs gibt (mit einem einfachen Algorithmus)?
9. Sei G ein Digraph, $c : E(G) \rightarrow \mathbb{R}_+$, $E_1, E_2 \subseteq E(G)$ und $s, t \in V(G)$. Man betrachte das folgende LP:

$$\begin{aligned} \min \quad & \sum_{e \in E(G)} c(e)y_e \\ \text{bzgl.} \quad & y_e \geq z_w - z_v \quad (e = (v, w) \in E(G)) \\ & z_t - z_s = 1 \\ & y_e \geq 0 \quad (e \in E_1) \\ & y_e \leq 0 \quad (e \in E_2) \end{aligned}$$

und beweise, dass es eine optimale Lösung (y, z) und ein $s \in X \subseteq V(G) \setminus \{t\}$ gibt, mit $y_e = 1$ für alle $e \in \delta^+(X)$, $y_e = -1$ für alle $e \in \delta^-(X) \setminus E_1$ und $y_e = 0$ für alle anderen Kanten e .

Hinweis: Man betrachte die Bedingungen des komplementären Schlupfes für diejenigen Kanten, welche ein Ende in $\{v \in V(G) : z_v \leq z_s\}$ haben.

10. Sei $Ax \leq b$ ein lineares Ungleichungssystem mit n Variablen. Durch Multiplikation der Zeilen von A mit geeigneten positiven Konstanten können wir erreichen, dass die erste Spalte von A nur die Komponenten 0, -1 und 1 hat. Somit können wir $Ax \leq b$ in der folgenden äquivalenten Form schreiben:

$$\begin{aligned} a'_i x' &\leq b_i & (i = 1, \dots, m_1), \\ -x_1 + a'_j x' &\leq b_j & (j = m_1 + 1, \dots, m_2), \\ x_1 + a'_k x' &\leq b_k & (k = m_2 + 1, \dots, m), \end{aligned}$$

wobei $x' = (x_2, \dots, x_n)$ ist und a'_1, \dots, a'_m die Zeilen von A ohne ihre ersten Komponenten sind. Dann kann man x_1 eliminieren: Man beweise, dass $Ax \leq b$ genau dann eine Lösung hat, wenn das System

$$\begin{aligned} a'_i x' &\leq b_i & (i = 1, \dots, m_1), \\ a'_j x' - b_j &\leq b_k - a'_k x' & (j = m_1 + 1, \dots, m_2, k = m_2 + 1, \dots, m) \end{aligned}$$

eine Lösung hat. Man zeige: Die schrittweise Wiederholung dieses Verfahrens führt zu einem Algorithmus zur Lösung eines linearen Ungleichungssystems $Ax \leq b$ (oder zum Beweis seiner Unzulässigkeit).

Bemerkung: Diese Methode ist als Fourier-Motzkin-Elimination bekannt, weil sie von Fourier vorgeschlagen und von Motzkin [1936] studiert wurde. Man kann zeigen, dass dies kein polynomieller Algorithmus ist.

11. Man benutze Fourier-Motzkin-Elimination (Aufgabe 10), um Satz 3.24 direkt zu beweisen.
(Kuhn [1956])
12. Man zeige, dass der Dualitätssatz 3.20 aus Satz 3.24 folgt.
13. Man beweise den Dekompositionssatz für Polyeder: Jedes Polyeder P kann in der Form $P = \{x + c : x \in X, c \in C\}$ geschrieben werden, wobei X ein Polytop und C ein polyedischer Kegel ist.
(Motzkin [1936])
- * 14. Es sei P ein rationales Polyeder und F eine Seitenfläche von P . Man zeige, dass

$$\{c : cz = \max \{cx : x \in P\} \text{ für alle } z \in F\}$$

ein rationaler polyedrischer Kegel ist.

15. Man beweise den Satz von Carathéodory:
Ist $X \subseteq \mathbb{R}^n$ und $y \in \text{conv}(X)$, so gibt es $x_1, \dots, x_{n+1} \in X$ mit $y \in \text{conv}(\{x_1, \dots, x_{n+1}\})$.
(Carathéodory [1911])
16. Man beweise die folgende Erweiterung des Satzes von Carathéodory (Aufgabe 15):
Ist $X \subseteq \mathbb{R}^n$ und sind $y, z \in \text{conv}(X)$, so gibt es $x_1, \dots, x_n \in X$ mit $y \in \text{conv}(\{z, x_1, \dots, x_n\})$.
17. Man beweise, dass die Extrempunkte eines Polyeders gerade seine Ecken sind.

18. Sei P ein nichtleeres Polytop. Man betrachte den Graphen $G(P)$, dessen Knoten den Ecken von P und dessen Kanten den eindimensionalen Seitenflächen von P entsprechen. Sei x eine Ecke von P und c ein Vektor mit $c^\top x < \max\{c^\top z : z \in P\}$. Man beweise, dass es dann einen Nachbarn y von x in $G(P)$ gibt, mit $c^\top x < c^\top y$.
- * 19. Man verweise Aufgabe 18 um zu beweisen, dass $G(P)$ für jedes n -dimensionale Polytop P ($n \geq 1$) n -fach zusammenhängend ist.
20. Sei $P \subseteq \mathbb{R}^n$ ein nicht notwendigerweise rationales Polytop und $y \notin P$. Man beweise, dass es einen rationalen Vektor c mit $\max\{cx : x \in P\} < cy$ gibt. Man zeige, dass die Aussage nicht für allgemeine Polyeder gilt.
21. Sei $X \subset \mathbb{R}^n$ eine nichtleere konvexe Menge, \bar{X} der Abschluss von X und $y \notin X$. Man beweise:
- (a) Es gibt einen eindeutig bestimmten Punkt in \bar{X} mit minimalem Abstand von y .
 - (b) Es gibt einen Vektor $a \in \mathbb{R}^n \setminus \{0\}$ mit $a^\top x \leq a^\top y$ für alle $x \in X$.
 - (c) Ist $y \notin \bar{X}$, so gibt es einen Vektor $a \in \mathbb{R}^n$ mit $a^\top x < a^\top y$ für alle $x \in X$.
 - (d) Ist X beschränkt und $y \notin \bar{X}$, so gibt es einen Vektor $a \in \mathbb{Q}^n$ mit $a^\top x < a^\top y$ für alle $x \in X$.
 - (e) Eine abgeschlossene konvexe Menge ist der Durchschnitt aller abgeschlossenen Halbräume, in denen sie enthalten ist.

Literatur

Allgemeine Literatur:

- Bertsimas, D., und Tsitsiklis, J.N. [1997]: Introduction to Linear Optimization. Athena Scientific, Belmont 1997
- Chvátal, V. [1983]: Linear Programming. Freeman, New York 1983
- Matoušek, J., und Gärtner, B. [2007]: Understanding and Using Linear Programming. Springer, Berlin 2007
- Padberg, M. [1999]: Linear Optimization and Extensions. 2. Aufl. Springer, Berlin 1999
- Schrijver, A. [1986]: Theory of Linear and Integer Programming. Wiley, Chichester 1986

Zitierte Literatur:

- Avis, D., und Chvátal, V. [1978]: Notes on Bland's pivoting rule. Mathematical Programming Study 8 (1978), 24–34
- Bland, R.G. [1977]: New finite pivoting rules for the simplex method. Mathematics of Operations Research 2 (1977), 103–107
- Borgwardt, K.-H. [1982]: The average number of pivot steps required by the simplex method is polynomial. Zeitschrift für Operations Research 26 (1982), 157–177
- Carathéodory, C. [1911]: Über den Variabilitätsbereich der Fourierschen Konstanten von positiven harmonischen Funktionen. Rendiconto del Circolo Matematico di Palermo 32 (1911), 193–217
- Dantzig, G.B. [1951]: Maximization of a linear function of variables subject to linear inequalities. In: Activity Analysis of Production and Allocation (T.C. Koopmans, Hrsg.), Wiley, New York 1951, pp. 359–373

- Dantzig, G.B., Orden, A., und Wolfe, P. [1955]: The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics* 5 (1955), 183–195
- Farkas, G. [1894]: A Fourier-féle mechanikai elv alkalmazásai. *Mathematikai és Természettudományi Értesítő* 12 (1894), 457–472
- Gale, D., Kuhn, H.W., und Tucker, A.W. [1951]: Linear programming and the theory of games. In: *Activity Analysis of Production and Allocation* (T.C. Koopmans, Hrsg.), Wiley, New York 1951, pp. 317–329
- Hoffman, A.J., und Kruskal, J.B. [1956]: Integral boundary points of convex polyhedra. In: *Linear Inequalities and Related Systems*; Annals of Mathematical Study 38 (H.W. Kuhn, A.W. Tucker, Hrsg.), Princeton University Press, Princeton 1956, pp. 223–246
- Kelner, J.A., und Spielman, D.A. [2006]: A randomized polynomial-time simplex algorithm for linear programming. *Proceedings of the 38th Annual ACM Symposium on Theory of Computing* (2006), 51–60
- Klee, V., und Minty, G.J. [1972]: How good is the simplex algorithm? In: *Inequalities III* (O. Shisha, Hrsg.), Academic Press, New York 1972, pp. 159–175
- Kuhn, H.W. [1956]: Solvability and consistency for linear equations and inequalities. *The American Mathematical Monthly* 63 (1956), 217–232
- Minkowski, H. [1896]: *Geometrie der Zahlen*. Teubner, Leipzig 1896
- Motzkin, T.S. [1936]: Beiträge zur Theorie der linearen Ungleichungen (Dissertation). Azriel, Jerusalem 1936
- von Neumann, J. [1947]: Discussion of a maximum problem. Working paper. Published in: John von Neumann, *Collected Works*; Vol. VI (A.H. Taub, Hrsg.), Pergamon Press, Oxford 1963, pp. 27–28
- Spielman, D.A., und Teng, S.-H. [2004]: Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. *Journal of the ACM* 51 (2004), 385–463
- Steinitz, E. [1916]: Bedingt konvergente Reihen und konvexe Systeme. *Journal für die reine und angewandte Mathematik* 146 (1916), 1–52
- Weyl, H. [1935]: Elementare Theorie der konvexen Polyeder. *Commentarii Mathematici Helvetici* 7 (1935), 290–306



4 Algorithmen für lineare Optimierung

Drei Typen von Algorithmen für die LINEARE OPTIMIERUNG hatten die größte Auswirkung, nämlich der SIMPLEXALGORITHMUS (siehe Abschnitt 3.2), die Innere-Punkte-Algorithmen und die ELLIPSOIDMETHODE.

Jede dieser Typen hat aber Nachteile. Im Gegensatz zu den anderen beiden, gibt es bis dato keine Variante des SIMPLEXALGORITHMUS, die bewiesenermaßen polynomiell läuft. In den Abschnitten 4.4 und 4.5 präsentieren wir die ELLIPSOIDMETHODE und beweisen, dass sie zu einem polynomiellen Algorithmus für die LINEARE OPTIMIERUNG führt. Die ELLIPSOIDMETHODE ist jedoch zu ineffizient, um in der Praxis von Nutzen zu sein. Innere-Punkte-Algorithmen und der SIMPLEXALGORITHMUS, trotz seiner exponentiellen Worst-Case-Laufzeit, sind bei weitem effizienter und werden beide in der Praxis zur Lösung von LPs benutzt. Tatsächlich können sowohl die ELLIPSOIDMETHODE als auch die Innere-Punkte-Algorithmen für allgemeinere konvexe Optimierungsprobleme benutzt werden, z. B. für sogenannte semidefinite Optimierungsprobleme.

Ein Vorteil des SIMPLEXALGORITHMUS und auch der ELLIPSOIDMETHODE ist, dass man für deren Anwendung das zu lösende LP nicht explizit anzugeben braucht. Es genügt, ein Orakel (eine Subroutine) zu haben, welches bestimmt, ob ein gegebener Vektor zulässig ist und, wenn nicht, eine verletzte Nebenbedingung liefert. Für die ELLIPSOIDMETHODE werden wir dies in Abschnitt 4.6 noch ausführlich besprechen, da daraus folgt, dass viele kombinatorische Optimierungsprobleme polynomiell lösbar sind; für manche dieser Probleme ist dies in der Tat der einzige bekannte Weg, um ihre polynomielle Lösbarkeit zu beweisen. Aus diesem Grunde widmen wir uns in diesem Buch der ELLIPSOIDMETHODE, aber nicht den Innere-Punkte-Algorithmen.

Eine Vorbedingung für polynomielle Algorithmen ist die Existenz einer optimalen Lösung, die eine binäre Repräsentation hat, deren Länge durch ein Polynom in der Inputgröße beschränkt ist. In Abschnitt 4.1 werden wir zeigen, dass diese Bedingung für LINEARE OPTIMIERUNG erfüllt ist. In den Abschnitten 4.2 und 4.3 werden wir einige später benötigte Algorithmen zusammenstellen, darunter die bekannte Gauß-Elimination für die Lösung von linearen Gleichungssystemen.

4.1 Die Größe von Ecken und Seitenflächen

Vektoren und Matrizen sind Instanzen der LINEAREN OPTIMIERUNG. Da kein streng polynomieller Algorithmus für die LINEARE OPTIMIERUNG bekannt ist, müssen wir uns bei der Untersuchung der Laufzeiten von Algorithmen auf rationale

Instanzen beschränken. Wir setzen voraus, dass alle Zahlen binär kodiert sind. Um die Größe (Anzahl der Bits) dieser Darstellung abschätzen zu können, definieren wir $\text{size}(n) := 1 + \lceil \log(|n|+1) \rceil$ für ganze Zahlen $n \in \mathbb{Z}$ und $\text{size}(r) := \text{size}(p) + \text{size}(q)$ für rationale Zahlen $r = \frac{p}{q}$ mit p und q teilerfremd (d. h. der größte gemeinsame Teiler von p und q ist 1). Für Vektoren $x = (x_1, \dots, x_n) \in \mathbb{Q}^n$ speichern wir die Komponenten und haben dann $\text{size}(x) := n + \text{size}(x_1) + \dots + \text{size}(x_n)$. Für eine Matrix $A \in \mathbb{Q}^{m \times n}$ mit den Elementen a_{ij} haben wir $\text{size}(A) := mn + \sum_{i,j} \text{size}(a_{ij})$.

Natürlich sind diese Werte recht willkürlich gewählt, aber wir erinnern daran, dass es hier nicht auf konstante Faktoren ankommt. Für polynomielle Algorithmen ist es wichtig, dass die Größen von Zahlen bei einfachen arithmetischen Operationen nicht allzu sehr wachsen. Wir haben folgendes Resultat:

Proposition 4.1. *Für rationale Zahlen r_1, \dots, r_n gilt*

$$\begin{aligned}\text{size}(r_1 \cdots r_n) &\leq \text{size}(r_1) + \cdots + \text{size}(r_n); \\ \text{size}(r_1 + \cdots + r_n) &\leq 2(\text{size}(r_1) + \cdots + \text{size}(r_n)).\end{aligned}$$

Beweis: Für ganze Zahlen s_1, \dots, s_n haben wir offensichtlich $\text{size}(s_1 \cdots s_n) \leq \text{size}(s_1) + \cdots + \text{size}(s_n)$ und $\text{size}(s_1 + \cdots + s_n) \leq \text{size}(s_1) + \cdots + \text{size}(s_n)$.

Sei nun $r_i = \frac{p_i}{q_i}$, wobei p_i und q_i , ($i = 1, \dots, n$), nichtverschwindende ganze Zahlen sind. Dann gilt $\text{size}(r_1 \cdots r_n) \leq \text{size}(p_1 \cdots p_n) + \text{size}(q_1 \cdots q_n) \leq \text{size}(r_1) + \cdots + \text{size}(r_n)$.

Für die zweite Aussage beachte man, dass der Nenner $q_1 \cdots q_n$ höchstens die Größe $\text{size}(q_1) + \cdots + \text{size}(q_n)$ hat. Der Zähler ist die Summe der Zahlen $q_1 \cdots q_{i-1} p_i q_{i+1} \cdots q_n$ ($i = 1, \dots, n$), also ist sein Betrag höchstens gleich $(|p_1| + \cdots + |p_n|)|q_1 \cdots q_n|$. Damit ist die Größe des Zählers höchstens gleich $\text{size}(r_1) + \cdots + \text{size}(r_n)$. \square

Aus dem ersten Teil von Proposition 4.1 folgt ferner, dass wir oft o. B. d. A. annehmen können, dass alle Zahlen in einer gegebenen Instanz eines Problems ganzzahlig sind, da wir sonst jede von ihnen mit dem Produkt aller Nenner multiplizieren können. Für die Addition und das innere Produkt von Vektoren gilt:

Proposition 4.2. *Für rationale Vektoren $x, y \in \mathbb{Q}^n$ gilt*

$$\begin{aligned}\text{size}(x + y) &\leq 2(\text{size}(x) + \text{size}(y)); \\ \text{size}(x^\top y) &\leq 2(\text{size}(x) + \text{size}(y)).\end{aligned}$$

Beweis: Mit Proposition 4.1 haben wir $\text{size}(x + y) = n + \sum_{i=1}^n \text{size}(x_i + y_i) \leq n + 2 \sum_{i=1}^n \text{size}(x_i) + 2 \sum_{i=1}^n \text{size}(y_i) = 2(\text{size}(x) + \text{size}(y)) - 3n$ und $\text{size}(x^\top y) = \text{size}(\sum_{i=1}^n x_i y_i) \leq 2 \sum_{i=1}^n \text{size}(x_i y_i) \leq 2 \sum_{i=1}^n \text{size}(x_i) + 2 \sum_{i=1}^n \text{size}(y_i) = 2(\text{size}(x) + \text{size}(y)) - 4n$. \square

Auch bei komplizierteren numerischen Operationen wachsen die betroffenen Zahlen nicht schnell. Bekanntlich wird die Determinante einer Matrix $A = (a_{ij})_{1 \leq i,j \leq n}$ durch die Formel

$$\det A := \sum_{\pi \in S_n} \text{sgn}(\pi) \prod_{i=1}^n a_{i,\pi(i)} \quad (4.1)$$

definiert, wobei S_n die Menge der Permutationen von $\{1, \dots, n\}$ und $\text{sgn}(\pi)$ das Signum einer Permutation π ist: $\text{sgn}(\pi)$ ist gleich 1, falls π aus der identischen Permutation mittels einer geraden Anzahl von Transpositionen hervorgeht, sonst gleich -1 .

Proposition 4.3. *Für jede rationale quadratische Matrix $A \in \mathbb{Q}^{m \times n}$ gilt $\text{size}(\det A) \leq 2 \text{size}(A)$.*

Beweis: Es seien die $a_{ij} = \frac{p_{ij}}{q_{ij}}$, wobei p_{ij} und q_{ij} teilerfremde ganze Zahlen sind. Ferner sei $\det A = \frac{p}{q}$, wobei p und q teilerfremde ganze Zahlen sind. Dann folgt $|\det A| \leq \prod_{i,j} (|p_{ij}| + 1)$ und $|q| \leq \prod_{i,j} |q_{ij}|$. Also ist $\text{size}(q) \leq \text{size}(A)$ und mit $|p| = |\det A||q| \leq \prod_{i,j} (|p_{ij}| + 1)|q_{ij}|$ folgt

$$\text{size}(p) \leq \sum_{i,j} (\text{size}(p_{ij}) + 1 + \text{size}(q_{ij})) = \text{size}(A).$$

□

Mit diesem Resultat können wir den folgenden Satz beweisen:

Satz 4.4. *Angenommen, das rationale LP $\max\{cx : Ax \leq b\}$ hat eine optimale Lösung. Dann hat es auch eine optimale Lösung x mit $\text{size}(x) \leq 4n(\text{size}(A) + \text{size}(b))$ und mit Komponenten der Größe kleiner oder gleich $4(\text{size}(A) + \text{size}(b))$. Gilt $b = e_i$ oder $b = -e_i$ für einen geeigneten Einheitsvektor e_i , dann gibt es eine nichtsinguläre Untermatrix A' von A und eine optimale Lösung x mit $\text{size}(x) \leq 4n \text{size}(A')$.*

Beweis: Nach Korollar 3.5 folgt, dass das Maximum in einer Seitenfläche F von $\{x : Ax \leq b\}$ angenommen wird. Sei $F' \subseteq F$ eine minimale Seitenfläche. Nach Proposition 3.9 folgt $F' = \{x : A'x = b'\}$ für ein passendes Teilsystem $A'x \leq b'$ von $Ax \leq b$. O. B. d. A. können wir annehmen, dass die Zeilen von A' linear unabhängig sind. Nun nehmen wir eine maximale Menge linear unabhängiger Spalten (die eine Untermatrix A'' bilden) und setzen alle anderen Komponenten gleich 0. Dann folgt, dass $x = (A'')^{-1}b'$, mit den restlichen Komponenten gleich 0, eine optimale Lösung unseres LP ist. Nach der Cramerschen Regel werden die Komponenten von x durch $x_j = \frac{\det A'''}{\det A''}$ gegeben, wobei A''' aus A'' mittels Ersetzen der j -en Spalte durch b' hervorgeht. Mit Proposition 4.3 folgt $\text{size}(x) \leq n + 2n(\text{size}(A''') + \text{size}(A'')) \leq 4n(\text{size}(A'') + \text{size}(b''))$. Ist $b = \pm e_i$, so ist $|\det(A'')|$ gleich dem Betrag einer Unterdeterminante von A'' . □

Die Kodierungslänge der Seitenflächen eines durch seine Ecken gegebenen Polytops kann wie folgt abgeschätzt werden:

Lemma 4.5. *Sei $P \subseteq \mathbb{R}^n$ ein rationales Polytop und $T \in \mathbb{N}$ mit $\text{size}(x) \leq T$ für jede Ecke x . Dann folgt $P = \{x : Ax \leq b\}$ für ein geeignetes lineares Ungleichungssystem $Ax \leq b$, in welchem jede Ungleichung $ax \leq \beta$ die Bedingung $\text{size}(a) + \text{size}(\beta) \leq 75n^2T$ erfüllt.*

Beweis: Zunächst nehmen wir an, dass P volldimensional ist. Sei $F = \{x \in P : ax = \beta\}$ eine Facette von P , wobei $P \subseteq \{x : ax \leq \beta\}$.

Seien y_1, \dots, y_t die Ecken von F (nach Proposition 3.6 sind sie auch Ecken von P). Sei c die Lösung von $Mc = e_1$, wobei M eine $t \times n$ -Matrix ist, deren i -te Zeile $y_i - y_1$ ($i = 2, \dots, t$) ist und deren erste Zeile ein von den restlichen Zeilen linear unabhängiger Einheitsvektor ist. Beachte, dass $\text{rank}(M) = n$ (weil $\dim F = n - 1$). Also folgt $c^\top = \kappa a$ für ein $\kappa \in \mathbb{R} \setminus \{0\}$.

Mit Satz 4.4 folgt $\text{size}(c) \leq 4n \text{size}(M')$, wobei M' eine nichtsinguläre $n \times n$ -Untermatrix von M ist. Nach Proposition 4.2 folgt $\text{size}(M') \leq 4nT$ und $\text{size}(c^\top y_1) \leq 2(\text{size}(c) + \text{size}(y_1))$. Also erfüllt die Ungleichung $c^\top x \leq \delta$ (oder $c^\top x \geq \delta$, falls $\kappa < 0$), wobei $\delta := c^\top y_1 = \kappa\beta$, die Bedingung $\text{size}(c) + \text{size}(\delta) \leq 3\text{size}(c) + 2T \leq 48n^2T + 2T \leq 50n^2T$. Die Menge dieser Ungleichungen für alle Facetten F ergibt eine Beschreibung von P .

Ist $P = \emptyset$, so ist die Aussage trivial, also nehmen wir nun an, dass P weder volldimensional noch leer ist. Sei V die Menge der Ecken von P . Für $s = (s_1, \dots, s_n) \in \{-1, 1\}^n$ bezeichnen wir mit P_s die konvexe Hülle von $V \cup \{x + s_i t_i e_i : x \in V, i = 1, \dots, n\}$, wobei $t_i = |x_i|$, falls $x_i \neq 0$, und $t_i = 1$, falls $x_i = 0$ (e_i ist wieder der i -te Einheitsvektor). Jedes P_s ist ein volldimensionales Polytop (Satz 3.31) und die Größe jeder seiner Ecken ist höchstens $T+1$ (siehe Korollar 3.32). Nach Obigem kann P_s durch Ungleichungen der Größe kleiner oder gleich $50n^2(T+1) \leq 75n^2T$ beschrieben werden (unter Benutzung von $T \geq 2$). Da $P = \bigcap_{s \in \{-1, 1\}^n} P_s$, folgt der Beweis. \square

4.2 Kettenbrüche

Wenn wir von den in einem bestimmten Algorithmus auftretenden Zahlen behaupten, sie wachsen nicht allzu schnell, so nehmen wir oft an, dass für jede rationale Zahl $\frac{p}{q}$ der Zähler p und der Nenner q teilerfremd sind. Diese Annahme ist kein Problem, wenn sich der größte gemeinsame Teiler zweier natürlicher Zahlen leicht bestimmen lässt. Dazu benutzt man einen der ältesten Algorithmen überhaupt:

EUKLIDISCHER ALGORITHMUS

Input: Zwei natürliche Zahlen p und q .

Output: Der größte gemeinsame Teiler d von p und q ,
d. h. $\frac{p}{d}$ und $\frac{q}{d}$ sind teilerfremde Zahlen.

- ① **While** $p > 0$ und $q > 0$ **do:**
 - If $p < q$ **then** setze $q := q - \lfloor \frac{q}{p} \rfloor p$ **else** setze $p := p - \lfloor \frac{p}{q} \rfloor q$.
- ② **Return** $d := \max\{p, q\}$.

Satz 4.6. Der EUKLIDISCHE ALGORITHMUS arbeitet korrekt. Die Anzahl der Iterationen ist höchstens $\text{size}(p) + \text{size}(q)$.

Beweis: Die Korrektheit folgt aus der Tatsache, dass die Menge der gemeinsamen Teiler von p und q sich während des Ablaufs des Algorithmus nicht verändert bis eine der Zahlen verschwindet. In jeder Iteration wird entweder p oder q um wenigstens den Faktor 2 reduziert, also terminiert der Algorithmus nach höchstens $\log p + \log q + 1$ Iterationen. \square

Da keine der in den Zwischenstufen auftauchenden Zahlen größer als p oder q ist, folgt, dass der Euklidische Algorithmus ein polynomieller Algorithmus ist.

Ein ähnlicher Algorithmus ist die sogenannte KETTENBRUCH-ERWEITERUNG. Mithilfe dieser kann jede Zahl durch eine rationale Zahl approximiert werden, deren Nenner nicht allzu groß ist. Für jede positive reelle Zahl x definieren wir $x_0 := x$ und $x_{i+1} := \frac{1}{x_i - \lfloor x_i \rfloor}$ für $i = 1, 2, \dots$, bis $x_k \in \mathbb{N}$ für ein gewisses k . Dann haben wir

$$x = x_0 = \lfloor x_0 \rfloor + \frac{1}{x_1} = \lfloor x_0 \rfloor + \frac{1}{\lfloor x_1 \rfloor + \frac{1}{x_2}} = \lfloor x_0 \rfloor + \frac{1}{\lfloor x_1 \rfloor + \frac{1}{\lfloor x_2 \rfloor + \frac{1}{x_3}}} = \dots$$

Wir behaupten nun, dass diese Folge genau dann endlich ist, wenn x rational ist. Die eine Richtung folgt sofort aus der Tatsache, dass x_{i+1} genau dann rational ist, wenn x_i rational ist. Die andere Richtung folgt auch leicht: Ist $x = \frac{p}{q}$, so ist das obige Verfahren äquivalent zum EUKLIDISCHEN ALGORITHMUS angewendet auf p und q . Dies zeigt ferner, dass die obige (endliche) Folge x_1, x_2, \dots, x_k für eine rationale Zahl $\frac{p}{q}$ mit $p, q > 0$ in polynomieller Zeit berechnet werden kann. Der folgende Algorithmus ist fast identisch mit dem EUKLIDISCHEN ALGORITHMUS bis auf die Berechnung der Zahlen g_i und h_i ; wir werden weiter unten beweisen, dass die Folge $\left(\frac{g_i}{h_i}\right)_{i \in \mathbb{N}}$ gegen x konvergiert.

KETTENBRUCH-ERWEITERUNG

Input: Natürliche Zahlen p und q (sei $x := \frac{p}{q}$).

Output: Die Folge $\left(x_i = \frac{p_i}{q_i}\right)_{i=0,1,\dots}$ mit $x_0 = \frac{p}{q}$ und $x_{i+1} := \frac{1}{x_i - \lfloor x_i \rfloor}$.

- ① Setze $i := 0$, $p_0 := p$ und $q_0 := q$.
Setze $g_{-2} := 0$, $g_{-1} := 1$, $h_{-2} := 1$ und $h_{-1} := 0$.
- ② **While** $q_i \neq 0$ **do**:
 - Setze $a_i := \lfloor \frac{p_i}{q_i} \rfloor$.
 - Setze $g_i := a_i g_{i-1} + g_{i-2}$.
 - Setze $h_i := a_i h_{i-1} + h_{i-2}$.
 - Setze $q_{i+1} := p_i - a_i q_i$.
 - Setze $p_{i+1} := q_i$.
 - Setze $i := i + 1$.

Wir behaupten, dass die Folge $\frac{g_i}{h_i}$ gute Approximationen für x liefert. Bevor wir dies beweisen können, benötigen wir noch einige vorbereitende Resultate:

Proposition 4.7. Die folgenden Aussagen gelten für alle Iterationen i im obigen Algorithmus:

- (a) $a_i \geq 1$ (außer womöglich für $i = 0$) und $h_i \geq h_{i-1}$.
- (b) $g_{i-1}h_i - g_ih_{i-1} = (-1)^i$.
- (c) $\frac{p_i g_{i-1} + q_i g_{i-2}}{p_i h_{i-1} + q_i h_{i-2}} = x$.
- (d) $\frac{g_i}{h_i} \leq x$, falls i gerade, und $\frac{g_i}{h_i} \geq x$, falls i ungerade ist.

Beweis: (a) ist offensichtlich. (b) folgt leicht mit Induktion über i : Für $i = 0$ haben wir $g_{i-1}h_i - g_ih_{i-1} = g_{-1}h_0 = 1$ und für $i \geq 1$ haben wir

$$\begin{aligned} g_{i-1}h_i - g_ih_{i-1} &= g_{i-1}(a_i h_{i-1} + h_{i-2}) - h_{i-1}(a_i g_{i-1} + g_{i-2}) \\ &= g_{i-1}h_{i-2} - h_{i-1}g_{i-2}. \end{aligned}$$

(c) wird auch mittels Induktion bewiesen: Für $i = 0$ haben wir

$$\frac{p_i g_{i-1} + q_i g_{i-2}}{p_i h_{i-1} + q_i h_{i-2}} = \frac{p_i \cdot 1 + 0}{0 + q_i \cdot 1} = x$$

und für $i \geq 1$ haben wir

$$\begin{aligned} \frac{p_i g_{i-1} + q_i g_{i-2}}{p_i h_{i-1} + q_i h_{i-2}} &= \frac{q_{i-1}(a_{i-1}g_{i-2} + g_{i-3}) + (p_{i-1} - a_{i-1}q_{i-1})g_{i-2}}{q_{i-1}(a_{i-1}h_{i-2} + h_{i-3}) + (p_{i-1} - a_{i-1}q_{i-1})h_{i-2}} \\ &= \frac{q_{i-1}g_{i-3} + p_{i-1}g_{i-2}}{q_{i-1}h_{i-3} + p_{i-1}h_{i-2}}. \end{aligned}$$

Abschließend beweisen wir nun (d). Es gilt $\frac{g_{-2}}{h_{-2}} = 0 < x < \infty = \frac{g_{-1}}{h_{-1}}$ und wir benutzen wieder Induktion über i . Der Induktionsschritt folgt leicht aus den zwei Tatsachen, dass die Funktion $f(\alpha) := \frac{\alpha g_{i-1} + g_{i-2}}{\alpha h_{i-1} + h_{i-2}}$ für $\alpha > 0$ monoton ist und dass nach (c) $f\left(\frac{p_i}{q_i}\right) = x$ ist. \square

Satz 4.8. (Khinchine [1956]) Gegeben sei eine rationale Zahl α und eine natürliche Zahl n . Dann kann man eine rationale Zahl β mit Nenner kleiner oder gleich n , so dass $|\alpha - \beta|$ minimal ist, in polynomieller Zeit (polynomiell in $\text{size}(n) + \text{size}(\alpha)$) bestimmen.

Beweis: Wir wenden die KETTENBRUCH-ERWEITERUNG auf $x := \alpha$ an. Termiert der Algorithmus mit $q_i = 0$ und $h_{i-1} \leq n$, so können wir nach Proposition 4.7(c) $\beta = \frac{g_{i-1}}{h_{i-1}} = \alpha$ setzen. Andernfalls sei i der letzte Index mit $h_i \leq n$ und t die größte ganze Zahl mit $th_i + h_{i-1} \leq n$ (siehe Proposition 4.7(a)). Da $a_{i+1}h_i + h_{i-1} = h_{i+1} > n$, folgt $t < a_{i+1}$. Wir behaupten, dass

$$y := \frac{g_i}{h_i} \quad \text{oder} \quad z := \frac{tg_i + g_{i-1}}{th_i + h_{i-1}}$$

eine optimale Lösung ist. Für beide Zahlen ist der Nenner höchstens n .

Ist i gerade, so folgt $y \leq x < z$ nach Proposition 4.7(d). Analog folgt $y \geq x > z$ für i ungerade. Wir zeigen nun, dass für jede rationale Zahl $\frac{p}{q}$ zwischen y und z der Nenner größer als n ist.

Mit Proposition 4.7(b) folgt

$$|z - y| = \frac{|h_i g_{i-1} - h_{i-1} g_i|}{h_i(t h_i + h_{i-1})} = \frac{1}{h_i(t h_i + h_{i-1})}.$$

Andererseits gilt

$$|z - y| = \left| z - \frac{p}{q} \right| + \left| \frac{p}{q} - y \right| \geq \frac{1}{(t h_i + h_{i-1}) q} + \frac{1}{h_i q} = \frac{h_{i-1} + (t+1) h_i}{q h_i (t h_i + h_{i-1})},$$

also ist $q \geq h_{i-1} + (t+1) h_i > n$. \square

Der obige Beweis stammt aus Grötschel, Lovász und Schrijver [1988], wo auch weitere wichtige Verallgemeinerungen gezeigt werden.

4.3 Gauß-Elimination

Der wichtigste Algorithmus in der linearen Algebra ist die sogenannte Gauß-Elimination. Sie wurde von Gauß gebraucht, war aber schon lange Zeit vorher bekannt (siehe Schrijver [1986] für historische Bemerkungen). Die Gauß-Elimination wird zur Bestimmung des Rangs einer Matrix, zur Berechnung von Determinanten und zur Lösung linearer Gleichungssysteme verwendet. Ferner erscheint sie sehr oft als Subroutine in linearen Optimierungsalgorithmen; z. B. in ① des SIMPLEX-ALGORITHMUS.

Gegeben sei eine Matrix $A \in \mathbb{Q}^{m \times n}$. Unser Algorithmus für die Gauß-Elimination arbeitet mit einer erweiterten Matrix $Z = (B \ C) \in \mathbb{Q}^{m \times (n+m)}$; anfänglich sind $B = A$ und $C = I$. Der Algorithmus transformiert B in die Form $\begin{pmatrix} I & R \\ 0 & 0 \end{pmatrix}$ mittels der folgenden elementaren Operationen: Vertauschung zweier Zeilen bzw. Spalten, Addition des Vielfachen einer Zeile zu einer anderen Zeile und, im letzten Schritt, Multiplikation von Zeilen mit nichtverschwindenden Konstanten. In jeder Iteration wird C entsprechend verändert, so dass die Eigenschaft $C\tilde{A} = B$ durchweg erhalten bleibt, wobei \tilde{A} aus A durch Vertauschung zweier Zeilen bzw. Spalten hervorgeht.

Der erste Teil des Algorithmus, bestehend aus ② und ③, transformiert B in eine obere Dreiecksmatrix. Betrachten wir z. B. die Matrix Z nach zwei Iterationen, so hat sie die Form

$$\left(\begin{array}{cccccc|cccccc} z_{11} \neq 0 & z_{12} & z_{13} & \cdots & z_{1n} & 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & z_{22} \neq 0 & z_{23} & \cdots & z_{2n} & z_{2,n+1} & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & z_{33} & \cdots & z_{3n} & z_{3,n+1} & z_{3,n+2} & 1 & 0 & \cdots & 0 \\ \vdots & \cdot & \cdot & & \cdot & \cdot & \cdot & 0 & & \ddots & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot & \cdot & & I & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot & \cdot & & 0 & \cdot \\ 0 & 0 & z_{m3} & \cdots & z_{mn} & z_{m,n+1} & z_{m,n+2} & 0 & \cdots & 0 & 1 \end{array} \right).$$

Ist $z_{33} \neq 0$, so besteht der nächste Schritt bloß aus der Subtraktion von $\frac{z_{i3}}{z_{33}}$ mal der dritten Zeile von der i -ten Zeile, mit $i = 4, \dots, m$. Ist andererseits $z_{33} = 0$, so tauschen wir zunächst die dritte Zeile mit einer anderen Zeile und/oder die dritte Spalte mit einer anderen Spalte. Tauschen wir zwei Zeilen, so müssen wir in C auch die entsprechenden Spalten tauschen, damit die Eigenschaft $C\tilde{A} = B$ erhalten bleibt. Um jederzeit \tilde{A} zur Verfügung zu haben, speichern wir die Zeilen- und Spaltenvertauschungen mittels der Variablen $row(i)$, $i = 1, \dots, m$ und $col(j)$, $j = 1, \dots, n$. Dann folgt $\tilde{A} = (A_{row(i), col(j)})_{i \in \{1, \dots, m\}, j \in \{1, \dots, n\}}$.

Der zweite Teil des Algorithmus, bestehend aus ④ und ⑤, ist einfacher, da jetzt keine Zeilen- oder Spaltenvertauschungen mehr stattfinden.

GAUSS-ELIMINATION

Input: Eine Matrix $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$.

Output: Ihr Rang r , eine maximale nichtsinguläre Untermatrix $A' = (a_{row(i), col(j)})_{i, j \in \{1, \dots, r\}}$ von A , ihre Determinante $d = \det A'$ und ihre Inverse $(A')^{-1} = (z_{i, n+j})_{i, j \in \{1, \dots, r\}}$.

- ① Setze $r := 0$ und $d := 1$.
Setze $z_{ij} := a_{ij}$, $row(i) := i$ und $col(j) := j$ ($i = 1, \dots, m$, $j = 1, \dots, n$).
Setze $z_{i, n+j} := 0$ und $z_{i, n+i} := 1$ für $1 \leq i, j \leq m$, $i \neq j$.
- ② Sei $p \in \{r+1, \dots, m\}$ und $q \in \{r+1, \dots, n\}$ mit $z_{pq} \neq 0$. **If** es gibt kein solches Paar p und q **then go to** ④.
Setze $r := r + 1$.
If $p \neq r$ **then** vertausche z_{pj} und z_{rj} ($j = 1, \dots, n+m$), vertausche $z_{i, n+p}$ und $z_{i, n+r}$ ($i = 1, \dots, m$) und vertausche $row(p)$ und $row(r)$.
If $q \neq r$ **then** vertausche z_{iq} und z_{ir} ($i = 1, \dots, m$) und vertausche $col(q)$ und $col(r)$.
- ③ Setze $d := d \cdot z_{rr}$.
For $i := r+1$ **to** m **do**:
 Setze $\alpha := \frac{z_{ir}}{z_{rr}}$.
 For $j := r$ **to** $n+r$ **do**: $z_{ij} := z_{ij} - \alpha z_{rj}$.
Go to ②.
- ④ **For** $k := r$ **down to** 2 **do**:
 For $i := 1$ **to** $k-1$ **do**:
 Setze $\alpha := \frac{z_{ik}}{z_{kk}}$.
 For $j := k$ **to** $n+r$ **do** $z_{ij} := z_{ij} - \alpha z_{kj}$.
- ⑤ **For** $k := 1$ **to** r **do**:
 Setze $\alpha := \frac{1}{z_{kk}}$.
 For $j := 1$ **to** $n+r$ **do** $z_{kj} := \alpha z_{kj}$.

Satz 4.9. Die GAUSS-ELIMINATION arbeitet korrekt und endet nach $O(mnr)$ Schritten.

Beweis: Beachte zunächst, dass wir jedes Mal vor ② $z_{ii} \neq 0$ für alle $i \in \{1, \dots, r\}$ und $z_{ij} = 0$ für alle $j \in \{1, \dots, r\}$ und $i \in \{j+1, \dots, m\}$ haben. Somit

gilt

$$\det((z_{ij})_{i,j \in \{1,2,\dots,r\}}) = z_{11}z_{22} \cdots z_{rr} = d \neq 0.$$

Da die Addition eines Vielfachen einer Zeile zu einer anderen Zeile einer quadratischen Matrix den Wert der Determinante nicht verändert (diese bekannte Eigenschaft folgt sofort aus Definition (4.1)), haben wir

$$\det((z_{ij})_{i,j \in \{1,2,\dots,r\}}) = \det((a_{row(i),col(j)})_{i,j \in \{1,2,\dots,r\}})$$

zu jedem Zeitpunkt vor ⑤, also wird die Determinante d korrekt berechnet. Es ist A' eine nichtsinguläre $r \times r$ -Untermatrix von A . Da die Matrix $(z_{ij})_{i \in \{1,\dots,m\}, j \in \{1,\dots,n\}}$ am Ende Rang r hat und die Operationen den Rang nicht veränderten, folgt, dass A auch Rang r hat.

Ferner gilt durchweg, dass $\sum_{j=1}^m z_{i,n+j} a_{row(j),col(k)} = z_{ik}$ für alle $i \in \{1, \dots, m\}$ und $k \in \{1, \dots, n\}$ (d.h. $C\tilde{A} = B$ in der oben eingeführten Notation). (Beachte: Für $j = r+1, \dots, m$ haben wir zu jedem Zeitpunkt $z_{j,n+j} = 1$ und $z_{i,n+j} = 0$ für $i \neq j$.) Da $(z_{ij})_{i,j \in \{1,2,\dots,r\}}$ am Ende die Einheitsmatrix ist, wurde $(A')^{-1}$ auch korrekt berechnet. Die Anzahl der Schritte ist offensichtlich $O(rmn + r^2(n+r)) = O(mnr)$. \square

Um zu beweisen, dass die GAUSS-ELIMINATION ein polynomieller Algorithmus ist, müssen wir zeigen, dass alle vorkommenden Zahlen durch die Inputgröße polynomiell beschränkt sind. Dies ist nicht trivial, kann aber bewiesen werden:

Satz 4.10. (Edmonds [1967]) *Die GAUSS-ELIMINATION ist ein polynomieller Algorithmus. Jede im Laufe des Algorithmus vorkommende Zahl kann mit $O(m(m+n)\text{size}(A))$ Bits gespeichert werden.*

Beweis: Zunächst zeigen wir: In den Schritten ② und ③ sind alle Zahlen gleich 0 oder 1, oder sie sind Quotienten von Unterdeterminanten von A . Beachte als erstes, dass die Elemente z_{ij} mit $i \leq r$ oder $j \leq r$ nicht weiter verändert werden. Die Elemente z_{ij} mit $j > n+r$ sind gleich 0 (falls $j \neq n+i$) oder gleich 1 (falls $j = n+i$). Ferner haben wir für alle $s \in \{r+1, \dots, m\}$ und $t \in \{r+1, \dots, n+m\}$

$$|z_{st}| = \left| \frac{\det((z_{ij})_{i \in \{1,2,\dots,r,s\}, j \in \{1,2,\dots,r,t\}})}{\det((z_{ij})_{i,j \in \{1,2,\dots,r\}})} \right|.$$

(Dies folgt mittels Entwicklung der Determinante $\det((z_{ij})_{i \in \{1,2,\dots,r,s\}, j \in \{1,2,\dots,r,t\}})$ nach der letzten Zeile, da $z_{sj} = 0$ für alle $s \in \{r+1, \dots, m\}$ und $j \in \{1, \dots, r\}$.)

Im Beweis von Satz 4.9 haben wir bereits gesehen, dass

$$\det((z_{ij})_{i,j \in \{1,2,\dots,r\}}) = \det((a_{row(i),col(j)})_{i,j \in \{1,2,\dots,r\}}),$$

weil die Addition eines Vielfachen einer Zeile zu einer anderen Zeile einer quadratischen Matrix den Wert der Determinante nicht verändert. Analog folgt

$$\det((z_{ij})_{i \in \{1,2,\dots,r,s\}, j \in \{1,2,\dots,r,t\}}) = \det((a_{row(i),col(j)})_{i \in \{1,2,\dots,r,s\}, j \in \{1,2,\dots,r,t\}})$$

für alle $s \in \{r+1, \dots, m\}$ und $t \in \{r+1, \dots, n\}$. Ferner folgt

$$|\det((z_{ij})_{i \in \{1, 2, \dots, r, s\}, j \in \{1, 2, \dots, r, n+t\}})| = |\det((a_{row(i), col(j)})_{i \in \{1, 2, \dots, r, s\} \setminus \{t\}, j \in \{1, 2, \dots, r\}})|$$

für alle $s \in \{r+1, \dots, m\}$ und $t \in \{1, \dots, r\}$ mittels der Entwicklung der linken Determinante (nach ①) nach der $(n+t)$ ten Spalte.

Daraus schließen wir: Zu jedem Zeitpunkt sind in ② und ③ alle Zahlen z_{ij} gleich 0 oder 1, oder sie sind Quotienten von Unterdeterminanten von A . Nach Proposition 4.3 folgt sodann, dass jede in ② und ③ vorkommende Zahl mit $O(\text{size}(A))$ Bits gespeichert werden kann.

Beachte schließlich, dass ④ zur nochmaligen Anwendung von ② und ③ äquivalent ist, wobei p und q geeignet gewählt werden müssen (Umkehrung der Reihenfolge der ersten r Zeilen und Spalten). Demnach kann jede in ④ vorkommende Zahl mit $O(\text{size}((z_{ij})_{i \in \{1, \dots, m\}, j \in \{1, \dots, m+n\}}))$, d. h. $O(m(m+n)\text{size}(A))$ Bits gespeichert werden.

Der einfachste Weg, um die Darstellungen der Zahlen z_{ij} klein genug zu halten, ist, zu gewährleisten, dass Zähler und Nenner dieser Zahlen stets teilerfremd sind. Dies erreicht man durch Anwendung des EUKLIDISCHEN ALGORITHMUS nach jeder Berechnung. Damit ergibt sich eine insgesamt polynomielle Laufzeit. \square

Man kann die GAUSS-ELIMINATION sogar leicht als streng polynomiellen Algorithmus implementieren (Aufgabe 4).

Also können wir in polynomieller Zeit prüfen, ob eine Menge von Vektoren linear unabhängig ist. Ferner können wir die Determinante und die Inverse einer nichtsingulären Matrix in polynomieller Zeit berechnen (die Vertauschung zweier Zeilen oder zweier Spalten ändert bloß das Vorzeichen der Determinante). Wir haben auch das

Korollar 4.11. Gegeben sei eine Matrix $A \in \mathbb{Q}^{m \times n}$ und ein Vektor $b \in \mathbb{Q}^m$. Dann können wir in polynomieller Zeit einen Vektor $x \in \mathbb{Q}^n$ mit $Ax = b$ bestimmen oder entscheiden, dass es keinen solchen Vektor gibt.

Beweis: Zunächst berechnen wir eine maximale nichtsinguläre Untermatrix $A' = (a_{row(i), col(j)})_{i, j \in \{1, \dots, r\}}$ von A und ihre Inverse $(A')^{-1} = (z_{i, n+j})_{i, j \in \{1, \dots, r\}}$ mittels GAUSS-ELIMINATION. Dann setzen wir $x_{col(j)} := \sum_{k=1}^r z_{j, n+k} b_{row(k)}$ für $j = 1, \dots, r$ und $x_k := 0$ für $k \notin \{col(1), \dots, col(r)\}$. Für $i = 1, \dots, r$ bekommen wir:

$$\begin{aligned} \sum_{j=1}^n a_{row(i), j} x_j &= \sum_{j=1}^r a_{row(i), col(j)} x_{col(j)} \\ &= \sum_{j=1}^r a_{row(i), col(j)} \sum_{k=1}^r z_{j, n+k} b_{row(k)} \\ &= \sum_{k=1}^r b_{row(k)} \sum_{j=1}^r a_{row(i), col(j)} z_{j, n+k} \\ &= b_{row(i)}. \end{aligned}$$

Da die anderen Zeilen von A , deren Indizes nicht in $\{row(1), \dots, row(r)\}$ liegen, Linearkombinationen der Zeilen mit diesen Indizes sind, gilt: entweder erfüllt x das lineare Gleichungssystem $Ax = b$ oder kein Vektor erfüllt es. \square

4.4 Die Ellipsoidmethode

In diesem Abschnitt werden wir die sogenannte Ellipsoidmethode beschreiben. Sie wurde von Iudin und Nemirovskii [1976] und von Shor [1977] für die nichtlineare Optimierung entwickelt. Khachiyan [1979] entdeckte, dass sie in einer modifizierten Form auch zur Lösung von LPs in polynomieller Zeit dienen konnte. Der größte Teil unserer Darstellung basiert auf (Grötschel, Lovász und Schrijver [1981]), (Bland, Goldfarb und Todd [1981]) und dem Buch von Grötschel, Lovász und Schrijver [1988], das auch zum weiteren Studium empfohlen wird.

Die grundlegende Idee der Ellipsoidmethode ist in groben Zügen die folgende: In Proposition 4.16 werden wir zeigen, dass es ausreicht, eine zulässige Lösung zu einem LP zu finden oder zu entscheiden, dass keine solche existiert. Wir beginnen mit einem Ellipsoid, von dem wir a priori wissen, dass es alle Basislösungen enthält (z. B. eine große Kugel). Bei jeder Iteration k prüfen wir, ob der Mittelpunkt x_k des aktuellen Ellipsoids eine zulässige Lösung ist. Falls nicht, nehmen wir eine x_k enthaltende Hyperebene mit der Eigenschaft, dass alle Lösungen auf einer Seite liegen. Damit haben wir ein alle Lösungen enthaltendes Halb-Ellipsoid. Nun nehmen wir das kleinste dieses Halb-Ellipsoid gänzlich enthaltende Ellipsoid und wiederholen den obigen Schritt.

Definition 4.12. Ein **Ellipsoid** ist eine Menge $E(A, x) = \{z \in \mathbb{R}^n : (z - x)^\top A^{-1}(z - x) \leq 1\}$, wobei A eine symmetrische positiv definite $n \times n$ -Matrix ist.

Beachte, dass $B(x, r) := E(r^2 I, x)$ (I bezeichnet die $n \times n$ Einheitsmatrix) die n -dimensionale euklidische Kugel mit Mittelpunkt x und Radius r ist.

Das Volumen eines Ellipsoids $E(A, x)$ ist bekanntlich

$$\text{volume}(E(A, x)) = \sqrt{\det A} \text{ volume}(B(0, 1))$$

(siehe Aufgabe 7). Gegeben sei ein Ellipsoid $E(A, x)$ und eine Hyperebene $\{z : az = ax\}$. Das kleinste Ellipsoid $E(A', x')$, welches das Halb-Ellipsoid $E' = \{z \in E(A, x) : az \geq ax\}$ enthält, heißt das Löwner-John-Ellipsoid von E' (siehe Abb. 4.1). Es kann mittels der folgenden Formeln berechnet werden:

$$\begin{aligned} A' &= \frac{n^2}{n^2 - 1} \left(A - \frac{2}{n+1} bb^\top \right), \\ x' &= x + \frac{1}{n+1} b, \\ b &= \frac{1}{\sqrt{a^\top A a}} A a. \end{aligned}$$

Das Wurzelzeichen in der Formel für b bereitet uns Schwierigkeiten. Da wir Rundungsfehler akzeptieren müssen, ist es notwendig, den Radius des nächsten

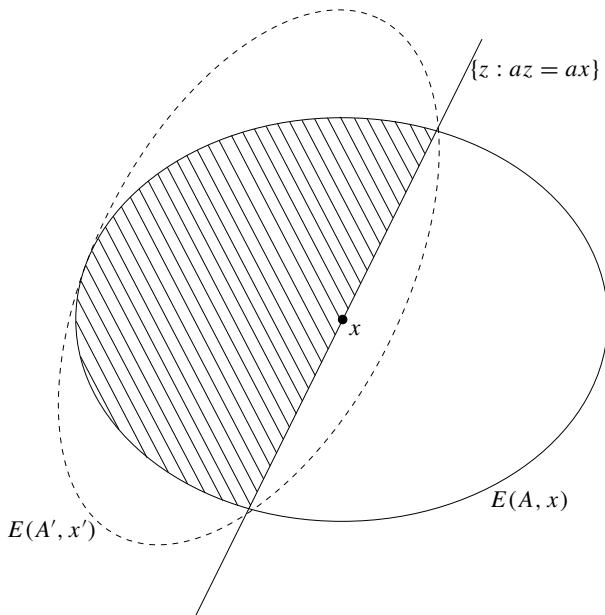


Abbildung 4.1.

Ellipsoide etwas zu vergrößern. Wir geben ein algorithmisches Schema an, welches diesem Problem Rechnung trägt:

ELLIPSOIDMETHODE

Input: Eine Zahl $n \in \mathbb{N}$, $n \geq 2$. Eine Zahl $N \in \mathbb{N}$. $x_0 \in \mathbb{Q}^n$ und $R \in \mathbb{Q}_+$, $R \geq 2$.

Output: Ein Ellipsoid $E(A_N, x_N)$.

- ① Setze $p := \lceil 6N + \log(9n^3) \rceil$.
Setze $A_0 := R^2 I$, wobei I die $n \times n$ Einheitsmatrix ist.
Setze $k := 0$.
- ② Wähle ein $a_k \in \mathbb{Q}^n \setminus \{0\}$.
- ③ Setze $b_k := \frac{1}{\sqrt{a_k^\top A_k a_k}} A_k a_k$.
Setze $x_{k+1} := x_{k+1}^* := x_k + \frac{1}{n+1} b_k$.
Setze $A_{k+1} := A_{k+1}^* := \frac{2n^2 + 3}{2n^2} \left(A_k - \frac{2}{n+1} b_k b_k^\top \right)$.
(Hier bedeutet \approx die Berechnung der Elemente bis auf p binäre Stellen hinter dem Komma, unter Sicherstellung, dass A_{k+1} symmetrisch wird).
- ④ Setze $k := k + 1$.
If $k < N$ **then go to** ② **else stop.**

Es wird also bei jeder der N Iterationen eine Approximation $E(A_{k+1}, x_{k+1})$ des kleinsten $E(A_k, x_k) \cap \{z : a_k z \geq a_k x_k\}$ enthaltenden Ellipsoids berechnet. Zwei Hauptfragen, wie man nämlich die a_k bekommt und N wählt, werden wir im nächsten Abschnitt aufgreifen. Zunächst werden wir einige Lemmata beweisen.

Es bezeichne $\|x\|$ die euklidische Norm des Vektors x und $\|A\| := \max\{\|Ax\| : \|x\| = 1\}$ die Norm der Matrix A . Für symmetrische Matrizen ist $\|A\|$ die größte Zahl unter den Beträgen der Eigenwerte von A und $\|A\| = \max\{x^\top Ax : \|x\| = 1\}$.

Das erste Lemma besagt, dass jedes $E_k := E(A_k, x_k)$ tatsächlich ein Ellipsoid ist und ferner, dass die Beträge der vorkommenden Zahlen kleiner als $R^2 2^N + 2^{\text{size}(x_0)}$ bleiben. Deshalb besteht jede Iteration der ELLIPSOIDMETHODE aus $O(n^2)$ Rechenschritten, wobei jeder dieser Schritte Zahlen mit $O(p + \text{size}(a_k) + \text{size}(R) + \text{size}(x_0))$ Bits benutzt.

Lemma 4.13. (Grötschel, Lovász und Schrijver [1981]) *Sei $k \in \{0, 1, \dots, N\}$. Dann ist A_k positiv definit und wir haben*

$$\|x_k\| \leq \|x_0\| + R2^k, \quad \|A_k\| \leq R^2 2^k \quad \text{und} \quad \|A_k^{-1}\| \leq R^{-2} 4^k.$$

Beweis: Wir benutzen Induktion über k . Für $k = 0$ sind alle Aussagen klar. Angenommen, sie gelten für ein $k \geq 0$. Eine einfache Berechnung ergibt

$$(A_{k+1}^*)^{-1} = \frac{2n^2}{2n^2 + 3} \left(A_k^{-1} + \frac{2}{n-1} \frac{a_k a_k^\top}{a_k^\top A_k a_k} \right). \quad (4.2)$$

Demnach ist $(A_{k+1}^*)^{-1}$ die Summe einer positiv definiten und einer positiv semidefiniten Matrix; also ist sie positiv definit. Damit ist auch A_{k+1}^* positiv definit.

Beachte: Für positiv semidefinite Matrizen A und B gilt $\|A\| \leq \|A + B\|$. Damit folgt

$$\|A_{k+1}^*\| = \frac{2n^2 + 3}{2n^2} \left\| A_k - \frac{2}{n+1} b_k b_k^\top \right\| \leq \frac{2n^2 + 3}{2n^2} \|A_k\| \leq \frac{11}{8} R^2 2^k.$$

Da die $n \times n$ Matrix mit lauter Einsen die Norm n hat, ist die Norm der Matrix $A_{k+1} - A_{k+1}^*$, deren Elemente sämtlich im Betrage kleiner oder gleich 2^{-p} sind, höchstens $n2^{-p}$. Daraus schließen wir

$$\|A_{k+1}\| \leq \|A_{k+1}^*\| + \|A_{k+1} - A_{k+1}^*\| \leq \frac{11}{8} R^2 2^k + n2^{-p} \leq R^2 2^{k+1}$$

(wobei wir die sehr grobe Abschätzung $2^{-p} \leq \frac{1}{n}$ benutzt haben).

Ein bekanntes Resultat der linearen Algebra besagt, dass es für jede symmetrische positiv definite $n \times n$ -Matrix A eine symmetrische positiv definite Matrix B mit $A = BB$ gibt. Schreiben wir $A_k = BB$ mit $B = B^\top$, so folgt

$$\|b_k\| = \frac{\|A_k a_k\|}{\sqrt{a_k^\top A_k a_k}} = \sqrt{\frac{a_k^\top A_k^2 a_k}{a_k^\top A_k a_k}} = \sqrt{\frac{(B a_k)^\top A_k (B a_k)}{(B a_k)^\top (B a_k)}} \leq \sqrt{\|A_k\|} \leq R 2^{k-1}.$$

Damit (zusammen mit der Induktionsvoraussetzung) bekommen wir

$$\begin{aligned} \|x_{k+1}\| &\leq \|x_k\| + \frac{1}{n+1} \|b_k\| + \|x_{k+1} - x_{k+1}^*\| \\ &\leq \|x_0\| + R2^k + \frac{1}{n+1} R2^{k-1} + \sqrt{n} 2^{-p} \leq \|x_0\| + R2^{k+1}. \end{aligned}$$

Mit (4.2) und $\|a_k a_k^\top\| = a_k^\top a_k$ berechnen wir nun

$$\begin{aligned} \left\| (A_{k+1}^*)^{-1} \right\| &\leq \frac{2n^2}{2n^2 + 3} \left(\|A_k^{-1}\| + \frac{2}{n-1} \frac{a_k^\top a_k}{a_k^\top A_k a_k} \right) \\ &= \frac{2n^2}{2n^2 + 3} \left(\|A_k^{-1}\| + \frac{2}{n-1} \frac{a_k^\top B A_k^{-1} B a_k}{a_k^\top B B a_k} \right) \\ &\leq \frac{2n^2}{2n^2 + 3} \left(\|A_k^{-1}\| + \frac{2}{n-1} \|A_k^{-1}\| \right) < \frac{n+1}{n-1} \|A_k^{-1}\| \\ &\leq 3R^{-2}4^k. \end{aligned} \quad (4.3)$$

Sei λ der kleinste Eigenwert von A_{k+1} und v ein Eigenvektor von λ mit $\|v\| = 1$. Schreiben wir $A_{k+1}^* = CC$ für eine symmetrische Matrix C , so folgt

$$\begin{aligned} \lambda &= v^\top A_{k+1} v = v^\top A_{k+1}^* v + v^\top (A_{k+1} - A_{k+1}^*) v \\ &= \frac{v^\top C C v}{v^\top C (A_{k+1}^*)^{-1} C v} + v^\top (A_{k+1} - A_{k+1}^*) v \\ &\geq \left\| (A_{k+1}^*)^{-1} \right\|^{-1} - \|A_{k+1} - A_{k+1}^*\| > \frac{1}{3} R^2 4^{-k} - n 2^{-p} \geq R^2 4^{-(k+1)}, \end{aligned}$$

wobei wir die Abschätzung $2^{-p} \leq \frac{1}{3n} 4^{-k}$ benutzt haben. Da $\lambda > 0$, ist A_{k+1} positiv definit. Ferner ist

$$\left\| (A_{k+1})^{-1} \right\| = \frac{1}{\lambda} \leq R^{-2} 4^{k+1}. \quad \square$$

Als Nächstes zeigen wir: In jeder Iteration enthält das Ellipsoid den Durchschnitt von E_0 und dem vorigen Halb-Ellipsoid:

Lemma 4.14. Für $k = 0, \dots, N-1$ haben wir $E_{k+1} \supseteq \{x \in E_k \cap E_0 : a_k x \geq a_k x_k\}$.

Beweis: Sei $x \in E_k \cap E_0$ mit $a_k x \geq a_k x_k$. Zunächst berechnen wir (mithilfe von (4.2))

$$\begin{aligned} &(x - x_{k+1}^*)^\top (A_{k+1}^*)^{-1} (x - x_{k+1}^*) \\ &= \frac{2n^2}{2n^2 + 3} \left(x - x_k - \frac{1}{n+1} b_k \right)^\top \left(A_k^{-1} + \frac{2}{n-1} \frac{a_k a_k^\top}{a_k^\top A_k a_k} \right) \left(x - x_k - \frac{1}{n+1} b_k \right) \\ &= \frac{2n^2}{2n^2 + 3} \left((x - x_k)^\top A_k^{-1} (x - x_k) + \frac{2}{n-1} (x - x_k)^\top \frac{a_k a_k^\top}{a_k^\top A_k a_k} (x - x_k) \right. \\ &\quad \left. + \frac{1}{(n+1)^2} \left(b_k^\top A_k^{-1} b_k + \frac{2}{n-1} \frac{b_k^\top a_k a_k^\top b_k}{a_k^\top A_k a_k} \right) \right. \\ &\quad \left. - \frac{2(x - x_k)^\top}{n+1} \left(A_k^{-1} b_k + \frac{2}{n-1} \frac{a_k a_k^\top b_k}{a_k^\top A_k a_k} \right) \right) \\ &= \frac{2n^2}{2n^2 + 3} \left((x - x_k)^\top A_k^{-1} (x - x_k) + \frac{2}{n-1} (x - x_k)^\top \frac{a_k a_k^\top}{a_k^\top A_k a_k} (x - x_k) + \right. \\ &\quad \left. \frac{1}{(n+1)^2} \left(1 + \frac{2}{n-1} \right) - \frac{2}{n+1} \frac{(x - x_k)^\top a_k}{\sqrt{a_k^\top A_k a_k}} \left(1 + \frac{2}{n-1} \right) \right). \end{aligned}$$

Da $x \in E_k$, haben wir $(x - x_k)^\top A_k^{-1}(x - x_k) \leq 1$. Mit der Abkürzung $t := \frac{a_k^\top(x - x_k)}{\sqrt{a_k^\top A_k a_k}}$ folgt

$$(x - x_{k+1}^*)^\top (A_{k+1}^*)^{-1}(x - x_{k+1}^*) \leq \frac{2n^2}{2n^2 + 3} \left(1 + \frac{2}{n-1}t^2 + \frac{1}{n^2-1} - \frac{2}{n-1}t \right).$$

Da $b_k^\top A_k^{-1} b_k = 1$ und $b_k^\top A_k^{-1}(x - x_k) = t$, folgt

$$\begin{aligned} 1 &\geq (x - x_k)^\top A_k^{-1}(x - x_k) \\ &= (x - x_k - tb_k)^\top A_k^{-1}(x - x_k - tb_k) + t^2 \\ &\geq t^2, \end{aligned}$$

weil A_k^{-1} positiv definit ist. Also (da $a_k x \geq a_k x_k$) haben wir $0 \leq t \leq 1$ und es folgt

$$(x - x_{k+1}^*)^\top (A_{k+1}^*)^{-1}(x - x_{k+1}^*) \leq \frac{2n^4}{2n^4 + n^2 - 3}.$$

Wir müssen nur noch den Rundungsfehler Z abschätzen:

$$\begin{aligned} Z &:= \left| (x - x_{k+1})^\top (A_{k+1})^{-1}(x - x_{k+1}) - (x - x_{k+1}^*)^\top (A_{k+1}^*)^{-1}(x - x_{k+1}^*) \right| \\ &\leq \left| (x - x_{k+1})^\top (A_{k+1})^{-1}(x_{k+1}^* - x_{k+1}) \right| \\ &\quad + \left| (x_{k+1}^* - x_{k+1})^\top (A_{k+1})^{-1}(x - x_{k+1}^*) \right| \\ &\quad + \left| (x - x_{k+1}^*)^\top \left((A_{k+1})^{-1} - (A_{k+1}^*)^{-1} \right) (x - x_{k+1}^*) \right| \\ &\leq \|x - x_{k+1}\| \|(A_{k+1})^{-1}\| \|x_{k+1}^* - x_{k+1}\| \\ &\quad + \|x_{k+1}^* - x_{k+1}\| \|(A_{k+1})^{-1}\| \|x - x_{k+1}^*\| \\ &\quad + \|x - x_{k+1}^*\|^2 \|(A_{k+1})^{-1}\| \|(A_{k+1}^*)^{-1}\| \|A_{k+1}^* - A_{k+1}\|. \end{aligned}$$

Mit Lemma 4.13 und da $x \in E_0$, gilt $\|x - x_{k+1}\| \leq \|x - x_0\| + \|x_{k+1} - x_0\| \leq R + R2^N$ und $\|x - x_{k+1}^*\| \leq \|x - x_{k+1}\| + \sqrt{n}2^{-p} \leq R2^{N+1}$. Mit (4.3) folgt dann

$$\begin{aligned} Z &\leq 2(R2^{N+1})(R^{-2}4^N)(\sqrt{n}2^{-p}) + (R^24^{N+1})(R^{-2}4^N)(3R^{-2}4^{N-1})(n2^{-p}) \\ &= 4R^{-1}2^{3N}\sqrt{n}2^{-p} + 3R^{-2}2^{6N}n2^{-p} \\ &\leq 2^{6N}n2^{-p} \\ &\leq \frac{1}{9n^2} \end{aligned}$$

nach der Definition von p . Insgesamt folgt somit

$$(x - x_{k+1})^\top (A_{k+1})^{-1}(x - x_{k+1}) \leq \frac{2n^4}{2n^4 + n^2 - 3} + \frac{1}{9n^2} \leq 1.$$

□

Die Volumen der Ellipsoide nehmen in jeder Iteration mit einem konstanten Faktor ab:

Lemma 4.15. Für $k = 0, \dots, N - 1$ haben wir $\frac{\text{volume}(E_{k+1})}{\text{volume}(E_k)} < e^{-\frac{1}{5n}}$.

Beweis: (Grötschel, Lovász und Schrijver [1988]) Wir schreiben

$$\frac{\text{volume}(E_{k+1})}{\text{volume}(E_k)} = \sqrt{\frac{\det A_{k+1}}{\det A_k}} = \sqrt{\frac{\det A_{k+1}^*}{\det A_k}} \sqrt{\frac{\det A_{k+1}}{\det A_{k+1}^*}}$$

und schätzen die beiden Faktoren separat ab. Beachte zunächst, dass

$$\frac{\det A_{k+1}^*}{\det A_k} = \left(\frac{2n^2 + 3}{2n^2} \right)^n \det \left(I - \frac{2}{n+1} \frac{a_k a_k^\top A_k}{a_k^\top A_k a_k} \right).$$

Die Matrix $\frac{a_k a_k^\top A_k}{a_k^\top A_k a_k}$ hat Rang 1 und der einzige nichtverschwindende Eigenwert ist 1 (mit Eigenvektor a_k). Da die Determinante gleich dem Produkt der Eigenwerte ist, schließen wir, dass

$$\frac{\det A_{k+1}^*}{\det A_k} = \left(\frac{2n^2 + 3}{2n^2} \right)^n \left(1 - \frac{2}{n+1} \right) < e^{\frac{3}{2n}} e^{-\frac{2}{n}} = e^{-\frac{1}{2n}},$$

wobei wir $1 + x \leq e^x$ für alle x und $\left(\frac{n-1}{n+1}\right)^n < e^{-2}$ für $n \geq 2$ benutzt haben.

Für die zweite Abschätzung benutzen wir (4.3) und das bekannte Resultat $\det B \leq \|B\|^n$ für jede Matrix B :

$$\begin{aligned} \frac{\det A_{k+1}}{\det A_{k+1}^*} &= \det \left(I + (A_{k+1}^*)^{-1} (A_{k+1} - A_{k+1}^*) \right) \\ &\leq \left\| I + (A_{k+1}^*)^{-1} (A_{k+1} - A_{k+1}^*) \right\|^n \\ &\leq \left(\|I\| + \|(A_{k+1}^*)^{-1}\| \|A_{k+1} - A_{k+1}^*\| \right)^n \\ &\leq \left(1 + (R^{-2} 4^{k+1}) (n 2^{-p}) \right)^n \\ &\leq \left(1 + \frac{1}{10n^2} \right)^n \\ &\leq e^{\frac{1}{10n}} \end{aligned}$$

(hier haben wir $2^{-p} \leq \frac{4}{10n^3 4^N} \leq \frac{R^2}{10n^3 4^{k+1}}$ gebraucht.) Damit folgt

$$\frac{\text{volume}(E_{k+1})}{\text{volume}(E_k)} = \sqrt{\frac{\det A_{k+1}^*}{\det A_k}} \sqrt{\frac{\det A_{k+1}}{\det A_{k+1}^*}} \leq e^{-\frac{1}{4n}} e^{\frac{1}{20n}} = e^{-\frac{1}{5n}}.$$

□

4.5 Der Satz von Khachiyan

In diesem Abschnitt werden wir den Satz von Khachiyan beweisen: Die ELLIPSOIDMETHODE kann auf LINEARE OPTIMIERUNG angewendet werden, um einen polynomiellen Algorithmus zu bekommen. Zunächst beweisen wir, dass es genügt, einen Algorithmus zur Prüfung der Zulässigkeit von linearen Ungleichungssystemen zu haben:

Proposition 4.16. *Angenommen, es gibt einen polynomiellen Algorithmus für das folgende Problem: „Gegeben sei eine Matrix $A \in \mathbb{Q}^{m \times n}$ und ein Vektor $b \in \mathbb{Q}^m$. Man entscheide, ob $\{x : Ax \leq b\}$ leer ist.“ Dann gibt es einen polynomiellen Algorithmus für LINEARE OPTIMIERUNG, der eine optimale Basislösung bestimmt, falls es eine solche gibt.*

Beweis: Gegeben sei ein LP $\max\{cx : Ax \leq b\}$. Zunächst prüfen wir, ob das primale und das duale LP beide zulässig sind. Ist eins von beiden unzulässig, so sind wir nach Satz 3.27 fertig. Andernfalls genügt es nach Korollar 3.21, ein Element aus $\{(x, y) : Ax \leq b, yA = c, y \geq 0, cx = yb\}$ zu finden.

Nun zeigen wir mittels Induktion über k , dass man eine Lösung eines zulässigen Systems von k Ungleichungen und l Gleichungen durch k Aufrufe einer Subroutine zur Prüfung des Leerseins von Polyedern finden kann, plus weiterer polynomieller Arbeit. Für $k = 0$ kann eine Lösung leicht mit GAUSS-ELIMINATION gefunden werden (Korollar 4.11).

Sei nun $k > 0$. Sei ferner $ax \leq \beta$ eine Ungleichung des Systems. Mit einem Aufruf der Subroutine prüfen wir, ob das System unzulässig wird, wenn $ax \leq \beta$ durch $ax = \beta$ ersetzt wird. Falls ja, so ist die Ungleichung redundant und kann entfernt werden (siehe Proposition 3.8). Falls nein, so ersetzen wir sie durch die Gleichung. In beiden Fällen hat sich die Anzahl der Ungleichungen um eins verringert, womit der Induktionsschritt vollzogen ist.

Wenn es überhaupt optimale Basislösungen gibt, so erzeugt das obige Verfahren eine, weil das allerletzte Gleichungssystem ein maximales zulässiges Teilsystem von $Ax = b$ enthält. \square

Bevor wir die ELLIPSOIDMETHODE anwenden können, müssen wir sicherstellen, dass das Polyeder beschränkt und volldimensional ist:

Proposition 4.17. (Khachiyan [1979], Gács und Lovász [1981]) *Sei $A \in \mathbb{Q}^{m \times n}$ und $b \in \mathbb{Q}^m$. Das System $Ax \leq b$ hat genau dann eine Lösung, wenn das System*

$$Ax \leq b + \epsilon \mathbb{1}, \quad -R \mathbb{1} \leq x \leq R \mathbb{1}$$

eine Lösung hat, wobei $\mathbb{1}$ der Vektor aus lauter Einsen ist, $\frac{1}{\epsilon} = 2n2^{4(\text{size}(A)+\text{size}(b))}$ und $R = 1 + 2^{4(\text{size}(A)+\text{size}(b))}$.

Hat das System $Ax \leq b$ eine Lösung, so ist $\text{volume}(\{x \in \mathbb{R}^n : Ax \leq b + \epsilon \mathbb{1}, -R \mathbb{1} \leq x \leq R \mathbb{1}\}) \geq \left(\frac{2\epsilon}{n2^{\text{size}(A)}}\right)^n$.

Beweis: Die Ungleichungen $-R \mathbb{1} \leq x \leq R \mathbb{1}$ ändern nach Satz 4.4 nicht die Lösbarkeit. Angenommen, das System $Ax \leq b$ hat keine Lösungen. Nach Satz 3.24 (eine Version von Farkas' Lemma) gibt es einen Vektor $y \geq 0$ mit $yA = 0$ und

$yb = -1$. Nun wenden wir Satz 4.4 auf $\min\{\|y\| : y \geq 0, A^\top y = 0, b^\top y = -1\}$ an und folgern, dass y so gewählt werden kann, dass alle Komponenten von y im Betrage kleiner als $2^{4(\text{size}(A)+\text{size}(b))}$ sind. Somit gilt $y(b + \epsilon \mathbb{1}) < -1 + (n+1)2^{4(\text{size}(A)+\text{size}(b))}\epsilon \leq 0$. Wiederum mit Satz 3.24 folgt dann, dass $Ax \leq b + \epsilon \mathbb{1}$ keine Lösungen hat.

Zum Beweis der zweiten Aussage bemerken wir: Ist $x \in \mathbb{R}^n$ mit $Ax \leq b$ und sind alle Komponenten von x im Betrage kleiner oder gleich $R - 1$ (siehe Satz 4.4), dann enthält die Menge $\{x \in \mathbb{R}^n : Ax \leq b + \epsilon \mathbb{1}, -R \mathbb{1} \leq x \leq R \mathbb{1}\}$ alle Punkte z mit $\|z - x\|_\infty \leq \frac{\epsilon}{n2^{\text{size}(A)}}$. \square

Beachte, dass die in Proposition 4.17 enthaltene Konstruktion die Größe des Ungleichungssystems um einen Faktor von höchstens $O(m+n)$ vergrößert.

Satz 4.18. (Khachiyan [1979]) *Es gibt einen polynomiellen Algorithmus für LINEARE OPTIMIERUNG (mit rationalem Input), und dieser Algorithmus bestimmt eine optimale Basislösung, falls es eine gibt.*

Beweis: Nach Proposition 4.16 genügt es, die Zulässigkeit eines Systems $Ax \leq b$ zu prüfen. Wir transformieren das System wie in Proposition 4.17, um ein Polytop P zu erhalten, welches entweder leer ist oder dessen Volumen mindestens $\left(\frac{2\epsilon}{n2^{\text{size}(A)}}\right)^n$ ist.

Wir wenden die ELLIPSOIDMETHODE mit $x_0 = 0$, $R = n(1 + 2^{4(\text{size}(A)+\text{size}(b))})$ und $N = \lceil 10n^2(2\log n + 5(\text{size}(A) + \text{size}(b))) \rceil$ an. In jedem Schritt ② prüfen wir, ob $x_k \in P$. Falls ja, so sind wir fertig. Sonst nehmen wir eine verletzte Ungleichung $ax \leq \beta$ des Systems $Ax \leq b$ und setzen $a_k := -a$.

Wir behaupten nun: Falls der Algorithmus kein $x_k \in P$ vor Iteration N findet, dann ist P leer. Um dies zu sehen, bemerken wir zunächst, dass $P \subseteq E_k$ für alle k : Für $k = 0$ ist dies nach der Konstruktion von P und R klar und der Induktionsschritt ist durch Lemma 4.14 gegeben. Also gilt $P \subseteq E_N$.

Schreiben wir $s := \text{size}(A) + \text{size}(b)$, so haben wir nach Lemma 4.15

$$\begin{aligned} \text{volume}(E_N) &\leq \text{volume}(E_0)e^{-\frac{N}{5n}} \leq (2R)^n e^{-\frac{N}{5n}} \\ &< \left(2n\left(1 + 2^{4s}\right)\right)^n n^{-4n} e^{-10ns} < n^{-2n} 2^{-5ns}. \end{aligned}$$

Andererseits folgt aus $P \neq \emptyset$ der Widerspruch

$$\text{volume}(P) \geq \left(\frac{2\epsilon}{n2^s}\right)^n = \left(\frac{1}{n^2 2^{5s}}\right)^n = n^{-2n} 2^{-5ns}.$$

\square

Verwenden wir die obige Methode zur Lösung eines LP $\max\{cx : Ax \leq b\}$, dann ergibt eine Abschätzung der Laufzeit die Schranke $O((n+m)^9(\text{size}(A) + \text{size}(b) + \text{size}(c))^2)$ (Aufgabe 9), welche zwar polynomiell aber für die Praxis vollkommen untauglich ist. In der Praxis benutzt man entweder den SIMPLEXALGORITHMUS oder Innere-Punkte-Algorithmen. Polynomielle Innere-Punkte-Algorithmen für LINEARE OPTIMIERUNG wurden zuerst von Karmarkar [1984] beschrieben. Die derzeit schnellste Version stammt von Lee und Sidford [2014]. Dieses Thema werden wir hier nicht behandeln.

Ein streng polynomieller Algorithmus für LINEARE OPTIMIERUNG ist bis dato nicht bekannt. Tardos [1986] hat jedoch zeigen können, dass es einen Algorithmus zur Lösung von $\max\{cx : Ax \leq b\}$ gibt, dessen Laufzeit nur polynomiell von $\text{size}(A)$ abhängt. Für viele kombinatorische Optimierungsprobleme, bei denen A eine 0-1-Matrix ist, ergibt dies einen streng polynomiellen Algorithmus. Tardos' Resultat wurde von Frank und Tardos [1987] erweitert.

4.6 Separation und Optimierung

Obige Methode (insbesondere Proposition 4.16) setzt voraus, dass das Polyeder explizit in Form einer Liste von Ungleichungen vorliegt. Eine genauere Betrachtung zeigt jedoch, dass dies nicht unbedingt notwendig ist. Es genügt eine Subroutine, welche für einen gegebenen Vektor x entscheidet, ob $x \in P$ ist, und falls nicht, eine trennende Hyperebene liefert, d. h. einen Vektor a mit $ax > \max\{ay : y \in P\}$. Wir werden dies für volldimensionale Polyeder beweisen. Für den komplizierteren allgemeinen Fall verweisen wir auf Grötschel, Lovász und Schrijver [1988] (oder Padberg [1999]). Die Resultate in diesem Abschnitt wurden von Grötschel, Lovász und Schrijver [1981] und unabhängig von Karp und Papadimitriou [1982] und von Padberg und Rao [1981] bewiesen.

Mit den Resultaten dieses Abschnitts wird man gewisse LPs polynomiell lösen können, obwohl das Polytop eine exponentielle Anzahl von Facetten hat. Viele Beispiele hierzu werden später noch besprochen, siehe z. B. Korollar 12.22 oder Satz 20.40. Indem man das duale LP betrachtet, kann man auch LPs mit einer riesigen Anzahl von Variablen lösen.

Sei $P \subseteq \mathbb{R}^n$ ein volldimensionales Polytop, oder allgemeiner, eine volldimensionale beschränkte konvexe Menge. Wir nehmen nun an, dass wir die Dimension n und zwei Kugeln $B(x_0, r)$ und $B(x_0, R)$ mit $B(x_0, r) \subseteq P \subseteq B(x_0, R)$ kennen. Wir nehmen jedoch nicht an, dass wir ein P definierendes lineares Ungleichungssystem kennen. Dies würde in der Tat auch nicht sinnvoll sein, wenn wir LPs mit einer exponentiellen Anzahl von Nebenbedingungen in polynomieller Zeit lösen wollten, oder auch lineare Zielfunktionen über konvexen Mengen, die mittels nichtlinearer Nebenbedingungen gegeben sind, optimieren wollten.

Weiter unten werden wir zeigen, dass wir unter einigen akzeptablen Bedingungen eine lineare Funktion über einem Polyeder P polynomiell (unabhängig von der Anzahl der Nebenbedingungen) optimieren können, falls wir ein sogenanntes **Trennungs-Orakel** haben, d. h. eine Subroutine für das folgende Problem:

SEPARATIONS-PROBLEM

Instanz: Eine konvexe Menge $P \subseteq \mathbb{R}^n$. Ein Vektor $y \in \mathbb{Q}^n$.

Aufgabe: Entweder entscheide, dass $y \in P$,
oder bestimme einen Vektor $d \in \mathbb{Q}^n$ mit $dx < dy$ für alle $x \in P$.

Beachte, dass es einen solchen Vektor d gibt, falls P ein rationales Polyeder oder eine kompakte konvexe Menge ist (siehe Aufgabe 21, Kapitel 3). Gegeben sei eine konvexe Menge P mit einem solchen Trennungs-Orakel. Wir suchen einen **Orakel-Algorithmus**, der dieses Trennungs-Orakel als Blackbox benutzt. In einem Orakel-

Algorithmus dürfen wir das Orakel jederzeit für ein beliebiges $y \in \mathbb{Q}^n$ aufrufen und erhalten die korrekte Antwort in einem Schritt. Diesen Vorgang betrachten wir als eine Subroutine, deren Laufzeit wir nicht in Betracht zu ziehen brauchen. (Eine formale Definition werden wir in Kapitel 15 geben.)

Tatsächlich genügt es oft, ein Orakel zu haben, welches das SEPARATIONS-PROBLEM nur approximativ löst. Genauer gesagt, setzen wir ein Orakel für das folgende Problem voraus:

SCHWACHES SEPARATIONS-PROBLEM

Instanz: Eine konvexe Menge $P \subseteq \mathbb{R}^n$, ein Vektor $c \in \mathbb{Q}^n$ und eine Zahl $\epsilon > 0$. Ein Vektor $y \in \mathbb{Q}^n$.

Aufgabe: Entweder bestimme einen Vektor $y' \in P$ mit $cy \leq cy' + \epsilon$
oder bestimme einen Vektor $d \in \mathbb{Q}^n$ mit $dx < dy$ für alle $x \in P$.

Zunächst werden wir LPs mit einem schwachen Separations-Orakel approximativ lösen:

SCHWACHES OPTIMIERUNGSPROBLEM

Instanz: Eine Zahl $n \in \mathbb{N}$. Ein Vektor $c \in \mathbb{Q}^n$. Eine Zahl $\epsilon > 0$.
Eine konvexe Menge $P \subseteq \mathbb{R}^n$, gegeben durch ein Orakel für das SCHWACHE SEPARATIONS-PROBLEM, c und $\frac{\epsilon}{2}$.

Aufgabe: Bestimme einen Vektor $y \in P$ mit $cy \geq \sup\{cx : x \in P\} - \epsilon$.

Wir weisen darauf hin, dass die obigen beiden Definitionen nicht mit denen übereinstimmen, die z. B. in Grötschel, Lovász und Schrijver [1981] gegeben sind. Sie sind jedoch im Grunde mit diesen äquivalent und wir werden die obige Form wieder in Abschnitt 18.3 brauchen.

Die folgende Variante der ELLIPSOIDMETHODE löst das SCHWACHE OPTIMIERUNGSPROBLEM für beschränkte volldimensionale konvexe Mengen:

GRÖTSCHEL-LOVÁSZ-SCHRIJVER-ALGORITHMUS

Input: Eine Zahl $n \in \mathbb{N}$, $n \geq 2$. Ein Vektor $c \in \mathbb{Q}^n$. Eine Zahl $0 < \epsilon \leq 1$.
Eine konvexe Menge $P \subseteq \mathbb{R}^n$, gegeben durch ein Orakel für das SCHWACHE SEPARATIONS-PROBLEM, c und $\frac{\epsilon}{2}$.
Ein $x_0 \in \mathbb{Q}^n$ und $r, R \in \mathbb{Q}_+$, so dass $B(x_0, r) \subseteq P \subseteq B(x_0, R)$.

Output: Ein Vektor $y^* \in P$ mit $cy^* \geq \sup\{cx : x \in P\} - \epsilon$.

- ① Setze $R := \max\{R, 2\}$, $r := \min\{r, 1\}$ und $\gamma := \max\{|c|, 1\}$.
Setze $N := 5n^2 \left\lceil \ln \frac{6R^2\gamma}{r\epsilon} \right\rceil$. Setze $y^* := x_0$.
- ② Wende die ELLIPSOIDMETHODE an, wobei a_k in ② wie folgt berechnet wird: Rufe das Orakel für das SCHWACHE SEPARATIONS-PROBLEM mit $y = x_k$ auf.
If es liefert ein $y' \in P$ mit $cy \leq cy' + \frac{\epsilon}{2}$ **then:**
If $cy' > cy^*$ **then** setze $y^* := y'$.
Setze $a_k := c$.
If es liefert ein $d \in \mathbb{Q}^n$ mit $dx < dy$ für alle $x \in P$ **then:**
Setze $a_k := -d$.

Satz 4.19. Der GRÖTSCHEL-LOVÁSZ-SCHRIJVER ALGORITHMUS löst das SCHWACHE OPTIMIERUNGSPROBLEM für beschränkte volldimensionale konvexe Mengen korrekt. Die Laufzeit ist durch

$$O\left(n^6\alpha^2 + n^4\alpha f(\text{size}(c), \text{size}(\epsilon), n \text{ size}(x_0) + n^3\alpha)\right)$$

beschränkt, wobei $\alpha = \log \frac{R^2\gamma}{r\epsilon}$ und $f(\text{size}(c), \text{size}(\epsilon), \text{size}(y))$ eine obere Schranke für die Laufzeit des Orakels für das SCHWACHE SEPARATIONS-PROBLEM für P mit Input c, ϵ und y ist.

Beweis: (Grötschel, Lovász und Schrijver [1981]) In jeder der $N = O(n^2\alpha)$ Iterationen der ELLIPSOIDMETHODE ist die Laufzeit $O(n^2(n^2\alpha + \text{size}(R) + \text{size}(x_0) + q))$ plus ein Orakel-Aufruf, wobei q die Größe des Orakel-Outputs ist. Da $\text{size}(y) \leq n(\text{size}(x_0) + \text{size}(R) + N)$ nach Lemma 4.13, ist die Gesamlaufzeit $O(n^4\alpha(n^2\alpha + \text{size}(x_0) + f(\text{size}(c), \text{size}(\epsilon), n \text{ size}(x_0) + n^3\alpha)))$, wie erwünscht.

Mit Lemma 4.14 haben wir

$$\left\{x \in P : cx \geq cy^* + \frac{\epsilon}{2}\right\} \subseteq E_N.$$

Sei $z \in P$ mit $cz \geq \sup\{cx : x \in P\} - \frac{\epsilon}{6}$. Wir können annehmen, dass $cz > cy^* + \frac{\epsilon}{2}$; anderenfalls sind wir fertig.

Wir betrachten nun die konvexe Hülle U von z und der $(n-1)$ -dimensionalen Kugel $B(x_0, r) \cap \{x : cx = cx_0\}$ (siehe Abb. 4.2).

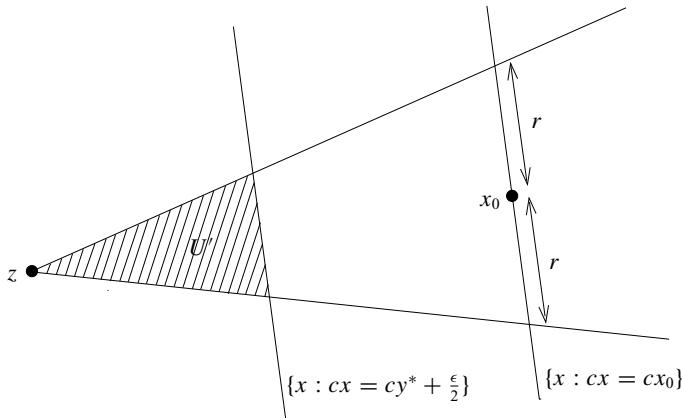


Abbildung 4.2.

Es folgt $U' \subseteq P$, also ist $U' := \{x \in U : cx \geq cy^* + \frac{\epsilon}{2}\}$ in E_N enthalten. Das Volumen von U' ist

$$\begin{aligned} \text{volume}(U') &= \text{volume}(U) \left(\frac{cz - cy^* - \frac{\epsilon}{2}}{cz - cx_0} \right)^n \\ &= V_{n-1} r^{n-1} \frac{cz - cx_0}{n||c||} \left(\frac{cz - cy^* - \frac{\epsilon}{2}}{cz - cx_0} \right)^n, \end{aligned}$$

wobei V_n das Volumen der n -dimensionalen Einheitskugel bezeichnet. Es ist $\text{volume}(U') \leq \text{volume}(E_N)$ und mit Lemma 4.15 folgt

$$\text{volume}(E_N) \leq e^{-\frac{N}{5n}} \text{volume}(E_0) = e^{-\frac{N}{5n}} V_n R^n.$$

Damit haben wir

$$cz - cy^* - \frac{\epsilon}{2} \leq e^{-\frac{N}{5n^2}} R \left(\frac{V_n(cz - cx_0)^{n-1} n ||c||}{V_{n-1} r^{n-1}} \right)^{\frac{1}{n}}.$$

Da $cz - cx_0 \leq ||c|| \cdot ||z - x_0|| \leq ||c||R$, folgt

$$cz - cy^* - \frac{\epsilon}{2} \leq ||c||e^{-\frac{N}{5n^2}} R \left(\frac{n V_n R^{n-1}}{V_{n-1} r^{n-1}} \right)^{\frac{1}{n}} < 2 ||c|| e^{-\frac{N}{5n^2}} \frac{R^2}{r} \leq \frac{\epsilon}{3}$$

und somit $cy^* \geq cz - \frac{5}{6}\epsilon \geq \sup\{cx : x \in P\} - \epsilon$. \square

Natürlich sind wir im Normalfall am exakten Optimum interessiert. Dazu beschränken wir uns auf rationale volldimensionale Polytope. Ferner brauchen wir eine Annahme zur Größe der Ecken des Polytops.

Lemma 4.20. Sei $n \in \mathbb{N}$, $P \subseteq \mathbb{R}^n$ ein rationales Polytop und $x_0 \in \mathbb{Q}^n$ ein Punkt im Innern von P . Sei $T \in \mathbb{N}$ mit $\text{size}(x_0) \leq \log T$ und $\text{size}(x) \leq \log T$ für alle Ecken x von P . Dann gilt $B(x_0, r) \subseteq P \subseteq B(x_0, R)$, wobei $r := \frac{1}{n}T^{-379n^2}$ und $R := 2nT$.

Sei ferner $K := 4T^{2n+1}$. Sei $c \in \mathbb{Z}^n$ und setze $c' := K^n c + (1, K, \dots, K^{n-1})$. Dann wird $\max\{c'x : x \in P\}$ für einen eindeutig bestimmten Vektor x^* angenommen, für alle anderen Ecken y von P gilt $c'(x^* - y) > T^{-2n}$ und x^* ist auch eine optimale Lösung von $\max\{cx : x \in P\}$.

Beweis: Für jede Ecke x von P gilt $||x|| \leq nT$ und $||x_0|| \leq nT$, also folgt $||x - x_0|| \leq 2nT$ und $x \in B(x_0, R)$.

Um zu zeigen, dass $B(x_0, r) \subseteq P$, sei $F = \{x \in P : ax = \beta\}$ eine Facette von P , wobei wir nach Lemma 4.5 annehmen können, dass $\text{size}(a) + \text{size}(\beta) < 75n^2 \log T$. Angenommen, es gäbe einen Punkt $y \in F$ mit $||y - x_0|| < r$.

Dann folgt

$$|ax_0 - \beta| = |ax_0 - ay| \leq ||a|| \cdot ||y - x_0|| < n2^{\text{size}(a)}r \leq T^{-304n^2}.$$

Andererseits kann aber die Größe von $ax_0 - \beta$ wie folgt abgeschätzt werden:

$$\begin{aligned} \text{size}(ax_0 - \beta) &\leq 4(\text{size}(a) + \text{size}(x_0) + \text{size}(\beta)) \\ &\leq 300n^2 \log T + 4 \log T \leq 304n^2 \log T. \end{aligned}$$

Da $ax_0 \neq \beta$ (x_0 ist ein Punkt im Innern von P), folgt hieraus, dass $|ax_0 - \beta| \geq T^{-304n^2}$. Dies ist jedoch ein Widerspruch.

Um die letzten Aussagen zu beweisen, seien x^* eine $c'x$ maximierende Ecke von P und y eine weitere Ecke von P . Mit der Annahme zur Größe der Ecken

von P können wir $x^* - y = \frac{1}{\alpha}z$ schreiben, wobei $\alpha \in \{1, 2, \dots, T^{2n} - 1\}$ und z ein ganzzahliger Vektor ist, dessen Komponenten höchstens den Betrag $\frac{K}{2}$ haben. Dann folgt

$$0 \leq c'(x^* - y) = \frac{1}{\alpha} \left(K^n cz + \sum_{i=1}^n K^{i-1} z_i \right).$$

Da $K^n > \sum_{i=1}^n K^{i-1} |z_i|$, folgt $cz \geq 0$ und somit ist $c x^* \geq c y$. Also maximiert x^* tatsächlich $c x$ über P . Ferner haben wir, wie erwünscht,

$$c'(x^* - y) \geq \frac{1}{\alpha} > T^{-2n},$$

da $z \neq 0$. □

Satz 4.21. Sei $n \in \mathbb{N}$ und $c \in \mathbb{Q}^n$. Sei $P \subseteq \mathbb{R}^n$ ein rationales Polytop und $x_0 \in \mathbb{Q}^n$ ein Punkt im Innern von P . Sei ferner $T \in \mathbb{N}$ mit $\text{size}(x_0) \leq \log T$ und $\text{size}(x) \leq \log T$ für alle Ecken x von P .

Sind n, c, x_0, T und ein polynomielles Orakel für das SEPARATIONS-PROBLEM für P gegeben, so kann man das Maximum $\max\{c^\top x : x \in P\}$ annehmende Ecke x^* von P in polynomieller Zeit bestimmen, wobei die Zeit polynomiell in n , $\log T$ und $\text{size}(c)$ ist.

Beweis: (Grötschel, Lovász und Schrijver [1981]) Als erstes wenden wir den GRÖTSCHEL-LOVÁSZ-SCHRIJVER-ALGORITHMUS an, um das SCHWACHE OPTIMIERUNGSPROBLEM zu lösen; wir setzen c' , r und R wie in Lemma 4.20 und $\epsilon := \frac{1}{8nT^{2n+3}}$. (Vorher müssen wir c noch ganzzahlig machen, indem wir mit dem Produkt der Nenner multiplizieren; dadurch wächst die Größe von c um höchstens den Faktor $2n$.)

Der GRÖTSCHEL-LOVÁSZ-SCHRIJVER-ALGORITHMUS liefert einen Vektor $y \in P$ mit $c'y \geq c'x^* - \epsilon$, wobei x^* eine optimale Lösung von $\max\{c'x : x \in P\}$ ist. Nach Satz 4.19 ist die Laufzeit $O(n^6\alpha^2 + n^4\alpha f(\text{size}(c'), \text{size}(\epsilon), n \text{size}(x_0) + n^3\alpha)) = O(n^6\alpha^2 + n^4\alpha f(\text{size}(c'), 6n \log T, n \log T + n^3\alpha))$, wobei $\alpha = \log \frac{R^2 \max\{\|c'\|, 1\}}{r\epsilon} \leq \log(32n^4 T^{400n^2} 2^{\text{size}(c')}) = O(n^2 \log T + \text{size}(c'))$ und f eine polynomielle obere Schranke für die Laufzeit des Orakels für das SEPARATIONS-PROBLEM für P ist. Da $\text{size}(c') \leq 6n^2 \log T + 2 \text{size}(c)$, haben wir eine in n , $\log T$ und $\text{size}(c)$ polynomielle Gesamlaufzeit.

Wir behaupten nun, dass $\|x^* - y\| \leq \frac{1}{2T^2}$. Um dies zu sehen, schreiben wir y als Konvexitätskombination der Ecken x^*, x_1, \dots, x_k von P :

$$y = \lambda_0 x^* + \sum_{i=1}^k \lambda_i x_i, \quad \lambda_i \geq 0, \quad \sum_{i=0}^k \lambda_i = 1.$$

Ferner haben wir nach Lemma 4.20:

$$\epsilon \geq c'(x^* - y) = \sum_{i=1}^k \lambda_i c' (x^* - x_i) > \sum_{i=1}^k \lambda_i T^{-2n} = (1 - \lambda_0) T^{-2n},$$

also folgt $1 - \lambda_0 < \epsilon T^{2n}$. Damit haben wir dann

$$\|y - x^*\| \leq \sum_{i=1}^k \lambda_i \|x_i - x^*\| \leq (1 - \lambda_0)2R < 4nT^{2n+1}\epsilon \leq \frac{1}{2T^2}.$$

Nach dem Runden der Komponenten von y zur nächsten rationalen Zahl mit Nenner kleiner oder gleich T , erhalten wir schließlich x^* . Dieses Runden kann nach Satz 4.8 in polynomieller Zeit vollzogen werden. \square

Wir haben bewiesen, dass man unter gewissen Annahmen immer dann über einem Polytop optimieren kann, wenn ein Trennungs-Orakel zur Verfügung steht. Wir schließen dieses Kapitel mit der Bemerkung, dass die Umkehrung auch gilt. Dazu benötigen wir den Begriff der Polarität: Für $X \subseteq \mathbb{R}^n$ definieren wir die zu X **polare Menge** als die Menge

$$X^\circ := \{y \in \mathbb{R}^n : y^\top x \leq 1 \text{ für alle } x \in X\}.$$

Angewendet auf volldimensionale Polytope hat diese Operation einige schöne Eigenschaften:

Satz 4.22. *Sei P ein Polytop in \mathbb{R}^n mit 0 im Innern von P . Dann gilt:*

- (a) P° ist ein Polytop und 0 liegt im Innern von P° ;
- (b) $(P^\circ)^\circ = P$;
- (c) x ist genau dann eine Ecke von P , wenn $x^\top y \leq 1$ eine facettenbestimmende Ungleichung für P° ist.

Beweis: (a): Sei P die konvexe Hülle der Punkte x_1, \dots, x_k (siehe Satz 3.31). Nach obiger Definition ist $P^\circ = \{y \in \mathbb{R}^n : y^\top x_i \leq 1 \text{ für alle } i \in \{1, \dots, k\}\}$, d.h. P° ist ein Polyeder und die facettenbestimmenden Ungleichungen für P° werden durch die Ecken von P gegeben. Ferner liegt 0 im Innern von P° , da 0 jede der endlich vielen Ungleichungen streng erfüllt. Angenommen, P° sei unbeschränkt, d.h. es gibt ein $w \in \mathbb{R}^n \setminus \{0\}$ mit $\alpha w \in P^\circ$ für alle $\alpha > 0$. Dann folgt $\alpha w x \leq 1$ für alle $\alpha > 0$ und alle $x \in P$, also ist $wx \leq 0$ für alle $x \in P$. Dann ist 0 aber nicht im Innern von P .

(b): Es gilt $P \subseteq (P^\circ)^\circ$ trivialerweise. Um die Umkehrung zu beweisen, nehmen wir an, es gäbe ein $z \in (P^\circ)^\circ \setminus P$. Dann gibt es eine Ungleichung $c^\top x \leq \delta$, welche von allen $x \in P$ aber nicht von z erfüllt wird. Es gilt $\delta > 0$, da 0 im Innern von P liegt. Damit folgt $\frac{1}{\delta}c \in P^\circ$, aber $\frac{1}{\delta}c^\top z > 1$, im Widerspruch zur Annahme $z \in (P^\circ)^\circ$.

(c): In (a) haben wir bereits gesehen, dass die facettenbestimmenden Ungleichungen für P° durch die Ecken von P gegeben werden. Umgekehrt gilt: Sind x_1, \dots, x_k die Ecken von P , dann ist $\bar{P} := \text{conv}(\{\frac{1}{2}x_1, x_2, \dots, x_k\}) \neq P$ und 0 liegt im Innern von \bar{P} . Nun folgt aber $\bar{P}^\circ \neq P^\circ$ aus (b). Damit haben wir $\{y \in \mathbb{R}^n : y^\top x_1 \leq 2, y^\top x_i \leq 1(i = 2, \dots, k)\} = \bar{P}^\circ \neq P^\circ = \{y \in \mathbb{R}^n : y^\top x_i \leq 1(i = 1, \dots, k)\}$. Somit ist $x_1^\top y \leq 1$ eine facettenbestimmende Ungleichung für P° . \square

Nun können wir den folgenden Satz beweisen:

Satz 4.23. Sei $n \in \mathbb{N}$ und $y \in \mathbb{Q}^n$. Sei $P \subseteq \mathbb{R}^n$ ein rationales Polytop und $x_0 \in \mathbb{Q}^n$ ein Punkt im Innern von P . Sei ferner $T \in \mathbb{N}$ mit $\text{size}(x_0) \leq \log T$ und $\text{size}(x) \leq \log T$ für alle Ecken x von P .

Sind n , y , x_0 , T und ein Orakel gegeben, welches für jedes gegebene $c \in \mathbb{Q}^n$ eine das Maximum $\max\{c^\top x : x \in P\}$ annehmende Ecke x^* von P liefert, so kann man das SEPARATIONS-PROBLEM für P und y in polynomieller Zeit lösen, wobei die Zeit polynomiell in n , $\log T$ und $\text{size}(y)$ ist. In der Tat kann man im Falle $y \notin P$ eine facettenbestimmende Ungleichung für P bestimmen, die durch y verletzt wird.

Beweis: Betrachte die Menge $Q := \{x - x_0 : x \in P\}$ und ihre polare Menge Q° . Sind x_1, \dots, x_k die Ecken von P , so haben wir

$$Q^\circ = \{z \in \mathbb{R}^n : z^\top(x_i - x_0) \leq 1 \text{ für alle } i \in \{1, \dots, k\}\}.$$

Nach Satz 4.4 gilt $\text{size}(z) \leq 4n(4n \log T + 3n) \leq 28n^2 \log T$ für alle Ecken z von Q° .

Beachte, dass das SEPARATIONS-PROBLEM für P und y äquivalent zum SEPARATIONS-PROBLEM für Q und $y - x_0$ ist. Da nach Satz 4.22

$$Q = (Q^\circ)^\circ = \{x : zx \leq 1 \text{ für alle } z \in Q^\circ\}$$

gilt, ist das SEPARATIONS-PROBLEM für Q und $y - x_0$ seinerseits äquivalent mit dem Problem der Lösung von $\max\{(y - x_0)^\top x : x \in Q^\circ\}$. Da jede Ecke von Q° einer facettenbestimmenden Ungleichung für Q (also auch für P) entspricht, bleibt zu zeigen, wie man eine das Maximum $\max\{(y - x_0)^\top x : x \in Q^\circ\}$ annehmende Ecke finden kann.

Dazu wenden wir Satz 4.21 auf Q° an. Nach Satz 4.22 ist Q° volldimensional und 0 liegt im Innern von Q° . Weiter oben haben wir gezeigt, dass die Größe der Ecken von Q° höchstens $28n^2 \log T$ ist. Also bleibt zu zeigen, dass wir das SEPARATIONS-PROBLEM für Q° polynomiell lösen können. Dies reduziert sich jedoch auf das Optimierungsproblem für Q , welches mit dem Orakel für die Optimierung über P gelöst werden kann. \square

Schließlich erwähnen wir noch, dass ein neuer Algorithmus, der schneller als die ELLIPSOIDMETHODE ist und aus welchem ferner die Äquivalenz von Optimierung und Separation folgt, von Vaidya [1996] entwickelt worden ist. Dieser Algorithmus scheint aber auch nicht für die Praxis geeignet zu sein.

Aufgaben

- Sei A eine nichtsinguläre rationale $n \times n$ -Matrix. Man beweise, dass $\text{size}(A^{-1}) \leq 4n^2 \text{size}(A)$.

- * 2. Sei $n \geq 2$, $c \in \mathbb{R}^n$ und $y_1, \dots, y_k \in \{-1, 0, 1\}^n$ mit $0 < c^\top y_{i+1} \leq \frac{1}{2}c^\top y_i$ für $i = 1, \dots, k-1$. Man zeige, dass dann $k \leq 3n \log n$.

Hinweis: Man betrachte das LP $\max\{y_1^\top x : y_k^\top x = 1, (y_i - 2y_{i+1})^\top x \geq 0 \ (i = 1, \dots, k-1)\}$ und den Beweis von Satz 4.4.

(M. Goemans)

- 3. Man betrachte die in der KETTENBRUCH-ERWEITERUNG vorkommenden Zahlen h_i . Man beweise, dass $h_i \geq F_{i+1}$ für alle i , wobei F_i die i -te Fibonacci-Zahl ist ($F_1 = F_2 = 1$ und $F_n = F_{n-1} + F_{n-2}$ für $n \geq 3$). Man beachte, dass

$$F_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right).$$

Daraus schließe man, dass die Anzahl der Iterationen der KETTENBRUCH-ERWEITERUNG $O(\log q)$ ist.

(Grötschel, Lovász und Schrijver [1988])

- 4. Man zeige, dass die GAUSS-ELIMINATION als streng polynomieller Algorithmus formuliert werden kann.

Hinweis: Man nehme zunächst an, dass A ganzzahlig ist. Ferner betrachte man den Beweis von Satz 4.10 und beachte, dass man d als gemeinsamen Nenner der Elemente wählen kann.

(Edmonds [1967])

- * 5. Seien $x_1, \dots, x_k \in \mathbb{R}^l$, $d := 1 + \dim\{x_1, \dots, x_k\}$, $\lambda_1, \dots, \lambda_k \in \mathbb{R}_+$ mit $\sum_{i=1}^k \lambda_i = 1$, und $x := \sum_{i=1}^k \lambda_i x_i$. Man zeige, wie man Zahlen $\mu_1, \dots, \mu_k \in \mathbb{R}_+$ berechnen kann, von denen höchstens d nicht verschwinden, so dass $\sum_{i=1}^k \mu_i = 1$ und $x = \sum_{i=1}^k \mu_i x_i$ (siehe Aufgabe 15, Kapitel 3). Man zeige, dass alle Berechnungen in $O((k+l)^3)$ -Zeit erfolgen können.

Hinweis: Man wende GAUSS-ELIMINATION auf die Matrix $A \in \mathbb{R}^{(l+1) \times k}$ an, deren i -te Spalte gleich $\begin{pmatrix} 1 \\ x_i \end{pmatrix}$ ist. Falls $d < k$, sei $w \in \mathbb{R}^k$ der Vektor mit $w_{col(i)} := z_{i,d+1}$ ($i = 1, \dots, d$), $w_{col(d+1)} := -1$ und $w_{col(i)} := 0$ ($i = d+2, \dots, k$). Man beachte, dass $Aw = 0$ und addiere ein Vielfaches von w zu λ , eliminiere wenigstens einen Vektor und iteriere.

- 6. Sei $A = \begin{pmatrix} \alpha & b^\top \\ b & C \end{pmatrix} \in \mathbb{R}^{n \times n}$ eine symmetrische positiv semidefinite Matrix mit $\alpha > 0$ und sei $b \in \mathbb{R}^{n-1}$. Sei ferner $A' := \begin{pmatrix} \alpha & 0 \\ 0 & C - \frac{1}{\alpha}bb^\top \end{pmatrix}$ und $U := \begin{pmatrix} 1 & \frac{1}{\alpha}b \\ 0 & I \end{pmatrix}$. Man beweise, dass $A = U^\top A' U$ und dass $C - \frac{1}{\alpha}bb^\top$ positiv semidefinit ist. Man iteriere und schließe daraus, dass es für jede positiv semidefinite Matrix A eine Matrix U mit $A = U^\top U$ gibt, und dass eine solche Matrix beliebig genau in $O(n^3)$ Schritten berechnet werden kann (wobei einige der Schritte die approximative Berechnung von Quadratwurzeln enthalten).

Bemerkung: Dieses Verfahren heißt Cholesky-Faktorisierung. Da U irrational sein kann, können exakte Resultate nicht erwartet werden.

- * 7. Sei A eine symmetrische positiv definite $n \times n$ -Matrix. Ferner seien v_1, \dots, v_n paarweise orthogonale Eigenvektoren von A mit den zugehörigen Eigenwerten $\lambda_1, \dots, \lambda_n$. O. B. d. A. können wir annehmen, dass $\|v_i\| = 1$ für $i = 1, \dots, n$. Dann beweise man, dass

$$E(A, 0) = \left\{ \mu_1 \sqrt{\lambda_1} v_1 + \cdots + \mu_n \sqrt{\lambda_n} v_n : \mu \in \mathbb{R}^n, \|\mu\| \leq 1 \right\}.$$

(Die Eigenvektoren entsprechen den Symmetriearchsen des Ellipsoids.)

Man schließe hieraus, dass $\text{volume}(E(A, 0)) = \sqrt{\det A}$ $\text{volume}(B(0, 1))$.

8. Sei $E(A, x) \subseteq \mathbb{R}^n$ ein Ellipsoid und $a \in \mathbb{R}^n$. Sei ferner $E(A', x')$ wie auf Seite 91 definiert. Man beweise, dass $\{z \in E(A, x) : az \geq ax\} \subseteq E(A', x')$.
9. Man beweise, dass der in Satz 4.18 angegebene Algorithmus ein LP $\max\{cx : Ax \leq b\}$ in $O((n+m)^9(\text{size}(A) + \text{size}(b) + \text{size}(c))^2)$ -Zeit löst.
10. Man zeige, dass man die Annahme der Beschränktheit von P in Satz 4.21 weglassen kann. Man kann nämlich feststellen, ob das LP unbeschränkt ist und, falls nicht, eine optimale Lösung finden.
- * 11. Sei $P \subseteq \mathbb{R}^3$ ein 3-dimensionales Polytop mit 0 im Innern von P . Man betrachte wiederum den Graphen $G(P)$, dessen Knoten den Ecken von P und dessen Kanten den eindimensionalen Seitenflächen von P entsprechen (siehe Aufgaben 18 und 19, Kapitel 3). Man zeige, dass $G(P^\circ)$ der planare duale Graph von $G(P)$ ist.
Bemerkung: Steinitz [1922] hat bewiesen, dass es für jeden einfachen 3-fach zusammenhängenden planaren Graphen G ein 3-dimensionales Polytop P mit $G = G(P)$ gibt.
- * 12. Sei G ein einfacher, zusammenhängender, ungerichteter Graph. Man zeige, dass das LP

$$\begin{aligned} \min \quad & \sum_{e=\{v,w\} \in E(G)} x_{vw} \\ \text{bzgl.} \quad & \sum_{w \in S} x_{vw} \geq \left\lceil \frac{1}{4}|S|^2 + \frac{1}{2}|S| \right\rceil \quad (v \in V(G), S \subseteq V(G) \setminus \{v\}) \\ & x_{uw} \leq x_{uv} + x_{vw} \quad (u, v, w \in V(G)) \\ & x_{vw} \geq 0 \quad (v, w \in V(G)) \\ & x_{vv} = 0 \quad (v \in V(G)) \end{aligned}$$

polynomiell in $|V(G)|$ gelöst werden kann.

Bemerkung: Dies kann als Relaxierung des OPTIMALEN LINEAR-ARRANGEMENT-PROBLEMS betrachtet werden; siehe Aufgabe 8, Kapitel 19.

13. Man zeige, dass die polare Menge eines Polyeders wieder ein Polyeder ist. Für welche Polyeder P gilt $(P^\circ)^\circ = P$?

Literatur

Allgemeine Literatur:

- Grötschel, M., Lovász, L., und Schrijver, A. [1988]: Geometric Algorithms and Combinatorial Optimization. Springer, Berlin 1988
 Padberg, M. [1999]: Linear Optimization and Extensions. 2. Aufl. Springer, Berlin 1999
 Schrijver, A. [1986]: Theory of Linear and Integer Programming. Wiley, Chichester 1986

Zitierte Literatur:

- Bland, R.G., Goldfarb, D., und Todd, M.J. [1981]: The ellipsoid method: a survey. *Operations Research* 29 (1981), 1039–1091
- Edmonds, J. [1967]: Systems of distinct representatives and linear algebra. *Journal of Research of the National Bureau of Standards B* 71 (1967), 241–245
- Frank, A., und Tardos, É. [1987]: An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica* 7 (1987), 49–65
- Gács, P., und Lovász, L. [1981]: Khachiyan's algorithm for linear programming. *Mathematical Programming Study* 14 (1981), 61–68
- Grötschel, M., Lovász, L., und Schrijver, A. [1981]: The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1 (1981), 169–197
- Iudin, D.B., und Nemirovskii, A.S. [1976]: Informational complexity and effective methods of solution for convex extremal problems. *Ekonomika i Matematicheskie Metody* 12 (1976), 357–369 [auf Russisch]
- Karmarkar, N. [1984]: A new polynomial-time algorithm for linear programming. *Combinatorica* 4 (1984), 373–395
- Karp, R.M., und Papadimitriou, C.H. [1982]: On linear characterizations of combinatorial optimization problems. *SIAM Journal on Computing* 11 (1982), 620–632
- Khachiyan, L.G. [1979]: A polynomial algorithm in linear programming [auf Russisch]. *Doklady Akademii Nauk SSSR* 244 (1979) 1093–1096. English translation: Soviet Mathematics Doklady 20 (1979), 191–194
- Khintchine, A. [1956]: Kettenbrüche. Teubner, Leipzig 1956
- Lee, Y.T., und Sidford, A. [2014]: Path finding methods for linear programming. *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science* (2014), 424–433
- Padberg, M.W., und Rao, M.R. [1981]: The Russian method for linear programming III: Bounded integer programming. *Research Report* 81-39, New York University 1981
- Shor, N.Z. [1977]: Cut-off method with space extension in convex programming problems. *Cybernetics* 13 (1977), 94–96
- Steinitz, E. [1922]: Polyeder und Raumeinteilungen. *Enzyklopädie der Mathematischen Wissenschaften*, Band 3 (1922), 1–139
- Tardos, É. [1986]: A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research* 34 (1986), 250–256
- Vaidya, P.M. [1996]: A new algorithm for minimizing convex functions over convex sets. *Mathematical Programming* 73 (1996), 291–341

5 Ganzzahlige Optimierung

In diesem Kapitel betrachten wir lineare Programme mit ganzzahligen Nebenbedingungen:

GANZZAHLIGE OPTIMIERUNG

Instanz: Eine Matrix $A \in \mathbb{Z}^{m \times n}$ und Vektoren $b \in \mathbb{Z}^m$ und $c \in \mathbb{Z}^n$.

Aufgabe: Bestimme einen Vektor $x \in \mathbb{Z}^n$, so dass $Ax \leq b$ und cx maximal ist;
entscheide, dass $\{x \in \mathbb{Z}^n : Ax \leq b\} = \emptyset$;
oder entscheide, dass $\sup\{cx : x \in \mathbb{Z}^n, Ax \leq b\} = \infty$.

Wir werden keine gemischt-ganzzahligen Programme betrachten, d. h. LPs mit ganzzahligen Nebenbedingungen für bloß eine echte Teilmenge der Variablen. Der größte Teil der Theorie für lineare und ganzzahlige Programme lässt sich auf natürliche Weise auf gemischt-ganzzahlige Programme erweitern.

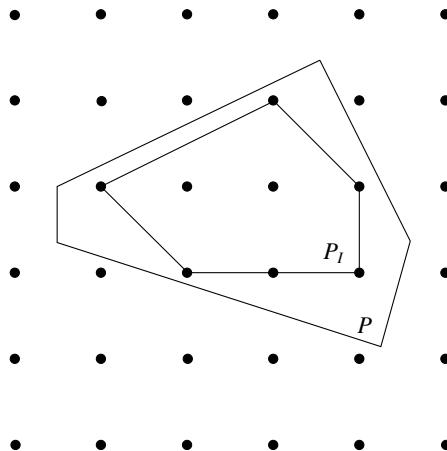


Abbildung 5.1.

Nahezu alle kombinatorischen Optimierungsprobleme können als ganzzahlige LPs formuliert werden. Die Menge der zulässigen Lösungen ist dann die Menge $\{x : Ax \leq b, x \in \mathbb{Z}^n\}$ für eine bestimmte Matrix A und einen bestimmten Vektor

b. Es ist $P := \{x \in \mathbb{R}^n : Ax \leq b\}$ ein Polyeder und mit $P_I = \{x : Ax \leq b\}_I$ bezeichnen wir die konvexe Hülle der ganzzahligen Vektoren in P . Wir nennen P_I die **ganzzahlige Hülle** von P . Offensichtlich gilt $P_I \subseteq P$.

Ist P beschränkt, so ist P_I nach Satz 3.31 wieder ein Polytop (siehe Abb. 5.1). Der folgende Satz wurde von Meyer [1974] bewiesen:

Satz 5.1. *Die ganzzahlige Hülle P_I eines jeden rationalen Polyeders P ist wieder ein rationales Polyeder.*

Beweis: Sei $P = \{x : Ax \leq b\}$. Nach Lemma 3.12 wird der rationale polyedrische Kegel $C := \{(x, \xi) : x \in \mathbb{R}^n, \xi \geq 0, Ax - \xi b \leq 0\}$ durch eine endliche Menge rationaler Vektoren erzeugt. Wir können annehmen, dass $(x_1, 1), \dots, (x_k, 1), (y_1, 0), \dots, (y_l, 0)$, mit x_1, \dots, x_k rational und y_1, \dots, y_l ganzzahlig, C erzeugen (mittels Multiplikation einer endlichen Anzahl von erzeugenden Vektoren mit geeigneten positiven Skalaren).

Betrachte das Polytop

$$Q := \left\{ \sum_{i=1}^k \kappa_i x_i + \sum_{i=1}^l \lambda_i y_i \ : \ \kappa_i \geq 0 \ (i = 1, \dots, k), \ \sum_{i=1}^k \kappa_i = 1, \right. \\ \left. 0 \leq \lambda_i \leq 1 \ (i = 1, \dots, l) \right\}.$$

Beachte, dass $Q \subseteq P$. Seien z_1, \dots, z_m die ganzzahligen Punkte in Q . Nach Satz 3.29 ist der von $(y_1, 0), \dots, (y_l, 0), (z_1, 1), \dots, (z_m, 1)$ erzeugte Kegel C' polyedrisch, d.h. er kann durch $\{(x, \xi) : Mx + \xi d \leq 0\}$, mit Matrix M und Vektor d rational, beschrieben werden.

Wir behaupten, dass $P_I = \{x : Mx \leq -d\}$.

Um „ \subseteq “ zu zeigen, sei $x \in P \cap \mathbb{Z}^n$. Es ist $(x, 1) \in C$, d.h. $x = \sum_{i=1}^k \kappa_i x_i + \sum_{i=1}^l \lambda_i y_i$ für irgendwelche $\kappa_1, \dots, \kappa_k \geq 0$ mit $\sum_{i=1}^k \kappa_i = 1$ und $\lambda_1, \dots, \lambda_l \geq 0$. Dann ist $c := \sum_{i=1}^l \lfloor \lambda_i \rfloor y_i$ ganzzahlig, also ist auch $x - c$ ganzzahlig. Ferner gilt $x - c = \sum_{i=1}^k \kappa_i x_i + \sum_{i=1}^l (\lambda_i - \lfloor \lambda_i \rfloor) y_i \in Q$, also ist $x - c = z_i$ für irgendein i . Damit folgt $(x, 1) = (c, 0) + (x - c, 1) \in C'$ und somit ist $Mx + d \leq 0$.

Um „ \supseteq “ zu zeigen, sei x ein Vektor mit $Mx \leq -d$, d.h. $(x, 1) \in C'$. Dann gibt es $\lambda_1, \dots, \lambda_l, \mu_1, \dots, \mu_m \geq 0$ mit $\sum_{i=1}^m \mu_i = 1$, so dass $x = \sum_{i=1}^l \lambda_i y_i + \sum_{i=1}^m \mu_i z_i$. O.B.d.A. können wir annehmen, dass $l \geq 1$ und $\mu_1 > 0$. Sei $\lambda'_i := \frac{l \lambda_i}{\mu_1}$ für $i = 1, \dots, l$. Dann ist $(z_1 + v y_i, 1) \in C$ für alle $i = 1, \dots, l$ und $v > 0$, und somit ist

$$x = \sum_{i=1}^l \frac{\mu_1}{l} (\lfloor \lambda'_i \rfloor + 1 - \lambda'_i) (z_1 + \lfloor \lambda'_i \rfloor y_i) \\ + \sum_{i=1}^l \frac{\mu_1}{l} (\lambda'_i - \lfloor \lambda'_i \rfloor) (z_1 + (\lfloor \lambda'_i \rfloor + 1) y_i) + \sum_{i=2}^m \mu_i z_i$$

eine Konvexitätskombination von ganzzahligen Punkten von P . □

Dies gilt im Allgemeinen nicht für irrationale Polyeder; siehe Aufgabe 1. Nach Satz 5.1 können wir eine Instanz von GANZZAHLIGE OPTIMIERUNG in der Form $\max\{c^\top x : x \in P_I\}$ schreiben, wobei $P = \{x : Ax \leq b\}$.

In Satz 5.9 (Abschnitt 5.1) werden wir eine Verallgemeinerung des Resultats von Meyer (Satz 5.1) beweisen. Nach einigen Vorbereitungen in Abschnitt 5.2 werden wir in den Abschnitten 5.3 und 5.4 Bedingungen betrachten, unter denen Polyeder ganzzahlig sind (d. h. $P = P_I$). Beachte, dass in diesem Fall das ganzzahlige LP mit seiner LP-Relaxierung (d. h. die Ganzzahligkeitsbedingungen werden weggelassen) äquivalent ist und deswegen in polynomieller Zeit gelöst werden kann. Wir werden dieser Situation in späteren Kapiteln mehrfach begegnen.

Im Allgemeinen ist die GANZZAHLIGE OPTIMIERUNG jedoch sehr viel schwieriger als die LINEARE PROGRAMMIERUNG, und polynomiale Algorithmen sind hier nicht bekannt. Dies ist aber auch nicht so überraschend, da wir viele anscheinend schwere Probleme als ganzzahlige LPs formulieren können. Trotzdem werden wir in Abschnitt 5.5 eine allgemeine Methode zur Bestimmung der ganzzahligen Hülle beschreiben, die durch wiederholtes Abschneiden von Teilen von $P \setminus P_I$ erfolgt. Obwohl wir damit keinen polynomiellem Algorithmus erhalten, ist diese Methode in einigen Fällen durchaus nützlich. Abschließend werden wir in Abschnitt 5.6 eine effiziente Methode zur Approximation des optimalen Zielfunktionswertes eines ganzzahligen LP beschreiben.

5.1 Die ganzzahlige Hülle eines Polyeders

Ganzzahlige LPs können wie LPs unzulässig oder auch unbeschränkt sein. Es ist jedoch nicht leicht zu entscheiden, ob $P_I = \emptyset$ für ein Polyeder P . Ist aber ein ganzzahliges LP zulässig, so kann man leicht entscheiden, ob es beschränkt ist: Man braucht nur seine LP-Relaxierung zu betrachten.

Proposition 5.2. *Sei $P = \{x : Ax \leq b\}$ ein rationales Polyeder mit nichtleerer ganzzahliger Hülle und c ein (nicht notwendigerweise rationaler) Vektor. Dann gilt: $\max\{cx : x \in P\}$ ist genau dann beschränkt, wenn $\max\{cx : x \in P_I\}$ beschränkt ist.*

Beweis: Angenommen, $\max\{cx : x \in P\}$ ist unbeschränkt. Dann folgt nach Korollar 3.28, dass das System $yA = c$, $y \geq 0$ keine Lösung hat. Nach Korollar 3.26 gibt es einen Vektor z mit $cz < 0$ und $Az \geq 0$. Damit ist das LP $\min\{cz : Az \geq 0, -\mathbb{1} \leq z \leq \mathbb{1}\}$ zulässig. Sei z^* eine optimale Basislösung dieses LP. Es ist z^* rational, da es Ecke eines rationalen Polytops ist. Multipliziere z^* mit einer geeigneten natürlichen Zahl, um einen ganzzahligen Vektor w mit $Aw \geq 0$ und $cw < 0$ zu erhalten. Sei $v \in P_I$ ein ganzzahliger Vektor. Dann ist $v - kw \in P_I$ für alle $k \in \mathbb{N}$, also ist $\max\{cx : x \in P_I\}$ unbeschränkt.

Die umgekehrte Richtung ist trivial. □

Man kann auf verschiedene Arten quantifizieren, wie stark sich P und P_I unterscheiden. Für Approximationsalgorithmen (die an späterer Stelle in diesem Buch behandelt werden) hat sich die folgende Notation sehr bewährt: Für $P \subseteq \mathbb{R}_+^n$

bezeichnen wir $\sup\left\{\frac{\min\{cx : x \in P_I\}}{\min\{cx : x \in P\}} : c \in \mathbb{R}_+^n\right\}$ als **Ganzzahligkeitslücke** von P . Diese Zahl kann auch auf andere Weise charakterisiert werden. Das folgende Ergebnis wird oft Carr und Vempala [2004] zugeschrieben, geht aber schon auf Goemans [1995] zurück.

Satz 5.3. *Sei $P \subseteq \mathbb{R}_+^n$ ein rationales Polyeder und $r \geq 0$. Dann gilt: Die Ganzzahligkeitslücke von P ist genau dann höchstens r , wenn für jedes $x \in P$ ein $y \in P_I$ mit $rx \geq y$ existiert.*

Beweis: „ \Leftarrow “: Angenommen, für jedes $x \in P$ gibt es ein $y \in P_I$ mit $rx \geq y$. Wir müssen zeigen, dass die Ganzzahligkeitslücke nicht größer als r ist. Sei dazu $c \in \mathbb{R}_+^n$ und x^* eine optimale Lösung des LPs $\min\{cx : x \in P\}$. Dann gibt es ein $y \in P_I$ mit $y \leq rx^*$ und somit $cy \leq r(cx^*)$.

„ \Rightarrow “: Wir nehmen an, dass es einen Vektor $x \in P$ gibt, so dass $rx \not\geq y$ für jedes $y \in P_I$ gilt. Um zu zeigen, dass die Ganzzahligkeitslücke größer als r ist, definieren wir $Q := \{y + z : y \in P_I, z \geq 0\}$. Dann gilt $rx \notin Q$, und, da Q nach Satz 5.1 ein Polyeder ist, gibt es einen Vektor w mit $w(rx) < \min\{wq : q \in Q\}$. Der Vektor w ist nichtnegativ (sonst wäre $\min\{wq : q \in Q\} = -\infty$), also gilt $r(wx) = w(rx) < \min\{wq : q \in Q\} = \min\{wy : y \in P_I\}$, und somit ist die Ganzzahligkeitslücke größer als r . \square

Als Nächstes suchen wir Schranken für den Abstand zwischen einer optimalen gebrochenen Lösung und einer optimalen ganzzahligen Lösung. Dazu benötigen wir einige Ergebnisse über rationale polyedrische Kegel.

Definition 5.4. *Sei A eine ganzzahlige Matrix. Eine **Unterdeterminante** von A ist die Determinante $\det B$ einer quadratischen Untermatrix B von A (definiert mittels beliebiger Zeilen- und Spaltenindizes). Wir schreiben $\Xi(A)$ für den größten aller Beträge der Unterdeterminanten von A .*

Lemma 5.5. *Sei $C = \{x : Ax \leq 0\}$ ein polyedrischer Kegel, wobei A eine ganzzahlige Matrix ist. Dann wird C von einer endlichen Menge ganzzahliger Vektoren erzeugt, deren Komponenten alle im Betrag kleiner oder gleich $\Xi(A)$ sind.*

Beweis: Nach Lemma 3.12 wird C erzeugt von einer Teilmenge der Vektoren y_1, \dots, y_t mit der Eigenschaft: für jedes i ist y_i die Lösung eines Systems $My = b'$, wobei M aus n linear unabhängigen Zeilen der Matrix (A) besteht und $b' = \pm e_j$ für einen bestimmten Einheitsvektor e_j . Setze $z_i := |\det M|y_i|$. Nach der Cramerschen Regel ist z_i ganzzahlig und $\|z_i\|_\infty \leq \Xi(A)$. Da dies für jedes i gilt, hat die Menge $\{z_1, \dots, z_t\}$ die gewünschten Eigenschaften. \square

Ein ähnliches Lemma werden wir im nächsten Abschnitt benötigen:

Lemma 5.6. *Jeder rationale polyedrische Kegel C wird erzeugt von einer endlichen Menge ganzzahliger Vektoren $\{a_1, \dots, a_t\}$ mit der Eigenschaft, dass jeder ganzzahlige Vektor in C eine nichtnegative ganzzahlige Linearkombination der a_1, \dots, a_t ist. (Eine solche Menge bezeichnet man als **Hilberthbasis** von C .)*

Beweis: Sei C erzeugt von den ganzzahligen Vektoren b_1, \dots, b_k . Seien a_1, \dots, a_t alle ganzzahligen Vektoren in dem Polytop

$$\{\lambda_1 b_1 + \dots + \lambda_k b_k : 0 \leq \lambda_i \leq 1 \quad (i = 1, \dots, k)\}.$$

Wir werden beweisen, dass $\{a_1, \dots, a_t\}$ eine Hilbertbasis von C ist. In der Tat erzeugen sie C , da die b_1, \dots, b_k unter den a_1, \dots, a_t sind.

Für jeden ganzzahligen Vektor $x \in C$ gibt es $\mu_1, \dots, \mu_k \geq 0$ mit

$$\begin{aligned} x = \mu_1 b_1 + \dots + \mu_k b_k &= \lfloor \mu_1 \rfloor b_1 + \dots + \lfloor \mu_k \rfloor b_k + \\ &\quad (\mu_1 - \lfloor \mu_1 \rfloor) b_1 + \dots + (\mu_k - \lfloor \mu_k \rfloor) b_k, \end{aligned}$$

also ist x eine nichtnegative ganzzahlige Linearkombination der a_1, \dots, a_t . \square

Eine wichtige grundlegende Eigenschaft ganzzahliger LPs ist, dass optimale ganzzahlige Lösungen und optimale gebrochene Lösungen nicht sehr weit von einander entfernt liegen. Dies wird in folgendem Satz präzisiert:

Satz 5.7. (Cook et al. [1986]) *Sei A eine ganzzahlige $m \times n$ -Matrix und seien $b \in \mathbb{R}^m$ und $c \in \mathbb{R}^n$ beliebige Vektoren. Sei $P := \{x : Ax \leq b\}$ mit $P_I \neq \emptyset$.*

- (a) *Angenommen, y ist eine optimale Lösung von $\max \{cx : x \in P\}$. Dann gibt es eine optimale ganzzahlige Lösung z von $\max \{cx : x \in P_I\}$ mit $\|z - y\|_\infty \leq n \Xi(A)$.*
- (b) *Angenommen, y ist eine zulässige ganzzahlige Lösung von $\max \{cx : x \in P_I\}$, aber keine optimale. Dann gibt es eine zulässige ganzzahlige Lösung $z \in P_I$ mit $cz > cy$ und $\|z - y\|_\infty \leq n \Xi(A)$.*

Beweis: Die Beweise der beiden Teile des Satzes sind fast identisch. Sei zunächst $y \in P$ beliebig. Sei $z^* \in P \cap \mathbb{Z}^n$ (a) eine optimale Lösung von $\max \{cx : x \in P_I\}$ (beachte, dass $P_I = \{x : Ax \leq \lfloor b \rfloor\}_I$ nach Satz 5.1 ein Polyeder ist, also wird das Maximum angenommen), oder (b) ein Vektor mit $cz^* > cy$.

Nun spalten wir das System $Ax \leq b$ in zwei Teilsysteme $A_1 x \leq b_1$ und $A_2 x \leq b_2$ auf, so dass $A_1 z^* \geq A_1 y$ und $A_2 z^* < A_2 y$. Dann liegt $z^* - y$ in dem polyedrischen Kegel $C := \{x : A_1 x \geq 0, A_2 x \leq 0\}$.

Es wird C von irgendwelchen Vektoren x_i ($i = 1, \dots, s$) erzeugt. Nach Lemma 5.5 können wir annehmen, dass x_i ganzzahlig ist und dass $\|x_i\|_\infty \leq \Xi(A)$ für alle i .

Da $z^* - y \in C$, gibt es nichtnegative Zahlen $\lambda_1, \dots, \lambda_s$ mit $z^* - y = \sum_{i=1}^s \lambda_i x_i$. Auch können wir annehmen, dass höchstens n der λ_i nicht verschwinden.

Für $\mu = (\mu_1, \dots, \mu_s)$ mit $0 \leq \mu_i \leq \lambda_i$ ($i = 1, \dots, s$) definieren wir nun

$$z_\mu := z^* - \sum_{i=1}^s \mu_i x_i = y + \sum_{i=1}^s (\lambda_i - \mu_i) x_i.$$

Beachte, dass $z_\mu \in P$: Aus der ersten Darstellung von z_μ folgt $A_1 z_\mu \leq A_1 z^* \leq b_1$ und aus der zweiten folgt $A_2 z_\mu \leq A_2 y \leq b_2$.

Fall 1: Es gibt ein $i \in \{1, \dots, s\}$ mit $\lambda_i \geq 1$ und $cx_i > 0$. Sei $z := y + x_i$. Dann folgt $cz > cy$, also kann dieser Fall nicht unter (a) vorkommen. Unter (b), wo y ganzzahlig ist, ist z eine ganzzahlige Lösung von $Ax \leq b$, für welche $cz > cy$ und $\|z - y\|_\infty = \|x_i\|_\infty \leq \Xi(A)$ gilt.

Fall 2: Für alle $i \in \{1, \dots, s\}$ mit $\lambda_i \geq 1$ gilt $cx_i \leq 0$. Sei

$$z := z_{\lfloor \lambda \rfloor} = z^* - \sum_{i=1}^s \lfloor \lambda_i \rfloor x_i.$$

Es ist z ein ganzzahliger Vektor von P mit $cz \geq cz^*$ und

$$\|z - y\|_\infty \leq \sum_{i=1}^s (\lambda_i - \lfloor \lambda_i \rfloor) \|x_i\|_\infty \leq n \Xi(A).$$

Also ist dieses z für (a) und auch für (b) der gesuchte Vektor. □

In folgendem Korollar zeigen wir, dass man die Größe von optimalen Lösungen ganzzahliger LPs beschränken kann:

Korollar 5.8. Ist $P = \{x \in \mathbb{Q}^n : Ax \leq b\}$ ein rationales Polyeder und hat $\max\{cx : x \in P_I\}$ eine optimale Lösung, dann hat es auch eine optimale ganzzahlige Lösung x mit $\text{size}(x) \leq 13n(\text{size}(A) + \text{size}(b))$.

Beweis: Nach Proposition 5.2 und Satz 4.4 hat $\max\{cx : x \in P\}$ eine optimale Lösung y mit $\text{size}(y) \leq 4n(\text{size}(A) + \text{size}(b))$. Nach Satz 5.7(a) gibt es eine optimale Lösung x von $\max\{cx : x \in P_I\}$ mit $\|x - y\|_\infty \leq n \Xi(A)$. Mit den Propositionen 4.1 und 4.3 haben wir

$$\begin{aligned} \text{size}(x) &\leq 2 \text{size}(y) + 2n \text{size}(n \Xi(A)) \\ &\leq 8n(\text{size}(A) + \text{size}(b)) + 2n \log n + 4n \text{size}(A) \\ &\leq 13n(\text{size}(A) + \text{size}(b)). \end{aligned}$$
□

Nach Satz 5.7(b) folgt: Für jede gegebene zulässige Lösung eines ganzzahligen LP kann man die Optimalität eines Vektors x auf einfache Weise dadurch prüfen, dass man $x + y$ für eine endliche Menge von Vektoren y , die nur von der Matrix A abhängen, testet. Eine solche endliche Testmenge (deren Existenz zuerst von Graver [1975] bewiesen wurde) erlaubt uns, einen grundlegenden Satz der ganzzahligen Optimierung zu beweisen:

Satz 5.9. (Wolsey [1981], Cook et al. [1986]) Für jede ganzzahlige $m \times n$ -Matrix A gibt es eine ganzzahlige Matrix M , deren Elemente im Betrage kleiner oder gleich $n^{2n} \Xi(A)^n$ sind, so dass es für jeden Vektor $b \in \mathbb{Q}^m$ einen rationalen Vektor d gibt, mit

$$\{x : Ax \leq b\}_I = \{x : Mx \leq d\}.$$

Beweis: Wir können annehmen, dass $A \neq 0$. Sei C der von den Zeilen von A erzeugte Kegel. Sei

$$L := \{z \in \mathbb{Z}^n : \|z\|_\infty \leq n\Xi(A)\}.$$

Für jedes $K \subseteq L$ betrachten wir nun den Kegel

$$C_K := C \cap \{y : zy \leq 0 \text{ für alle } z \in K\}.$$

Mit dem Beweis von Satz 3.29 und mit Lemma 5.5 folgt, dass $C_K = \{y : Uy \leq 0\}$ für eine ganzzahlige Matrix U (deren Zeilen Erzeugende von $\{x : Ax \leq 0\}$ und Elemente von K sind), deren Elemente im Betrage kleiner oder gleich $n\Xi(A)$ sind. Dann folgt, wiederum nach Lemma 5.5, dass es eine endliche Menge $G(K)$ ganzzahliger C_K erzeugender Vektoren gibt, die allesamt nur Komponenten im Betrage kleiner oder gleich $\Xi(U) \leq n!(n\Xi(A))^n \leq n^{2n}\Xi(A)^n$ haben.

Sei M die Matrix mit den Zeilen $\bigcup_{K \subseteq L} G(K)$. Da $C_\emptyset = C$, können wir annehmen, dass die Zeilen von A auch Zeilen von M sind.

Nun sei b ein fester Vektor. Hat $Ax \leq b$ keine Lösungen, so können wir b auf beliebige Weise zu einem Vektor d auffüllen und haben dann $\{x : Mx \leq d\} \subseteq \{x : Ax \leq b\} = \emptyset$.

Hat $Ax \leq b$ eine Lösung aber keine ganzzahlige Lösung, so setzen wir $b' := b - A'\mathbb{1}$, wobei A' aus A mittels Ersetzen eines jeden Elements durch seinen Betrag hervorgeht. Dann hat $Ax \leq b'$ keine Lösungen, da eine solche durch Rundung eine ganzzahlige Lösung von $Ax \leq b$ liefern würde. Wiederum füllen wir b' beliebig zu einem Vektor d auf.

Nun können wir annehmen, dass $Ax \leq b$ eine ganzzahlige Lösung hat. Für jedes $y \in C$ definieren wir

$$\delta_y := \max \{yx : Ax \leq b, x \text{ ganzzahlig}\}$$

(nach Korollar 3.28 ist dieses Maximum beschränkt, falls $y \in C$). Es genügt zu zeigen, dass

$$\{x : Ax \leq b\}_I = \left\{ x : yx \leq \delta_y \text{ für jedes } y \in \bigcup_{K \subseteq L} G(K) \right\}. \quad (5.1)$$

Der Schritt „ \subseteq “ ist trivial. Nun zeigen wir die Umkehrung. Sei c ein Vektor, für den

$$\max \{cx : Ax \leq b, x \text{ ganzzahlig}\}$$

beschränkt ist, und x^* ein Vektor, für den dieses Maximum angenommen wird. Wir werden zeigen, dass $cx \leq cx^*$ für jedes die Ungleichungen auf der rechten Seite von (5.1) erfüllende x .

Nach Proposition 5.2 ist das LP $\max \{cx : Ax \leq b\}$ beschränkt, somit folgt $c \in C$ nach Korollar 3.28.

Sei $\bar{K} := \{z \in L : A(x^* + z) \leq b\}$. Nach Definition ist $cz \leq 0$ für alle $z \in \bar{K}$, also ist $c \in C_{\bar{K}}$. Somit gibt es nichtnegative Zahlen λ_y ($y \in G(\bar{K})$), so dass

$$c = \sum_{y \in G(\bar{K})} \lambda_y y.$$

Nun behaupten wir, dass x^* für jedes $y \in G(\bar{K})$ eine optimale Lösung von

$$\max \{yx : Ax \leq b, x \text{ ganzzahlig}\}$$

ist: Das Gegenteil würde nach Satz 5.7(b) einen Vektor $z \in \bar{K}$ mit $yz > 0$ liefern. Dies ist aber unmöglich, da $y \in C_{\bar{K}}$. Daraus folgern wir, dass

$$\sum_{y \in G(\bar{K})} \lambda_y \delta_y = \sum_{y \in G(\bar{K})} \lambda_y y x^* = \left(\sum_{y \in G(\bar{K})} \lambda_y y \right) x^* = cx^*.$$

Somit ist die Ungleichung $cx \leq cx^*$ eine nichtnegative Linearkombination der Ungleichungen $yx \leq \delta_y$ für $y \in G(\bar{K})$. Damit ist (5.1) bewiesen. \square

Siehe Lasserre [2004] für ein ähnliches Resultat.

5.2 Unimodulare Transformationen

In diesem Abschnitt beweisen wir zwei Lemmata, die wir später benötigen. Eine quadratische Matrix heißt **unimodular**, wenn sie ganzzahlig und ihre Determinante gleich 1 oder -1 ist. Drei Typen von unimodularen Matrizen werden uns besonders interessieren: Sei $n \in \mathbb{N}$, $p \in \{1, \dots, n\}$ und $q \in \{1, \dots, n\} \setminus \{p\}$ und betrachte die drei folgendermaßen definierten Matrizen $(a_{ij})_{i,j \in \{1, \dots, n\}}$:

$$a_{ij} = \begin{cases} 1 & \text{für } i = j \neq p \\ -1 & \text{für } i = j = p \\ 0 & \text{sonst} \end{cases} \quad a_{ij} = \begin{cases} 1 & \text{für } i = j \notin \{p, q\} \\ 1 & \text{für } \{i, j\} = \{p, q\} \\ 0 & \text{sonst} \end{cases}$$

$$a_{ij} = \begin{cases} 1 & \text{für } i = j \\ -1 & \text{für } (i, j) = (p, q) \\ 0 & \text{sonst.} \end{cases}$$

Offensichtlich sind diese drei Matrizen unimodular. Ist U eine von ihnen, so ist das Ersetzen einer gegebenen Matrix A (mit n Spalten) durch AU äquivalent mit der Anwendung einer der folgenden elementaren Spaltenoperationen auf A :

- Multiplikation einer Spalte mit -1 ;
- Vertauschung zweier Spalten;
- Subtraktion einer Spalte von einer anderen Spalte.

Eine endliche Folge solcher Operationen heißt eine **unimodulare Transformation**. Offensichtlich ist das Produkt von unimodularen Matrizen wieder unimodular.

Man kann zeigen, dass eine Matrix genau dann unimodular ist, wenn sie aus der Einheitsmatrix durch Anwendung einer unimodularen Transformation hervorgeht (oder, äquivalent dazu, wenn sie das Produkt von Matrizen der drei obigen Typen ist); siehe Aufgabe 6. Diese Eigenschaft werden wir hier nicht benötigen.

Proposition 5.10. *Die Inverse einer unimodularen Matrix ist wieder unimodular. Für jede unimodulare Matrix U sind die Abbildungen $x \mapsto Ux$ und $x \mapsto xU$ Bijektionen auf \mathbb{Z}^n .*

Beweis: Sei U eine unimodulare Matrix. Nach der Cramerschen Regel ist die Inverse einer unimodularen Matrix ganzzahlig. Da $(\det U)(\det U^{-1}) = \det(UU^{-1}) = \det I = 1$, ist U^{-1} auch unimodular. Die zweite Aussage folgt sofort aus der ersten. \square

Lemma 5.11. *Für jede rationale Matrix A mit linear unabhängigen Zeilen gibt es eine unimodulare Matrix U , so dass AU die Form $(B \ 0)$ hat, wobei B eine nichtsinguläre quadratische Matrix ist.*

Beweis: Angenommen, wir haben eine unimodulare Matrix U gefunden, so dass

$$AU = \begin{pmatrix} B & 0 \\ C & D \end{pmatrix}$$

für eine nichtsinguläre quadratische Matrix B . (Am Anfang sind $U = I$ und $D = A$ und die Teile B , C und 0 haben keine Elemente.)

Sei $(\delta_1, \dots, \delta_k)$ die erste Zeile von D . Wende unimodulare Transformationen an, um zu erreichen, dass alle δ_i nichtnegativ sind und $\sum_{i=1}^k \delta_i$ minimal ist. O. B. d. A. können wir annehmen, dass $\delta_1 \geq \delta_2 \geq \dots \geq \delta_k$. Dann ist $\delta_1 > 0$, da die Zeilen von A (also auch die von AU) linear unabhängig sind. Wäre $\delta_2 > 0$, so würde die Subtraktion der zweiten Spalte in D von der ersten den Wert von $\sum_{i=1}^k \delta_i$ verringern. Also gilt $\delta_2 = \delta_3 = \dots = \delta_k = 0$. Somit können wir B um eine Zeile und eine Spalte erweitern und fortfahren. \square

Die konstruierte Matrix B ist in der Tat eine untere Diagonalmatrix. Mit etwas mehr Mühe kann man die sogenannte hermitesche Normalform von A bekommen. Das folgende Lemma liefert ein Kriterium für die ganzzahlige Lösbarkeit von linearen Gleichungssystemen, analog Farkas' Lemma.

Lemma 5.12. *Sei A eine rationale Matrix und b ein rationaler Spaltenvektor. Dann gilt: Das System $Ax = b$ hat genau dann eine ganzzahlige Lösung, wenn yb für jeden rationalen Vektor y mit yA ganzzahlig eine ganze Zahl ist.*

Beweis: Die Notwendigkeit ist einfach: Sind x und yA ganzzahlige Vektoren und gilt $Ax = b$, dann ist $yb = yAx$ eine ganze Zahl.

Um zu beweisen, dass die Bedingung auch hinreichend ist, nehmen wir an, dass yb stets eine ganze Zahl ist, wenn yA ganzzahlig ist. Wir können ferner annehmen, dass $Ax = b$ keine redundanten Gleichungen enthält, d. h. aus $yA = 0$

folgt $yb \neq 0$ für alle $y \neq 0$. Sei m die Anzahl der Zeilen von A . Ist $\text{rank}(A) < m$, so folgt: $\{y : yA = 0\}$ enthält einen nichtverschwindenden Vektor y' und der Vektor $y'' := \frac{1}{2yb}y'$ erfüllt $y''A = 0$ und $y''b = \frac{1}{2} \notin \mathbb{Z}$. Also sind die Zeilen von A linear unabhängig.

Nach Lemma 5.11 gibt es eine unimodulare Matrix U mit $AU = (B \ 0)$, wobei B eine nichtsinguläre $m \times m$ -Matrix ist. Da $B^{-1}AU = (I \ 0)$ eine ganzzahlige Matrix ist, folgt, dass yAU für jede Zeile y von B^{-1} ganzzahlig ist. Nach Proposition 5.10 ist dann yA ganzzahlig. Also ist yb eine ganze Zahl für jede Zeile y von B^{-1} , folglich ist $B^{-1}b$ ein ganzzahler Vektor. Somit ist $U \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ eine ganzzahlige Lösung von $Ax = b$. \square

5.3 Vollständige duale Ganzzahligkeit (TDI)

In diesem und dem nächsten Abschnitt geht es hauptsächlich um ganzzahlige Polyeder:

Definition 5.13. Ein Polyeder P ist **ganzzahlig**, falls $P = P_I$.

Satz 5.14. (Hoffman [1974], Edmonds und Giles [1977]) Sei P ein rationales Polyeder. Dann sind die folgenden sieben Aussagen äquivalent:

- (a) P ist ganzzahlig.
- (b) Jede Seitenfläche von P enthält ganzzahlige Vektoren.
- (c) Jede minimale Seitenfläche von P enthält ganzzahlige Vektoren.
- (d) Jede stützende Hyperebene von P enthält ganzzahlige Vektoren.
- (e) Jede rationale stützende Hyperebene von P enthält ganzzahlige Vektoren.
- (f) Für jedes c mit $\max \{cx : x \in P\}$ endlich, wird dieses Maximum mit einem ganzzahligen Vektor angenommen.
- (g) Für jedes ganzzahlige c mit $\max \{cx : x \in P\}$ endlich, ist dieses Maximum eine ganze Zahl.

Beweis: Als erstes beweisen wir (a) \Rightarrow (b) \Rightarrow (f) \Rightarrow (a), dann (b) \Rightarrow (d) \Rightarrow (e) \Rightarrow (c) \Rightarrow (b) und schließlich (f) \Rightarrow (g) \Rightarrow (e).

(a) \Rightarrow (b): Sei F eine Seitenfläche, etwa $F = P \cap H$, wobei H eine stützende Hyperebene ist. Sei ferner $x \in F$. Ist $P = P_I$, so ist x eine Konvexitätskombination von ganzzahligen Punkten in P und diese müssen in H , also auch in F liegen.

(b) \Rightarrow (f) folgt sofort aus Proposition 3.4, da für jedes c mit $\max \{cx : x \in P\}$ endlich die Menge $\{y \in P : cy = \max \{cx : x \in P\}\}$ eine Seitenfläche von P ist.

(f) \Rightarrow (a): Angenommen, es gäbe einen Vektor $y \in P \setminus P_I$. Da P_I nach Satz 5.1 ein Polyeder ist, gibt es somit eine für P_I gültige Ungleichung $ay \leq \beta$ mit $ay > \beta$. Damit wird (f) aber verletzt, weil $\max \{ax : x \in P\}$ (nach Proposition 5.2 endlich) mit keinem ganzzahligen Vektor angenommen wird.

(b) \Rightarrow (d) ist trivial, da der Durchschnitt einer stützenden Hyperebene mit P eine Seitenfläche von P ist. (d) \Rightarrow (e) und (c) \Rightarrow (b) sind beide auch trivial.

(e) \Rightarrow (c): Sei $P = \{x : Ax \leq b\}$. Wir können annehmen, dass A und b ganzzahlig sind. Sei $F = \{x : A'x = b'\}$ eine minimale Seitenfläche von P , wobei $A'x \leq b'$ ein Teilsystem von $Ax \leq b$ ist (mittels Proposition 3.9). Hat $A'x = b'$ keine ganzzahlige Lösung, dann gibt es nach Lemma 5.12 einen rationalen Vektor y , für den $c := yA'$ ganzzahlig aber $\delta := yb'$ keine ganze Zahl ist. Die Addition ganzer Zahlen zu den Komponenten von y ändert diese Eigenschaft nicht (A' und b' sind ganzzahlig), also können wir annehmen, dass alle Komponenten von y positiv sind. Beachte, dass $H := \{x : cx = \delta\}$ eine rationale Hyperebene ist, die keine ganzzahligen Vektoren enthält.

Schließlich zeigen wir noch, dass H eine stützende Hyperebene ist, indem wir $H \cap P = F$ beweisen. Da $F \subseteq H$ trivial ist, müssen wir nur noch $H \cap P \subseteq F$ zeigen. Für $x \in H \cap P$ haben wir aber $yA'x = cx = \delta = yb'$, folglich ist $y(A'x - b') = 0$. Da $y > 0$ und $A'x \leq b'$, haben wir $A'x = b'$, also ist $x \in F$.

(f) \Rightarrow (g) ist trivial, also bleibt nur noch (g) \Rightarrow (e) zu zeigen. Sei $H = \{x : cx = \delta\}$ eine rationale stützende Hyperebene von P , womit $\max\{cx : x \in P\} = \delta$ ist. Angenommen, H enthielte keine ganzzahligen Vektoren. Nach Lemma 5.12 gibt es dann eine Zahl γ mit γc ganzzahlig aber $\gamma \delta \notin \mathbb{Z}$. Damit folgt

$$\max\{(\lfloor \gamma \rfloor c)x : x \in P\} = \lfloor \gamma \rfloor \max\{cx : x \in P\} = \lfloor \gamma \rfloor \delta \notin \mathbb{Z},$$

im Widerspruch zu unserer Annahme. \square

Siehe auch Gomory [1963], Fulkerson [1971] und Chvátal [1973] zu früheren Teilresultaten. Mit (a) \Leftrightarrow (b) und Korollar 3.6 ist jede Seitenfläche eines ganzzahligen Polyeders ganzzahlig. Die Äquivalenz von (f) und (g) in Satz 5.14 regte Edmonds und Giles dazu an, sogenannte TDI-Systeme zu definieren:

Definition 5.15. (Edmonds und Giles [1977]) Ein System $Ax \leq b$ linearer Ungleichungen heißt vollständig dual ganzzahlig oder **TDI (totally dual integral)**, wenn das Minimum in der LP-Dualitätsgleichung

$$\max\{cx : Ax \leq b\} = \min\{yb : yA = c, y \geq 0\}$$

für jeden ganzzahligen Vektor c , für den das Minimum endlich ist, eine ganzzahlige optimale Lösung y hat.

Mit dieser Definition erhalten wir ein leichtes Korollar von Teil (g) \Rightarrow (a), Satz 5.14:

Korollar 5.16. Sei $Ax \leq b$ ein TDI-System mit A rational und b ganzzahlig. Dann ist das Polyeder $\{x : Ax \leq b\}$ ganzzahlig. \square

Es ist TDI aber keine Polyedereigenschaft (siehe Aufgabe 8). Im Allgemeinen enthält ein TDI-System mehr Ungleichungen als für die Bestimmung des Polyeders notwendig sind. Die Addition gültiger Ungleichungen zerstört nicht die Eigenschaft, TDI zu sein:

Proposition 5.17. Ist $Ax \leq b$ TDI und ist $ax \leq \beta$ eine gültige Ungleichung für $\{x : Ax \leq b\}$, so ist das System $Ax \leq b, ax \leq \beta$ auch TDI.

Beweis: Sei c ein ganzzahleriger Vektor, für den $\min \{yb + \gamma\beta : yA + \gamma a = c, y \geq 0, \gamma \geq 0\}$ endlich ist. Da $ax \leq \beta$ eine gültige Ungleichung für $\{x : Ax \leq b\}$ ist, folgt

$$\begin{aligned} \min \{yb : yA = c, y \geq 0\} &= \max \{cx : Ax \leq b\} \\ &= \max \{cx : Ax \leq b, ax \leq \beta\} \\ &= \min \{yb + \gamma\beta : yA + \gamma a = c, y \geq 0, \gamma \geq 0\}. \end{aligned}$$

Das erste Minimum wird mit einem ganzzahligen Vektor y^* angenommen, also ist $y = y^*, \gamma = 0$ eine ganzzahlige optimale Lösung für das zweite Minimum. \square

Satz 5.18. (Giles und Pulleyblank [1979]) Für jedes rationale Polyeder P gibt es ein rationales TDI-System $Ax \leq b$ mit A ganzzahlig und $P = \{x : Ax \leq b\}$. Hier kann b genau dann ganzzahlig gewählt werden, wenn P ganzzahlig ist.

Beweis: Sei $P = \{x : Cx \leq d\}$ mit C und d ganzzahlig. O. B. d. A. können wir annehmen, dass $P \neq \emptyset$. Für jede minimale Seitenfläche F von P sei

$$K_F := \{c : cz = \max \{cx : x \in P\} \text{ für alle } z \in F\}.$$

Nach Korollar 3.22 und Satz 3.29 ist K_F ein rationaler polyedrischer Kegel. Nach Lemma 5.6 gibt es dann eine ganzzahlige K_F erzeugende Hilbertbasis a_1, \dots, a_t . Sei \mathcal{S}_F das Ungleichungssystem

$$a_1x \leq \max \{a_1x : x \in P\}, \dots, a_tx \leq \max \{a_tx : x \in P\}.$$

Sei $Ax \leq b$ die Zusammenfügung aller solcher Systeme \mathcal{S}_F (für alle minimalen Seitenflächen F). Beachte: Ist P ganzzahlig, so auch b . Ferner gilt $P \subseteq \{x : Ax \leq b\}$.

Sei c ein ganzzahleriger Vektor mit $\max \{cx : x \in P\}$ endlich. Die Menge der dieses Maximum annehmenden Vektoren bildet eine Seitenfläche von P , also sei F eine minimale Seitenfläche mit $cz = \max \{cx : x \in P\}$ für alle $z \in F$. Sei \mathcal{S}_F das Ungleichungssystem $a_1x \leq \beta_1, \dots, a_tx \leq \beta_t$. Dann gilt $c = \lambda_1a_1 + \dots + \lambda_ta_t$ für irgendwelche nichtnegativen ganzen Zahlen $\lambda_1, \dots, \lambda_t$. Nun fügen wir den Zahlen $\lambda_1, \dots, \lambda_t$ Nullen hinzu, um einen ganzzahligen Vektor $\bar{\lambda} \geq 0$ mit $\bar{\lambda}A = c$ zu erhalten, womit $cx = (\bar{\lambda}A)x = \bar{\lambda}(Ax) \leq \bar{\lambda}b = \bar{\lambda}(Az) = (\bar{\lambda}A)z = cz$ für alle x mit $Ax \leq b$ und alle $z \in F$ folgt.

Wenden wir dies auf jede Zeile c von C an, so erhalten wir $Cx \leq d$ für alle x mit $Ax \leq b$; somit ist $P = \{x : Ax \leq b\}$. Ferner folgt für allgemeine c , dass $\bar{\lambda}$ eine optimale Lösung des dualen LP $\min \{yb : y \geq 0, yA = c\}$ ist. Somit ist $Ax \leq b$ TDI.

Ist P ganzzahlig, so ist b bereits ganzzahlig gewählt worden. Kann umgekehrt b ganzzahlig gewählt werden, so ist P nach Korollar 5.16 ganzzahlig. \square

In der Tat gibt es für volldimensionale rationale Polyeder ein eindeutig bestimmtes minimales das Polyeder bestimmendes TDI-System (Schrijver [1981]). Für spätere Zwecke beweisen wir noch, dass jede „Seitenfläche“ eines TDI-Systems wieder TDI ist:

Satz 5.19. (Cook [1983]) *Gegeben sei ein TDI-System $Ax \leq b$, $ax \leq \beta$ mit a ganzzahlig. Dann ist das System $Ax \leq b$, $ax = \beta$ auch TDI.*

Beweis: (Schrijver [1986]) Sei c ein ganzzahliger Vektor, für den

$$\begin{aligned} & \max \{cx : Ax \leq b, ax = \beta\} \\ &= \min \{yb + (\lambda - \mu)\beta : y, \lambda, \mu \geq 0, yA + (\lambda - \mu)a = c\} \end{aligned} \quad (5.2)$$

endlich ist. Seien $x^*, y^*, \lambda^*, \mu^*$ Vektoren, mit denen diese Optima angenommen werden. Wir setzen nun $c' := c + \lceil \mu^* \rceil a$ und bemerken, dass

$$\max \{c'x : Ax \leq b, ax \leq \beta\} = \min \{yb + \lambda\beta : y, \lambda \geq 0, yA + \lambda a = c'\} \quad (5.3)$$

endlich ist, da $x := x^*$ für das Maximum zulässig ist und $y := y^*$, $\lambda := \lambda^* + \lceil \mu^* \rceil - \mu^*$ für das Minimum.

Da das System $Ax \leq b$, $ax \leq \beta$ TDI ist, hat das Minimum in (5.3) eine ganzzahlige optimale Lösung $\tilde{y}, \tilde{\lambda}$. Schließlich setzen wir noch $y := \tilde{y}$, $\lambda := \tilde{\lambda}$ und $\mu := \lceil \mu^* \rceil$ und behaupten, dass (y, λ, μ) eine ganzzahlige optimale Lösung für das Minimum in (5.2) ist.

Offensichtlich ist (y, λ, μ) zulässig für das Minimum in (5.2). Ferner gilt

$$\begin{aligned} yb + (\lambda - \mu)\beta &= \tilde{y}b + \tilde{\lambda}\beta - \lceil \mu^* \rceil \beta \\ &\leq y^*b + (\lambda^* + \lceil \mu^* \rceil - \mu^*)\beta - \lceil \mu^* \rceil \beta, \end{aligned}$$

da $(y^*, \lambda^* + \lceil \mu^* \rceil - \mu^*)$ zulässig für das Minimum in (5.3) ist und $(\tilde{y}, \tilde{\lambda})$ eine optimale Lösung ist. Damit folgt

$$yb + (\lambda - \mu)\beta \leq y^*b + (\lambda^* - \mu^*)\beta,$$

womit bewiesen ist, dass (y, λ, μ) eine ganzzahlige optimale Lösung für das Minimum in (5.2) ist. \square

Die folgenden Aussagen sind allesamt leichte Folgerungen der obigen Definition von TDI-Systemen: Ein System $Ax = b$, $x \geq 0$ ist TDI, falls $\min \{yb : yA \geq c\}$ für jeden ganzzahligen Vektor c , für den das Minimum endlich ist, eine ganzzahlige optimale Lösung y hat. Ein System $Ax \leq b$, $x \geq 0$ ist TDI, falls $\min \{yb : yA \geq c, y \geq 0\}$ für jeden ganzzahligen Vektor c , für den das Minimum endlich ist, eine ganzzahlige optimale Lösung y hat. Man kann fragen, ob es Matrizen A gibt, so dass das System $Ax \leq b$, $x \geq 0$ für jeden ganzzahligen Vektor b TDI ist. Es wird sich herausstellen, dass dies gerade die vollständig-unimodularen Matrizen sind.

5.4 Vollständig-unimodulare Matrizen

Definition 5.20. Eine Matrix A heißt **vollständig-unimodular**, falls jede Unterdeterminante von A gleich 0, +1 oder -1 ist.

Insbesondere muss jedes Element einer vollständig-unimodularen Matrix gleich 0, +1 oder -1 sein. Das Hauptresultat dieses Abschnitts ist:

Satz 5.21. (Hoffman und Kruskal [1956]) Eine ganzzahlige Matrix A ist genau dann vollständig-unimodular, wenn das Polyeder $\{x : Ax \leq b, x \geq 0\}$ für jeden ganzzahligen Vektor b ganzzahlig ist.

Beweis: Sei A eine $m \times n$ -Matrix und $P := \{x : Ax \leq b, x \geq 0\}$. Beachte, dass die minimalen Seitenflächen von P Ecken sind.

Zum Beweis der Notwendigkeit nehmen wir an, dass A vollständig-unimodular ist. Sei b ein ganzzahliger Vektor und x eine Ecke von P . Es ist x die Lösung von $A'x = b'$ für irgendein Teilsystem $A'x \leq b'$ von $\begin{pmatrix} A \\ -I \end{pmatrix}x \leq \begin{pmatrix} b \\ 0 \end{pmatrix}$, wobei A' eine nichtsinguläre $n \times n$ -Matrix ist. Da A vollständig-unimodular ist, folgt $|\det A'| = 1$. Mit der Cramerschen Regel ist dann $x = (A')^{-1}b'$ ganzzahlig.

Nun beweisen wir, dass die Bedingung hinreichend ist. Angenommen, die Ecken von P sind ganzzahlig für jeden ganzzahligen Vektor b . Sei A' eine nichtsinguläre $k \times k$ -Untermatrix von A . Wir müssen zeigen, dass $|\det A'| = 1$. O. B. d. A. können wir annehmen, dass A' die Elemente der ersten k Zeilen und Spalten von A enthält.

Betrachte die ganzzahlige $m \times m$ -Matrix B , welche aus den ersten k und den letzten $m - k$ Spalten von $(A \ I)$ besteht (siehe Abb. 5.2). Offensichtlich gilt $|\det B| = |\det A'|$.

Um $|\det B| = 1$ zu beweisen, werden wir zeigen, dass B^{-1} ganzzahlig ist. Da $\det B \det B^{-1} = 1$, folgt dann $|\det B| = 1$ und wir sind fertig.

Sei $i \in \{1, \dots, m\}$. Wir beweisen nun, dass $B^{-1}e_i$ ganzzahlig ist. Wähle einen ganzzahligen Vektor y mit $z := y + B^{-1}e_i \geq 0$. Dann ist $b := Bz = By + e_i$ ganzzahlig. Nun fügen wir dem Vektor z Nullen hinzu, um einen Vektor z' mit

$$\begin{pmatrix} A & I \end{pmatrix} z' = Bz = b$$

zu erhalten. Es folgt, dass der aus den ersten n Komponenten von z' bestehende Vektor z'' in P liegt. Ferner werden n linear unabhängige Nebenbedingungen mit Gleichheit erfüllt, nämlich die ersten k und die letzten $n - k$ Ungleichungen des Systems

$$\begin{pmatrix} A \\ -I \end{pmatrix} z'' \leq \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Also ist z'' eine Ecke von P . Wegen unserer Annahme ist z'' ganzzahlig. Aber dann ist z' auch ganzzahlig: Seine ersten n Komponenten sind die Komponenten von z'' und seine letzten m Komponenten die Schlupfvariablen $b - Az''$ (und A und b sind ganzzahlig). Somit ist z auch ganzzahlig, folglich ist $B^{-1}e_i = z - y$ ganzzahlig. \square

	k	$n - k$	k	$m - k$	
k	A'		I	0	
$m - k$			0		I
		0	0		
					z'
					$\underbrace{z''}_{z''}$

Abbildung 5.2.

Der obige Beweis stammt von Veinott und Dantzig [1968].

Korollar 5.22. Eine ganzzahlige Matrix A ist genau dann vollständig-unimodular, wenn für alle ganzzahligen Vektoren b und c beide Optima in der LP-Dualitätsgleichung

$$\max \{cx : Ax \leq b, x \geq 0\} = \min \{yb : y \geq 0, yA \geq c\}$$

mit ganzzahligen Vektoren angenommen werden (falls die Optima endlich sind).

Beweis: Dieses Resultat folgt aus dem Satz von Hoffman und Kruskal (Satz 5.21) zusammen mit der einfachen Tatsache, dass die Transponierte einer vollständig-unimodularen Matrix wieder vollständig-unimodular ist. \square

Diese Aussagen kann man auch mittels TDI formulieren:

Korollar 5.23. Eine ganzzahlige Matrix A ist genau dann vollständig-unimodular, wenn das System $Ax \leq b, x \geq 0$ für jeden Vektor b TDI ist.

Beweis: Sei A (also auch A^\top) vollständig-unimodular. Dann folgt nach dem Satz von Hoffman und Kruskal, dass $\min \{yb : yA \geq c, y \geq 0\}$ für jeden Vektor b und jeden ganzzahligen Vektor c , für die das Minimum endlich ist, mit einem ganzzahligen Vektor angenommen wird. Mit anderen Worten, das System $Ax \leq b, x \geq 0$ ist TDI für jeden Vektor b .

Zum Beweis der Umkehrung nehmen wir an, dass das System $Ax \leq b, x \geq 0$ für jeden ganzzahligen Vektor b TDI ist. Mit Korollar 5.16 folgt dann, dass das Polyeder $\{x : Ax \leq b, x \geq 0\}$ für jeden ganzzahligen Vektor b ganzzahlig ist. Mit Satz 5.21 folgt daraus, dass A vollständig-unimodular ist. \square

Dies ist nicht der einzige Weg, um mit dem Begriff der vollständigen Unimodularität zu zeigen, dass ein System TDI ist. Das folgende Lemma beinhaltet eine weitere Beweismethode, welche wir später einige Male verwenden werden (in den Sätzen 6.14, 14.12 und 19.17).

Lemma 5.24. *Sei $Ax \leq b, x \geq 0$ ein lineares Ungleichungssystem, wobei $A \in \mathbb{R}^{m \times n}$ und $b \in \mathbb{R}^m$. Angenommen, für jedes $c \in \mathbb{Z}^n$, für welches $\min\{yb : yA \geq c, y \geq 0\}$ eine optimale Lösung hat, hat es auch eine optimale Lösung y^* mit der Eigenschaft, dass die den nichtverschwindenden Komponenten von y^* entsprechenden Zeilen von A eine vollständig-unimodulare Matrix bilden. Dann folgt, dass das System $Ax \leq b, x \geq 0$ TDI ist.*

Beweis: Sei $c \in \mathbb{Z}^n$ und y^* eine optimale Lösung von $\min\{yb : yA \geq c, y \geq 0\}$ mit der Eigenschaft, dass die den nichtverschwindenden Komponenten von y^* entsprechenden Zeilen von A eine vollständig-unimodulare Matrix A' bilden. Wir behaupten, dass

$$\min\{yb : yA \geq c, y \geq 0\} = \min\{yb' : yA' \geq c, y \geq 0\}, \quad (5.4)$$

wobei b' der Vektor der den Zeilen von A' entsprechenden Komponenten von b ist. Um die Ungleichung „ \leq “ von (5.4) zu zeigen, beachte man, dass das rechte LP aus dem linken LP durch Nullsetzung einiger Variablen entsteht. Die Ungleichung „ \geq “ folgt aus der Tatsache, dass der Vektor y^* ohne seine verschwindenden Komponenten eine zulässige Lösung des rechten LP ist.

Da A' vollständig-unimodular ist, hat das zweite Minimum in (5.4) nach dem Satz von Hoffman und Kruskal (Satz 5.21) eine ganzzahlige optimale Lösung. Füllt man diese Lösung mit Nullen auf, so erhält man eine ganzzahlige optimale Lösung des ersten Minimums in (5.4), womit der Beweis abgeschlossen ist. \square

Ein sehr nützliches Kriterium für die vollständige Unimodularität ist im folgenden Satz enthalten:

Satz 5.25. (Ghouila-Houri [1962]) *Eine Matrix $A = (a_{ij}) \in \mathbb{Z}^{m \times n}$ ist genau dann vollständig-unimodular, wenn es für jedes $R \subseteq \{1, \dots, m\}$ Mengen R_1 und R_2 gibt mit $R = R_1 \dot{\cup} R_2$ und*

$$\sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij} \in \{-1, 0, 1\}$$

für alle $j = 1, \dots, n$.

Beweis: Sei A vollständig-unimodular und $R \subseteq \{1, \dots, m\}$. Sei ferner $d_r := 1$ für $r \in R$ und $d_r := 0$ für $r \in \{1, \dots, m\} \setminus R$. Die Matrix $\begin{pmatrix} A^\top \\ -A^\top \\ I \end{pmatrix}$ ist auch vollständig-unimodular, also ist das Polytop

$$\left\{ x : xA \leq \left\lceil \frac{1}{2}dA \right\rceil, xA \geq \left\lfloor \frac{1}{2}dA \right\rfloor, x \leq d, x \geq 0 \right\}$$

nach Satz 5.21 ganzzahlig. Es ist ferner nicht leer, da es $\frac{1}{2}d$ enthält. Also hat es eine ganzzahlige Ecke, etwa z . Setzen wir $R_1 := \{r \in R : z_r = 0\}$ und $R_2 := \{r \in R : z_r = 1\}$, so erhalten wir

$$\left(\sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij} \right)_{1 \leq j \leq n} = (d - 2z)A \in \{-1, 0, 1\}^n,$$

wie erwünscht.

Nun beweisen wir die Umkehrung. Mittels Induktion über k beweisen wir, dass jede $k \times k$ -Untermatrix Determinante 0, 1 oder -1 hat. Für $k = 1$ folgt dies sofort aus dem Kriterium für $|R| = 1$.

Nun sei $k > 1$ und $B = (b_{ij})_{i,j \in \{1, \dots, k\}}$ eine nichtsinguläre $k \times k$ -Untermatrix von A . Mit der Cramerschen Regel folgt, dass jedes Element von B^{-1} gleich $\frac{\det B'}{\det B}$ ist, wobei B' aus B mittels Ersetzen einer Spalte durch einen Einheitsvektor hervorgeht. Nach der Induktionsvoraussetzung ist $\det B' \in \{-1, 0, 1\}$. Somit ist $B^* := (\det B)B^{-1}$ eine Matrix, deren Elemente nur $-1, 0$ oder 1 sind.

Sei b_1^* die erste Zeile von B^* . Es gilt $b_1^*B = (\det B)e_1$, wobei e_1 der erste Einheitsvektor ist. Sei $R := \{i : b_{1i}^* \neq 0\}$. Dann haben wir für $j = 2, \dots, k$: $0 = (b_1^*B)_j = \sum_{i \in R} b_{1i}^* b_{ij}$, somit ist $|\{i \in R : b_{ij} \neq 0\}|$ gerade.

Nach Voraussetzung gibt es eine Partition $R = R_1 \dot{\cup} R_2$ mit $\sum_{i \in R_1} b_{ij} - \sum_{i \in R_2} b_{ij} \in \{-1, 0, 1\}$ für alle j . Also gilt $\sum_{i \in R_1} b_{ij} - \sum_{i \in R_2} b_{ij} = 0$ für alle $j = 2, \dots, k$. Gilt auch $\sum_{i \in R_1} b_{1i} - \sum_{i \in R_2} b_{1i} = 0$, so ist die Summe der R_1 -Zeilen gleich der Summe der R_2 -Zeilen, im Widerspruch zu der Annahme, dass B nichtsingulär ist (da $R \neq \emptyset$).

Also ist $\sum_{i \in R_1} b_{1i} - \sum_{i \in R_2} b_{1i} \in \{-1, 1\}$ und wir haben $yB \in \{e_1, -e_1\}$, wobei

$$y_i := \begin{cases} 1 & \text{für } i \in R_1 \\ -1 & \text{für } i \in R_2 \\ 0 & \text{für } i \notin R. \end{cases}$$

Da $b_1^*B = (\det B)e_1$ und B nichtsingulär ist, folgt $b_1^* \in \{(\det B)y, -(\det B)y\}$. Da sowohl y als auch b_1^* nicht verschwindende Vektoren sind, dessen Komponenten nur $-1, 0$ oder 1 sind, folgt, dass $|\det B| = 1$. \square

Dieses Kriterium wenden wir nun auf Inzidenzmatrizen von Graphen an:

Satz 5.26. *Die Inzidenzmatrix eines ungerichteten Graphen G ist genau dann vollständig-unimodular, wenn G bipartit ist.*

Beweis: Nach Satz 5.25 ist die Inzidenzmatrix M von G genau dann vollständig-unimodular, wenn es für jede Menge $X \subseteq V(G)$ eine Partition $X = A \dot{\cup} B$ mit $E(G[A]) = E(G[B]) = \emptyset$ gibt. Nach Definition gibt es eine solche Partition genau dann, wenn $G[X]$ bipartit ist. \square

Satz 5.27. *Die Inzidenzmatrix eines jeden Digraphen ist vollständig-unimodular.*

Beweis: Nach Satz 5.25 genügt es, $R_1 := R$ und $R_2 := \emptyset$ für jede Teilmenge $R \subseteq V(G)$ zu setzen. \square

Anwendungen der Sätze 5.26 und 5.27 werden wir in späteren Kapiteln besprechen. Es gibt eine interessante Erweiterung von Satz 5.27 auf kreuzungsfreie Familien:

Definition 5.28. Sei G ein Digraph und \mathcal{F} eine Familie von Teilmengen von $V(G)$.

Die Einweg-Schnitt-Inzidenz-Matrix von \mathcal{F} ist die

Matrix $M = (m_{X,e})_{X \in \mathcal{F}, e \in E(G)}$, wobei

$$m_{X,e} = \begin{cases} 1 & \text{für } e \in \delta^+(X) \\ 0 & \text{für } e \notin \delta^+(X). \end{cases}$$

Die Zweiwege-Schnitt-Inzidenz-Matrix von \mathcal{F} ist die

Matrix $M = (m_{X,e})_{X \in \mathcal{F}, e \in E(G)}$, wobei

$$m_{X,e} = \begin{cases} -1 & \text{für } e \in \delta^-(X) \\ 1 & \text{für } e \in \delta^+(X) \\ 0 & \text{sonst.} \end{cases}$$

Satz 5.29. Sei G ein Digraph und $(V(G), \mathcal{F})$ ein kreuzungsfreies Mengensystem. Dann ist die Zweiwege-Schnitt-Inzidenz-Matrix von \mathcal{F} vollständig-unimodular. Ist \mathcal{F} laminar, so ist auch die Einweg-Schnitt-Inzidenz-Matrix von \mathcal{F} vollständig-unimodular.

Beweis: Sei \mathcal{F} eine kreuzungsfreie Familie von Teilmengen von $V(G)$. Zunächst betrachten wir den Fall \mathcal{F} laminar.

Wir verwenden hier Satz 5.25. Um zu sehen, dass das Kriterium erfüllt ist, betrachten wir für gegebenes $\mathcal{R} \subseteq \mathcal{F}$ die Baumdarstellung (T, φ) von \mathcal{R} , wobei T eine Arboreszenz mit Wurzel r ist (Proposition 2.14). In der Notation von Definition 2.13 ist $\mathcal{R} = \{S_e : e \in E(T)\}$. Setze $\mathcal{R}_1 := \{S_{(v,w)} \in \mathcal{R} : \text{dist}_T(r, w) \text{ ist gerade}\}$ und $\mathcal{R}_2 := \mathcal{R} \setminus \mathcal{R}_1$. Für jede Kante $f \in E(G)$ bilden nun die Kanten $e \in E(T)$ mit $f \in \delta^+(S_e)$ einen Weg P_f in T (womöglich mit Länge Null). Damit folgt

$$|\{X \in \mathcal{R}_1 : f \in \delta^+(X)\}| - |\{X \in \mathcal{R}_2 : f \in \delta^+(X)\}| \in \{-1, 0, 1\},$$

wie für die Einweg-Schnitt-Inzidenz-Matrix erwünscht.

Ferner haben wir: Für jede Kante f bilden die Kanten $e \in E(T)$ mit $f \in \delta^-(S_e)$ einen Weg Q_f in T . Da P_f und Q_f einen gemeinsamen Endknoten haben, folgt

$$\begin{aligned} & |\{X \in \mathcal{R}_1 : f \in \delta^+(X)\}| - |\{X \in \mathcal{R}_2 : f \in \delta^+(X)\}| \\ & - |\{X \in \mathcal{R}_1 : f \in \delta^-(X)\}| + |\{X \in \mathcal{R}_2 : f \in \delta^-(X)\}| \in \{-1, 0, 1\}, \end{aligned}$$

wie für die Zweiwege-Schnitt-Inzidenz-Matrix erwünscht.

Ist nun $(V(G), \mathcal{F})$ ein allgemeines kreuzungsfreies Mengensystem, so sei

$$\mathcal{F}' := \{X \in \mathcal{F} : r \notin X\} \cup \{V(G) \setminus X : X \in \mathcal{F}, r \in X\}$$

für ein festes $r \in V(G)$. Es ist \mathcal{F}' laminar. Da die Zweiwege-Schnitt-Inzidenz-Matrix von \mathcal{F} eine Untermatrix von $(\begin{smallmatrix} M \\ -M \end{smallmatrix})$ ist, wobei M die Zweiwege-Schnitt-Inzidenz-Matrix von \mathcal{F}' ist, so folgt, dass auch sie vollständig-unimodular ist. \square

Für allgemeine kreuzungsfreie Familien ist die Einweg-Schnitt-Inzidenz-Matrix im Allgemeinen nicht vollständig-unimodular; siehe Aufgabe 13. Für eine notwendige und hinreichende Bedingung siehe Schrijver [1983]. Die Zweiwege-Schnitt-Inzidenz-Matrizen von kreuzungsfreien Familien werden auch Netzwerkmatrizen genannt (siehe Aufgabe 14).

Seymour [1980] hat bewiesen, dass alle vollständig-unimodularen Matrizen auf eine bestimmte Weise aus Netzwerkmatrizen und zwei weiteren vollständig-unimodularen Matrizen konstruiert werden können. Aus diesem tiefen Resultat folgt dann ein polynomieller Algorithmus zur Entscheidung, ob eine gegebene Matrix vollständig-unimodular ist (siehe Schrijver [1986]).

5.5 Schnittebenen

In den vorangegangenen Abschnitten haben wir ganzzahlige Polyeder betrachtet. Für allgemeine Polyeder P gilt $P \supset P_I$. Möchten wir ein ganzzahliges LP $\max \{cx : x \in P_I\}$ lösen, so ist es nahe liegend, gewisse ausgewählte Stücke von P wegzuschneiden, so dass die übrig bleibende Menge wieder ein Polyeder P' ist und $P \supset P' \supset P_I$ gilt. Man hofft, dass $\max \{cx : x \in P'\}$ mit einem ganzzahligen Vektor angenommen wird; falls nicht, so wendet man diese Wegschneide-Operation auf P' an, um ein Polyeder P'' zu erhalten, und so weiter. Dies ist die grundlegende Idee der sogenannten Schnittebenenmethode, welche zuerst von Dantzig, Fulkerson und Johnson [1954] zur Lösung eines speziellen Problems (das TRAVELING-SALESMAN-PROBLEM (TSP)) eingeführt wurde.

Gomory [1958, 1963] hat einen Algorithmus beschrieben, welcher allgemeine ganzzahlige LPs mittels der Schnittebenenmethode löst. In diesem Abschnitt beschränken wir uns jedoch auf den theoretischen Hintergrund. Gomorys Algorithmus läuft nicht in polynomieller Zeit und ist in seiner ursprünglichen Form kaum von praktischem Nutzen. Die allgemeine Idee, Schnittebenen zu gebrauchen, ist aber weit verbreitet und in der Praxis recht erfolgreich. Dieses Thema werden wir in Abschnitt 21.6 behandeln. Die folgende Einführung basiert hauptsächlich auf Schrijver [1986].

Definition 5.30. Sei $P \subseteq \mathbb{R}^n$ eine konvexe Menge. Dann definieren wir

$$P' := \bigcap_{P \subseteq H} H_I,$$

wobei der Durchschnitt über alle P enthaltenden rationalen affinen Halbräume $H = \{x : cx \leq \delta\}$ läuft. Wir setzen $P^{(0)} := P$ und $P^{(i+1)} := (P^{(i)})'$. Dann heißt $P^{(i)}$ die i -te **Gomory-Chvátal-Stutzung** von P .

Ist P ein rationales Polyeder, so ist P' , wie wir unten zeigen werden, auch ein rationales Polyeder. Somit gilt $P \supseteq P' \supseteq P^{(2)} \supseteq \dots \supseteq P_I$ und $P_I = (P')_I$. Wir weisen darauf hin, dass Dadush, Dey, und Vielma [2014] Folgendes bewiesen haben: Ist P eine kompakte konvexe Menge, so ist P' ein rationales Polytop.

Proposition 5.31. Für jedes rationale Polyeder $P = \{x : Ax \leq b\}$ gilt

$$P' = \{x : uAx \leq \lfloor ub \rfloor \text{ für alle } u \geq 0 \text{ mit } uA \text{ ganzzahlig}\}.$$

Beweis: Zunächst weisen wir auf zwei Fakten hin. Für jeden rationalen affinen Halbraum $H = \{x : cx \leq \delta\}$ mit c ganzzahlig gilt offensichtlich

$$H' = H_I \subseteq \{x : cx \leq \lfloor \delta \rfloor\}. \quad (5.5)$$

Sind auch noch die Komponenten von c teilerfremd (d. h. ihr größter gemeinsamer Teiler ist 1) so behaupten wir, dass

$$H' = H_I = \{x : cx \leq \lfloor \delta \rfloor\}. \quad (5.6)$$

Zum Beweis von (5.6): Sei c ein ganzzahiger Vektor mit teilerfremden Komponenten. Nach Lemma 5.12 enthält die Hyperebene $\{x : cx = \lfloor \delta \rfloor\}$ einen ganzzahligen Vektor y . Für irgendeinen rationalen Vektor $x \in \{x : cx \leq \lfloor \delta \rfloor\}$ nehme man ein $\alpha \in \mathbb{N}$ mit αx ganzzahlig. Dann folgt

$$x = \frac{1}{\alpha}(ax - (\alpha - 1)y) + \frac{\alpha - 1}{\alpha}y,$$

d. h. x ist eine Konvexitätskombination von ganzzahligen Punkten in H . Also ist $x \in H_I$, woraus (5.6) folgt.

Nun wenden wir uns dem Hauptteil des Beweises zu. Um „ \subseteq “ zu beweisen, beachte man, dass $\{x : uAx \leq ub\}$ für jedes $u \geq 0$ ein P enthaltender Halbraum ist, also folgt $P' \subseteq \{x : uAx \leq \lfloor ub \rfloor\}$ nach (5.5), falls uA ganzzahlig ist.

Nun beweisen wir „ \supseteq “. Für $P = \emptyset$ ist dies einfach, also nehmen wir an, dass $P \neq \emptyset$. Sei $H = \{x : cx \leq \delta\}$ ein rationaler affiner P enthaltender Halbraum. O. B. d. A. können wir annehmen, dass c ganzzahlig ist und dass die Komponenten von c teilerfremd sind. Es gilt

$$\delta \geq \max \{cx : Ax \leq b\} = \min \{ub : uA = c, u \geq 0\}.$$

Sei u^* eine optimale Lösung des Minimums. Dann folgt für ein beliebiges $z \in \{x : uAx \leq \lfloor ub \rfloor \text{ für alle } u \geq 0 \text{ mit } uA \text{ ganzzahlig}\} \subseteq \{x : u^*Ax \leq \lfloor u^*b \rfloor\}$,

dass

$$cz = u^*Az \leq \lfloor u^*b \rfloor \leq \lfloor \delta \rfloor.$$

Mit (5.6) folgt daraus, dass $z \in H_I$. □

Weiter unten werden wir beweisen, dass es für ein beliebiges rationales Polyeder P eine Zahl t mit $P_I = P^{(t)}$ gibt. Also löst Gomorys Schnittebenenmethode

schrittweise LPs über P, P', P'' und so weiter, bis das Optimum ganzzahlig ist. Es werden in jeder Iteration nur endlich viele neue Ungleichungen hinzugefügt, nämlich diejenigen, die einem das aktuelle Polyeder definierenden TDI-System entsprechen (siehe Satz 5.18):

Satz 5.32. (Schrijver [1980]) *Sei $P = \{x : Ax \leq b\}$ ein Polyeder mit $Ax \leq b$ TDI, A ganzzahlig und b rational. Dann gilt $P' = \{x : Ax \leq \lfloor b \rfloor\}$. Insbesondere gilt für rationale Polyeder P , dass P' auch ein rationales Polyeder ist.*

Beweis: Für P leer ist die Aussage trivial, also sei $P \neq \emptyset$. Offensichtlich ist $P' \subseteq \{x : Ax \leq \lfloor b \rfloor\}$. Zum Beweis der umgekehrten Inklusion sei $u \geq 0$ ein Vektor mit uA ganzzahlig. Nach Proposition 5.31 genügt es zu zeigen, dass $uAx \leq \lfloor ub \rfloor$ für alle x mit $Ax \leq \lfloor b \rfloor$.

Wir wissen, dass

$$ub \geq \max \{uAx : Ax \leq b\} = \min \{yb : y \geq 0, yA = uA\}.$$

Da $Ax \leq b$ TDI ist, wird das Minimum mit einem ganzzahligen Vektor y^* angenommen. Mit $Ax \leq \lfloor b \rfloor$ folgt dann

$$uAx = y^*Ax \leq y^*\lfloor b \rfloor \leq \lfloor y^*b \rfloor \leq \lfloor ub \rfloor.$$

Die zweite Aussage folgt mittels Satz 5.18. □

Um den Hauptsatz dieses Abschnitts beweisen zu können, benötigen wir noch zwei Lemmata.

Lemma 5.33. *Ist F eine Seitenfläche eines rationalen Polyeders P , so gilt $F' = P' \cap F$. Allgemeiner haben wir $F^{(i)} = P^{(i)} \cap F$ für alle $i \in \mathbb{N}$.*

Beweis: Sei $P = \{x : Ax \leq b\}$ mit A ganzzahlig, b rational und $Ax \leq b$ TDI (siehe Satz 5.18).

Nun sei $F = \{x : Ax \leq b, ax = \beta\}$ eine Seitenfläche von P , wobei $ax \leq \beta$ mit a und β ganzzahlig eine gültige Ungleichung für P ist.

Nach Proposition 5.17 ist das System $Ax \leq b, ax \leq \beta$ TDI, also folgt nach Satz 5.19, dass das System $Ax \leq b, ax = \beta$ auch TDI ist. Da β eine ganze Zahl ist, folgt

$$\begin{aligned} P' \cap F &= \{x : Ax \leq \lfloor b \rfloor, ax = \beta\} \\ &= \{x : Ax \leq \lfloor b \rfloor, ax \leq \lfloor \beta \rfloor, ax \geq \lceil \beta \rceil\} \\ &= F'. \end{aligned}$$

Hier haben wir Satz 5.32 zweimal benutzt.

Beachte, dass F' entweder leer oder eine Seitenfläche von P' ist. Nun folgt die Aussage mittels Induktion über i : Für jedes i ist $F^{(i)}$ entweder leer oder eine Seitenfläche von $P^{(i)}$ und $F^{(i)} = P^{(i)} \cap F^{(i-1)} = P^{(i)} \cap (P^{(i-1)} \cap F) = P^{(i)} \cap F$. □

Lemma 5.34. Sei P ein Polyeder in \mathbb{R}^n und U eine unimodulare $n \times n$ -Matrix. Setze $f(P) := \{Ux : x \in P\}$. Dann ist $f(P)$ wieder ein Polyeder. Ferner gilt: Ist P ein rationales Polyeder, so folgt $(f(P))' = f(P')$ und $(f(P))_I = f(P_I)$.

Beweis: Da die Abbildung $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $x \mapsto Ux$, linear und bijektiv ist, ist die erste Aussage klar. Da auch die Restriktionen von f und f^{-1} auf \mathbb{Z}^n nach Proposition 5.10 bijektiv sind, haben wir

$$\begin{aligned}(f(P))_I &= \text{conv}(\{y \in \mathbb{Z}^n : y = Ux, x \in P\}) \\ &= \text{conv}(\{y \in \mathbb{R}^n : y = Ux, x \in P, x \in \mathbb{Z}^n\}) \\ &= \text{conv}(\{y \in \mathbb{R}^n : y = Ux, x \in P_I\}) \\ &= f(P_I).\end{aligned}$$

Sei $P = \{x : Ax \leq b\}$ mit $Ax \leq b$ TDI, A ganzzahlig und b rational (siehe Satz 5.18). Dann folgt nach Definition, dass $AU^{-1}x \leq b$ auch TDI ist. Unter zweimaliger Benutzung von Satz 5.32 folgt somit:

$$(f(P))' = \{x : AU^{-1}x \leq b\}' = \{x : AU^{-1}x \leq \lfloor b \rfloor\} = f(P').$$

□

Satz 5.35. (Schrijver [1980]) Für jedes rationale Polyeder P gibt es eine Zahl t mit $P^{(t)} = P_I$.

Beweis: Sei P ein rationales Polyeder in \mathbb{R}^n . Der Beweis erfolgt mittels Induktion über $n + \dim P$. Der Fall $P = \emptyset$ ist trivial und der Fall $\dim P = 0$ ist einfach.

Wir nehmen zunächst an, dass P nicht volldimensional ist. Dann gibt es eine rationale Hyperebene K mit $P \subseteq K$.

Enthält K keine ganzzahligen Vektoren, so ist $K = \{x : ax = \beta\}$ für einen ganzzahligen Vektor a und ein β , welches keine ganze Zahl ist (nach Lemma 5.12). Dann folgt aber $P' \subseteq \{x : ax \leq \lfloor \beta \rfloor, ax \geq \lceil \beta \rceil\} = \emptyset = P_I$.

Enthält K ganzzahlige Vektoren, etwa $K = \{x : ax = \beta\}$ mit a ganzzahlig und β eine ganze Zahl, so können wir $\beta = 0$ annehmen, da die Aussage des Satzes unter Translation bezüglich eines ganzzahligen Vektors invariant ist. Nach Lemma 5.11 gibt es eine unimodulare Matrix U mit $aU = \alpha e_1$. Da die Aussage des Satzes nach Lemma 5.34 auch unter der Transformation $x \mapsto U^{-1}x$ invariant ist, können wir annehmen, dass $a = \alpha e_1$. Damit verschwindet die erste Komponente eines jeden Vektors in P , also können wir die Dimension des Raumes um eins verringern und die Induktionsvoraussetzung anwenden (beachte, dass $(\{0\} \times Q)_I = \{0\} \times Q_I$ und $(\{0\} \times Q)^{(t)} = \{0\} \times Q^{(t)}$ für jedes Polyeder Q in \mathbb{R}^{n-1} und jedes $t \in \mathbb{N}$).

Nun sei $P = \{x : Ax \leq b\}$ volldimensional. O. B. d. A. können wir annehmen, dass A ganzzahlig ist. Nach Satz 5.1 gibt es eine ganzzahlige Matrix C und einen Vektor d mit $P_I = \{x : Cx \leq d\}$. Ist $P_I = \emptyset$, so setzen wir $C := A$ und $d := b - A'\mathbf{1}$, wobei A' aus A mittels Ersetzen einer jeden Komponente durch ihren Betrag hervorgeht. (Beachte, dass $\{x : Ax \leq b - A'\mathbf{1}\} = \emptyset$.)

Sei $cx \leq \delta$ eine Ungleichung aus $Cx \leq d$. Wir behaupten nun, dass $P^{(s)} \subseteq H := \{x : cx \leq \delta\}$ für irgendein $s \in \mathbb{N}$, womit der Satz dann offensichtlich folgt.

Beachte zunächst, dass es ein $\beta \geq \delta$ gibt mit $P \subseteq \{x : cx \leq \beta\}$: Ist $P_I = \emptyset$, so folgt dies aus der Wahl von C und d ; ist andererseits $P_I \neq \emptyset$, so folgt dies aus Proposition 5.2.

Angenommen, unsere Behauptung wäre falsch. Dann gibt es eine ganze Zahl γ mit $\delta < \gamma \leq \beta$ und der Eigenschaft: Es gibt ein $s_0 \in \mathbb{N}$ mit $P^{(s_0)} \subseteq \{x : cx \leq \gamma\}$, aber kein $s \in \mathbb{N}$ mit $P^{(s)} \subseteq \{x : cx \leq \gamma - 1\}$.

Beachte, dass $\max\{cx : x \in P^{(s)}\} = \gamma$ für alle $s \geq s_0$, denn gäbe es ein s mit $\max\{cx : x \in P^{(s)}\} < \gamma$, so wäre $P^{(s+1)} \subseteq \{x : cx \leq \gamma - 1\}$.

Sei $F := P^{(s_0)} \cap \{x : cx = \gamma\}$. Es ist F eine Seitenfläche von $P^{(s_0)}$ und $\dim F < n = \dim P$. Nach der Induktionsvoraussetzung gibt es eine Zahl s_1 mit

$$F^{(s_1)} = F_I \subseteq P_I \cap \{x : cx = \gamma\} = \emptyset.$$

Wenden wir Lemma 5.33 auf F und $P^{(s_0)}$ an, so haben wir

$$\emptyset = F^{(s_1)} = P^{(s_0+s_1)} \cap F = P^{(s_0+s_1)} \cap \{x : cx = \gamma\}.$$

Damit folgt der Widerspruch $\max\{cx : x \in P^{(s_0+s_1)}\} < \gamma$. \square

Aus diesem Satz folgt nunmehr

Satz 5.36. (Chvátal [1973]) Für jedes Polytop P gibt es eine Zahl t mit $P^{(t)} = P_I$.

Beweis: Da P beschränkt ist, gibt es ein rationales Polytop $Q \supseteq P$ mit $Q_I = P_I$ (nehme einen P enthaltenden Hyperwürfel und schneide ihn mit einem rationalen Halbraum, der P enthält, aber keinen im Hyperwürfel aber nicht in P liegenden ganzzahligen Punkt z ; siehe Aufgabe 20, Kapitel 3). Nach Satz 5.35 ist $Q^{(t)} = Q_I$ für ein t . Also ist $P_I \subseteq P^{(t)} \subseteq Q^{(t)} = Q_I = P_I$, woraus $P^{(t)} = P_I$ folgt. \square

Diese Zahl t heißt der Chvátal-Rang von P . Eine analoge Aussage für P weder beschränkt noch rational gibt es nicht: siehe Aufgaben 1 und 17.

Einen effizienteren Algorithmus, der die ganzzahlige Hülle eines zweidimensionalen Polyeders berechnet, hat Harvey [1999] gefunden. Eine Version der Schnittebenenmethode, die in polynomieller Zeit eine lineare Zielfunktion über einem durch ein Trennungs-Orakel gegebenen ganzzahligen Polytop approximiert, hat Boyd [1997] beschrieben. Cook, Kannan und Schrijver [1990] haben das Gomory-Chvátal-Verfahren für gemischt-ganzzahlige Optimierung verallgemeinert. Eisenbrand [1999] hat bewiesen, dass das folgende Entscheidungsproblem *coNP*-vollständig ist: Entscheide, ob ein gegebener rationaler Vektor für ein gegebenes rationales Polyeder P in P' liegt. Cornuéjols und Li [2016] haben gezeigt, dass es *NP*-vollständig ist zu entscheiden, ob P' leer ist, auch wenn P rational ist und $P_I = \emptyset$.

5.6 Lagrange-Relaxierung

Angenommen, wir haben ein ganzzahliges LP $\max\{cx : Ax \leq b, A'x \leq b', x \text{ ganzzahlig}\}$, welches wesentlich leichter zu lösen ist, wenn man die Nebenbedingungen $A'x \leq b'$ weglässt. Wir setzen $Q := \{x \in \mathbb{Z}^n : Ax \leq b\}$ und nehmen an, dass wir lineare Zielfunktionen über Q optimieren können (z. B. wenn $\text{conv}(Q) = \{x : Ax \leq b\}$). Die Lagrange-Relaxierung ist eine Methode zur Ausschaltung lästiger Nebenbedingungen (hier die Nebenbedingungen $A'x \leq b'$). Anstatt die Erfüllung dieser Nebenbedingungen explizit zu erzwingen, ändern wir die Zielfunktion so ab, dass Unzulässigkeit bestraft wird. Genauer: Anstatt dass wir

$$\max\{c^\top x : A'x \leq b', x \in Q\} \quad (5.7)$$

optimieren, betrachten wir für Vektoren $\lambda \geq 0$ das Programm

$$LR(\lambda) := \max\{c^\top x + \lambda^\top(b' - A'x) : x \in Q\}. \quad (5.8)$$

Für jeden Vektor $\lambda \geq 0$ ist $LR(\lambda)$ eine relativ leicht zu berechnende obere Schranke für (5.7). Das Programm (5.8) heißt die Lagrange-Relaxierung von (5.7) und die Komponenten von λ heißen die **Lagrange-Multiplikatoren**.

Lagrange-Relaxierung ist eine nützliche Methode in der nichtlinearen Optimierung; hier beschränken wir uns jedoch auf (ganzzahlige) LPs.

Natürlich ist man an einer möglichst guten oberen Schranke interessiert. Beachte, dass $\lambda \mapsto LR(\lambda)$ eine konvexe Funktion ist. Das folgende sogenannte Subgradienten-Verfahren kann zur Minimierung von $LR(\lambda)$ herangezogen werden:

Man beginnt mit einem beliebigen Vektor $\lambda^{(0)} \geq 0$. In Iteration i bestimme man bei gegebenem $\lambda^{(i)}$ einen Vektor $x^{(i)}$, der $c^\top x + (\lambda^{(i)})^\top(b' - A'x)$ über Q maximiert (d. h. berechne $LR(\lambda^{(i)})$). Beachte, dass $LR(\lambda) - LR(\lambda^{(i)}) \geq (\lambda - \lambda^{(i)})^\top(b' - A'x^{(i)})$ für alle λ , d. h. $b' - A'x^{(i)}$ ist ein Subgradient von LR für $\lambda^{(i)}$. Setze $\lambda^{(i+1)} := \max\{0, \lambda^{(i)} - t_i(b' - A'x^{(i)})\}$ für irgendein $t_i > 0$. Polyak [1967] hat gezeigt: Wenn $\lim_{i \rightarrow \infty} t_i = 0$ und $\sum_{i=0}^{\infty} t_i = \infty$, dann gilt $\lim_{i \rightarrow \infty} LR(\lambda^{(i)}) = \min\{LR(\lambda) : \lambda \geq 0\}$. Weitere Resultate bezüglich der Konvergenz des Subgradienten-Verfahrens findet man bei Goffin [1977] und Anstreicher und Wolsey [2009].

Das Problem, die beste dieser oberen Schranken zu bestimmen, d. h.

$$\min\{LR(\lambda) : \lambda \geq 0\},$$

wird gelegentlich das **Lagrange-Dual** von (5.7) genannt. Wir werden zeigen, dass das Minimum immer angenommen wird, sofern $\{x : Ax \leq b, A'x \leq b'\} \neq \emptyset$. Die zweite Frage, der wir uns zuwenden werden, ist, wie gut diese obere Schranke tatsächlich ist. Natürlich hängt dies von der Struktur des ursprünglichen Optimierungsproblems ab. In Abschnitt 21.5 werden wir eine Anwendung kennenlernen, nämlich auf das TRAVELING-SALESMAN-PROBLEM (TSP), bei dem die Lagrange-Relaxierung sehr effektiv ist. Der folgende Satz hilft uns bei der Einschätzung der Qualität dieser oberen Schranke:

Satz 5.37. (Geoffrion [1974]) Sei $c \in \mathbb{R}^n$, $A' \in \mathbb{R}^{m \times n}$ und $b' \in \mathbb{R}^m$. Sei ferner $Q \subseteq \mathbb{R}^n$, so dass $\text{conv}(Q)$ ein Polyeder ist. Angenommen, $\max\{c^\top x : A'x \leq b', x \in \text{conv}(Q)\}$ hat eine optimale Lösung. Sei $LR(\lambda) := \max\{c^\top x + \lambda^\top(b' - A'x) : x \in Q\}$. Dann wird $\inf\{LR(\lambda) : \lambda \geq 0\}$ (der optimale Wert des Lagrange-Duals von $\max\{c^\top x : A'x \leq b', x \in Q\}$) für ein λ angenommen und dieses Minimum ist gleich $\max\{c^\top x : A'x \leq b', x \in \text{conv}(Q)\}$.

Beweis: Sei $\text{conv}(Q) = \{x : Ax \leq b\}$. Mittels Umformulierung und zweifacher Verwendung des LP-Dualitätssatzes (Satz 3.20) bekommen wir

$$\begin{aligned} & \max\{c^\top x : x \in \text{conv}(Q), A'x \leq b'\} \\ &= \max\{c^\top x : Ax \leq b, A'x \leq b'\} \\ &= \min\{\lambda^\top b' + y^\top b : y^\top A + \lambda^\top A' = c^\top, y \geq 0, \lambda \geq 0\} \\ &= \min\{\lambda^\top b' + \min\{y^\top b : y^\top A = c^\top - \lambda^\top A', y \geq 0\} : \lambda \geq 0\} \\ &= \min\{\lambda^\top b' + \max\{(c^\top - \lambda^\top A')x : Ax \leq b\} : \lambda \geq 0\} \\ &= \min\{\max\{c^\top x + \lambda^\top(b' - A'x) : x \in \text{conv}(Q)\} : \lambda \geq 0\} \\ &= \min\{\max\{c^\top x + \lambda^\top(b' - A'x) : x \in Q\} : \lambda \geq 0\} \\ &= \min\{LR(\lambda) : \lambda \geq 0\}. \end{aligned}$$

Die dritte Zeile beinhaltet ein LP und zeigt, dass es ein λ gibt, für welches das Minimum angenommen wird. \square

Als Spezialfall folgt: Haben wir ein ganzzahliges LP $\max\{cx : A'x \leq b', Ax \leq b, x \text{ ganzzahlig}\}$ mit $\{x : Ax \leq b\}$ ganzzahlig, so ergibt das Lagrange-Dual (indem man $A'x \leq b'$ wie oben relaxiert) dieselbe obere Schranke wie die LP-Relaxierung $\max\{cx : A'x \leq b', Ax \leq b\}$. Ist andererseits $\{x : Ax \leq b\}$ nicht ganzzahlig, so ist die obere Schranke meist niedriger, kann aber schwer zu berechnen sein (siehe Aufgabe 21 für ein Beispiel).

Lagrange-Relaxierung kann auch zur Approximation von LPs herangezogen werden. Betrachte z. B. das JOB-ZUORDNUNGSPROBLEM (siehe (1.1) in Abschnitt 1.3). Eine äquivalente Form dieses Problems ist

$$\min \left\{ T : \sum_{j \in S_i} x_{ij} \geq t_i \ (i = 1, \dots, n), (x, T) \in P \right\}, \quad (5.9)$$

wobei P das Polytop

$$\begin{aligned} & \left\{ (x, T) : 0 \leq x_{ij} \leq t_i \ (i = 1, \dots, n, j \in S_i), \right. \\ & \quad \sum_{i:j \in S_i} x_{ij} \leq T \ (j = 1, \dots, m), \\ & \quad \left. T \leq \sum_{i=1}^n t_i \right\}. \end{aligned}$$

ist. Nun wenden wir Lagrange-Relaxierung an und betrachten dazu

$$LR(\lambda) := \min \left\{ T + \sum_{i=1}^n \lambda_i \left(t_i - \sum_{j \in S_i} x_{ij} \right) : (x, T) \in P \right\}. \quad (5.10)$$

Wegen der besonderen Struktur dieses LP kann es für beliebige λ mittels eines einfachen kombinatorischen Algorithmus gelöst werden (siehe Aufgabe 23). Bezeichnen wir mit Q die Menge der Ecken von P (siehe Korollar 3.32), dann können wir unter Anwendung von Satz 5.37 folgern, dass der optimale Wert des Lagrange-Duals $\max\{LR(\lambda) : \lambda \geq 0\}$ gleich dem Optimum von (5.9) ist.

Aufgaben

1. Sei $P := \{(x, y) \in \mathbb{R}^2 : y \leq \sqrt{2}x\}$. Man beweise, dass P_I kein Polyeder ist. Man konstruiere ein Beispiel eines Polyeders P , für welches sogar der Abschluss von P_I kein Polyeder ist.
2. Sei $P = \{x \in \mathbb{R}^{k+l} : Ax \leq b\}$ ein rationales Polyeder. Man zeige, dass $\text{conv}(P \cap (\mathbb{Z}^k \times \mathbb{R}^l))$ ein Polyeder ist.

Hinweis: Man verallgemeinere den Beweis von Satz 5.1.

Bemerkung: Dies bildet die Basis der gemischt-ganzzahligen Optimierung; siehe Schrijver [1986].

- * 3. Man beweise die folgende ganzzahlige Version des Satzes von Carathéodory (siehe Aufgabe 15, Kapitel 3): Für jeden spitzen polyedrischen Kegel C , jede Hilbertbasis $\{a_1, \dots, a_t\}$ von C und jeden ganzzahligen Punkt $x \in C$ gibt es $2n - 1$ Vektoren unter den a_1, \dots, a_t , so dass x eine nichtnegative ganzzahlige Linearkombination dieser ist.

Hinweis: Man betrachte eine optimale Basislösung des LP $\max\{y\mathbf{1} : yA = x, y \geq 0\}$ (wobei die Zeilen von A die Vektoren a_1, \dots, a_t der Hilbertbasis sind) und runde die Komponenten ab.

Bemerkung: Die Zahl $2n - 1$ wurde von Sebő [1990] auf $2n - 2$ verbessert. Sie kann jedoch nicht auf eine Zahl kleiner als $\lfloor \frac{7}{6}n \rfloor$ verbessert werden (Bruns et al. [1999]).

(Cook, Fonlupt und Schrijver [1986])

4. Sei $C = \{x : Ax \geq 0\}$ ein rationaler polyedrischer Kegel und b ein Vektor mit $bx > 0$ für alle $x \in C \setminus \{0\}$. Man zeige, dass es eine eindeutig bestimmte inklusionsminimale C erzeugende ganzzahlige Hilbertbasis gibt.
(Schrijver [1981])

5. Sei A eine ganzzahlige $m \times n$ -Matrix, b und c Vektoren und y eine optimale Lösung von $\max\{cx : Ax \leq b, x \text{ ganzzahlig}\}$. Man beweise, dass es eine optimale Lösung z von $\max\{cx : Ax \leq b\}$ mit $\|y - z\|_\infty \leq n\Xi(A)$ gibt.
(Cook et al. [1986])

6. Man beweise, dass jede unimodulare Matrix aus einer Einheitsmatrix mittels einer unimodularen Transformation hervorgeht.

Hinweis: Man betrachte den Beweis von Lemma 5.11.

- * 7. Man beweise, dass es einen polynomiellen Algorithmus gibt, der für eine gegebene ganzzahlige Matrix A und einen ganzzahligen Vektor b einen ganzzahligen Vektor x mit $Ax = b$ findet oder entscheidet, dass es keinen solchen gibt.

Hinweis: Man betrachte die Beweise von Lemma 5.11 und Lemma 5.12.

- 8. Man betrachte die beiden Systeme

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Offensichtlich definieren sie beide dasselbe Polyeder. Man beweise, dass das erste System TDI ist, das zweite jedoch nicht.

- 9. Sei $a \neq 0$ ein ganzzahliges Vektor und β eine rationale Zahl. Man beweise, dass die Ungleichung $ax \leq \beta$ genau dann TDI ist, wenn die Komponenten von a teilerfremd sind.
- 10. Sei $Ax \leq b$ TDI, $k \in \mathbb{N}$ und $\alpha > 0$ rational. Man zeige, dass $\frac{1}{k}Ax \leq \alpha b$ wieder TDI ist. Ferner zeige man, dass $\alpha Ax \leq \alpha b$ im Allgemeinen nicht TDI ist.
- 11. Man beweise den Satz von König (Satz 10.2) mittels Satz 5.26 (siehe Aufgabe 2, Kapitel 11):
Die maximale Kardinalität eines Matchings in einem bipartiten Graphen ist gleich der minimalen Kardinalität einer Knotenüberdeckung.
- 12. Man zeige, dass $A = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ nicht vollständig-unimodular ist, dass aber $\{x : Ax = b\}$ für jeden ganzzahligen Vektor b ganzzahlig ist.
(Nemhauser und Wolsey [1988])
- 13. Sei G der Digraph $(\{1, 2, 3, 4\}, \{(1, 3), (2, 4), (2, 1), (4, 1), (4, 3)\})$ und $\mathcal{F} := \{\{1, 2, 4\}, \{1, 2\}, \{2\}, \{2, 3, 4\}, \{4\}\}$. Man beweise, dass $(V(G), \mathcal{F})$ kreuzungsfrei ist, dass aber die Einweg-Schnitt-Inzidenz-Matrix von \mathcal{F} nicht vollständig-unimodular ist.
- * 14. Seien G und T Digraphen, so dass $V(G) = V(T)$ und der T zugrunde liegende ungerichtete Graph ein Baum ist. Für $v, w \in V(G)$ sei $P(v, w)$ der eindeutig bestimmte ungerichtete Weg von v nach w in T . Sei $M = (m_{f,e})_{f \in E(T), e \in E(G)}$ die folgendermaßen definierte Matrix:

$$m_{(x,y),(v,w)} := \begin{cases} 1 & \text{für } (x, y) \in E(P(v, w)) \text{ und } (x, y) \in E(P(v, y)) \\ -1 & \text{für } (x, y) \in E(P(v, w)) \text{ und } (x, y) \in E(P(v, x)) \\ 0 & \text{für } (x, y) \notin E(P(v, w)). \end{cases}$$

Solche Matrizen heißen Netzwerkmatrizen. Man zeige, dass die Netzwerkmatrizen genau die Zweiwege-Schnitt-Inzidenz-Matrizen von kreuzungsfreien Mengensystemen sind.

15. Eine Intervallmatrix ist eine 0-1-Matrix mit der Eigenschaft, dass die Einserkomponenten in jeder Zeile einen fortlaufenden Block bilden. Man beweise, dass Intervallmatrizen vollständig-unimodular sind.
- Bemerkung:* Hochbaum und Levin [2006] haben gezeigt, wie man Optimierungsprobleme mit solchen Matrizen sehr effizient lösen kann.
16. Man betrachte das folgende Intervall-Packungsproblem: Gegeben sei eine Liste von Intervallen $[a_i, b_i]$, $i = 1, \dots, n$, mit den Gewichten c_1, \dots, c_n und eine Zahl $k \in \mathbb{N}$. Dann bestimme man eine maximal gewichtete Teilmenge der Liste der Intervalle, so dass kein Punkt in mehr als k von ihnen enthalten ist.
- Man formuliere dieses Problem als LP ohne Ganzzahligkeitsnebenbedingungen.
 - Man betrachte den Fall $k = 1$. Welche kombinatorische Interpretation hat das duale LP? Man zeige, wie man das duale LP mit einem einfachen kombinatorischen Algorithmus lösen kann.
 - Man beschreibe unter Verwendung von (b) einen Algorithmus mit $O(n \log n)$ -Laufzeit zur Lösung des Intervall-Packungsproblems für den Fall $k = 1$.
 - Man beschreibe einen einfachen $O(n \log n)$ -Algorithmus für allgemeine k und Einheitsgewichte.

Bemerkung: Siehe auch Aufgabe 11, Kapitel 9.

17. Sei $P := \{(x, y) \in \mathbb{R}^2 : y = \sqrt{2}x, x \geq 0\}$ und $Q := \{(x, y) \in \mathbb{R}^2 : y = \sqrt{2}x\}$. Man beweise, dass $P^{(t)} = P \neq P_I$ für alle $t \in \mathbb{N}$ und $Q' = \mathbb{R}^2$.
18. Sei P die konvexe Hülle der drei Punkte $(0, 0)$, $(0, 1)$ und $(k, \frac{1}{2})$ in \mathbb{R}^2 mit $k \in \mathbb{N}$. Man zeige, dass $P^{(2k-1)} \neq P_I$, aber $P^{(2k)} = P_I$.
- * 19. Sei $P \subseteq [0, 1]^n$ ein Polytop im Einheitshyperwürfel mit $P_I = \emptyset$. Man beweise, dass $P^{(n)} = \emptyset$.
- Bemerkung:* Eisenbrand und Schulz [2003] haben bewiesen, dass $P^{\lceil n^2(1+\log n) \rceil} = P_I$ für jedes Polytop $P \subseteq [0, 1]^n$. Siehe auch Pokutta und Schulz [2010] sowie Rothvoß und Sanit   [2017].
20. In dieser Aufgabe wende man Lagrange-Relaxierung auf lineare Gleichungssysteme an. Sei Q eine endliche Menge von Vektoren in \mathbb{R}^n , $c \in \mathbb{R}^n$, $A' \in \mathbb{R}^{m \times n}$ und $b' \in \mathbb{R}^m$. Man beweise, dass

$$\begin{aligned} & \min \left\{ \max \{c^\top x + \lambda^\top (b' - A'x) : x \in Q\} : \lambda \in \mathbb{R}^m \right\} \\ &= \max \{c^\top y : y \in \text{conv}(Q), A'y = b'\}. \end{aligned}$$

21. Man betrachte das folgende Standortproblem: Gegeben sei eine Menge von n Kunden, jeweils mit der Nachfrage d_j , $j = 1, \dots, n$, und m mögliche Standorte, von denen jeder einzeln bereitgestellt werden kann. Für jeden Standort $i = 1, \dots, m$ haben wir Bereitstellungskosten f_i , eine Kapazität u_i und eine Entfernung c_{ij} zu jedem der Kunden $j = 1, \dots, n$. Man möchte entscheiden, welche Standorte bereitgestellt werden sollen und möchte jeden Kunden einem der Standorte zuordnen. Die Gesamtnachfrage der einem Standort zugeteilten Kunden darf nicht die Kapazität dieses Standorts übersteigen. Ziel ist es, die Summe der Gesamtbereitstellungskosten und der Entferungen aller Kunden zu

ihren Standorten zu minimieren. Dieses Problem kann mittels GANZZAHLIGER OPTIMIERUNG folgendermaßen formuliert werden:

$$\min \left\{ \sum_{i,j} c_{ij} x_{ij} + \sum_i f_i y_i : \sum_j d_j x_{ij} \leq u_i y_i, \sum_i x_{ij} = 1, x_{ij}, y_i \in \{0, 1\} \right\}.$$

Man wende Lagrange-Relaxierung auf zwei verschiedene Weisen an: Man relaxiere $\sum_j d_j x_{ij} \leq u_i y_i$ für alle i , oder $\sum_i x_{ij} = 1$ für alle j . Welches Lagrange-Dual liefert eine schärfere Schranke?

Bemerkung: Beide Lagrange-Relaxierungen sind handhabbar (siehe Aufgabe 11, Kapitel 17).

- * 22. Man betrachte das UNBESCHRÄNKTE STANDORTPROBLEM: Gegeben seien Zahlen n, m, f_i und c_{ij} ($i = 1, \dots, m, j = 1, \dots, n$). Das Problem kann folgendermaßen formuliert werden:

$$\min \left\{ \sum_{i,j} c_{ij} x_{ij} + \sum_i f_i y_i : \sum_i x_{ij} = 1, x_{ij} \leq y_i, x_{ij}, y_i \in \{0, 1\} \right\}.$$

Für $S \subseteq \{1, \dots, n\}$ bezeichnen wir mit $c(S)$ die Kosten der Bereitstellung der Standorte für die Kunden in S , d. h.

$$\min \left\{ \sum_{i,j} c_{ij} x_{ij} + \sum_i f_i y_i : \sum_i x_{ij} = 1 \text{ für } j \in S, x_{ij} \leq y_i, x_{ij}, y_i \in \{0, 1\} \right\}.$$

Das Kostenzuteilungsproblem fragt, ob man die Gesamtkosten $c(\{1, \dots, n\})$ so unter den Kunden aufteilen kann, dass keine Teilmenge S mehr als $c(S)$ zahlt. Mit anderen Worten: Gibt es Zahlen p_1, \dots, p_n , so dass $\sum_{j=1}^n p_j = c(\{1, \dots, n\})$ und $\sum_{j \in S} p_j \leq c(S)$ für alle $S \subseteq \{1, \dots, n\}$? Man zeige, dass dies genau dann der Fall ist, wenn $c(\{1, \dots, n\})$ gleich

$$\min \left\{ \sum_{i,j} c_{ij} x_{ij} + \sum_i f_i y_i : \sum_i x_{ij} = 1, x_{ij} \leq y_i, x_{ij}, y_i \geq 0 \right\}$$

ist, d. h. wenn die Ganzzahligkeitsbedingungen weggelassen werden können.

Hinweis: Man wende Lagrange-Relaxierung auf obiges LP an. Dann zerlege man das resultierende Minimierungsproblem für jeden Satz von Lagrange-Multiplikatoren in Minimierungsprobleme über polyedrischen Kegeln. Welche Vektoren erzeugen diese Kegel?

(Goemans und Skutella [2004])

- 23. Man beschreibe einen kombinatorischen Algorithmus (ohne LINEARE OPTIMIERUNG zu verwenden), um (5.10) für beliebige (aber feste) Lagrange-Multiplikatoren λ zu lösen. Welche Laufzeit hat der Algorithmus?

Literatur

Allgemeine Literatur:

- Bertsimas, D., und Weismantel, R. [2005]: Optimization Over Integers. Dynamic Ideas, Belmont 2005
- Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., und Schrijver, A. [1998]: Combinatorial Optimization. Wiley, New York 1998, Kapitel 6
- Jünger, M., Liebling, T., Naddef, D., Nemhauser, G., Pulleyblank, W., Reinelt, G., Rinaldi, G. und Wolsey, L., Hrsg. [2010]: 50 Years of Integer Programming 1958–2008. Springer, Berlin 2010
- Nemhauser, G.L., und Wolsey, L.A. [1988]: Integer and Combinatorial Optimization. Wiley, New York 1988
- Schrijver, A. [1986]: Theory of Linear and Integer Programming. Wiley, Chichester 1986
- Wolsey, L.A. [1998]: Integer Programming. Wiley, New York 1998

Zitierte Literatur:

- Anstreicher, K.M., und Wolsey, L.A. [2009]: Two “well-known” properties of subgradient optimization. Mathematical Programming B 120 (2009), 213–220
- Boyd, E.A. [1997]: A fully polynomial epsilon approximation cutting plane algorithm for solving combinatorial linear programs containing a sufficiently large ball. Operations Research Letters 20 (1997), 59–63
- Brunn, W., Gubeladze, J., Henk, M., Martin, A., und Weismantel, R. [1999]: A counterexample to an integral analogue of Carathéodory’s theorem. Journal für die Reine und Angewandte Mathematik 510 (1999), 179–185
- Carr, R., und Vempala, S. [2004]: On the Held-Karp relaxation for the asymmetric and symmetric traveling salesman problems. Mathematical Programming A 100 (2004), 569–587
- Chvátal, V. [1973]: Edmonds’ polytopes and a hierarchy of combinatorial problems. Discrete Mathematics 4 (1973), 305–337
- Cook, W.J. [1983]: Operations that preserve total dual integrality. Operations Research Letters 2 (1983), 31–35
- Cook, W.J., Fonlupt, J., und Schrijver, A. [1986]: An integer analogue of Carathéodory’s theorem. Journal of Combinatorial Theory B 40 (1986), 63–70
- Cook, W.J., Gerards, A., Schrijver, A., und Tardos, É. [1986]: Sensitivity theorems in integer linear programming. Mathematical Programming 34 (1986), 251–264
- Cook, W.J., Kannan, R., und Schrijver, A. [1990]: Chvátal closures for mixed integer programming problems. Mathematical Programming 47 (1990), 155–174
- Cornuéjols, G., und Li, Y. [2016]: Deciding emptiness of the Gomory-Chvátal closure is NP-complete, even for a rational polyhedron containing no integer point. In: Integer Programming and Combinatorial Optimization; Proceedings of the 18th International IPCO Conference; LNCS 9682 (Q. Louveaux, M. Skutella, Hrsg.), Springer, Switzerland 2016, pp. 387–397
- Dadush, D., Dey, S.S., und Vielma, J.P. [2014]: On the Chvátal-Gomory closure of a compact convex set. Mathematical Programming A 145 (2014), 327–348
- Dantzig, G., Fulkerson, R., und Johnson, S. [1954]: Solution of a large-scale traveling-salesman problem. Operations Research 2 (1954), 393–410

- Edmonds, J., und Giles, R. [1977]: A min-max relation for submodular functions on graphs. In: Studies in Integer Programming; Annals of Discrete Mathematics 1 (P.L. Hammer, E.L. Johnson, B.H. Korte, G.L. Nemhauser, Hrsg.), North-Holland, Amsterdam 1977, 185–204
- Eisenbrand, F. [1999]: On the membership problem for the elementary closure of a polyhedron. *Combinatorica* 19 (1999), 297–300
- Eisenbrand, F., und Schulz, A.S. [2003]: Bounds on the Chvátal rank of polytopes in the 0/1-cube. *Combinatorica* 23 (2003), 245–261
- Fulkerson, D.R. [1971]: Blocking and anti-blocking pairs of polyhedra. *Mathematical Programming* 1 (1971), 168–194
- Geoffrion, A.M. [1974]: Lagrangean relaxation for integer programming. *Mathematical Programming Study* 2 (1974), 82–114
- Giles, F.R., und Pulleyblank, W.R. [1979]: Total dual integrality and integer polyhedra. *Linear Algebra and Its Applications* 25 (1979), 191–196
- Ghouila-Houri, A. [1962]: Caractérisation des matrices totalement unimodulaires. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris)* 254 (1962), 1192–1194
- Goemans, M.X. [1995]: Worst-case comparison of valid inequalities for the TSP. *Mathematical Programming* 69 (1995), 335–349
- Goemans, M.X., und Skutella, M. [2004]: Cooperative facility location games. *Journal of Algorithms* 50 (2004), 194–214
- Goffin, J.L. [1977]: On convergence rates of subgradient optimization methods. *Mathematical Programming* 13 (1977), 329–347
- Gomory, R.E. [1958]: Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society* 64 (1958), 275–278
- Gomory, R.E. [1963]: An algorithm for integer solutions of linear programs. In: Recent Advances in Mathematical Programming (R.L. Graves, P. Wolfe, Hrsg.), McGraw-Hill, New York, 1963, pp. 269–302
- Graver, J.E. [1975]: On the foundations of linear and integer programming I. *Mathematical Programming* 9 (1975), 207–226
- Harvey, W. [1999]: Computing two-dimensional integer hulls. *SIAM Journal on Computing* 28 (1999), 2285–2299
- Hochbaum, D.S., und Levin, A. [2006]: Optimizing over consecutive 1's and circular 1's constraints. *SIAM Journal on Optimization* 17 (2006), 311–330
- Hoffman, A.J. [1974]: A generalization of max flow-min cut. *Mathematical Programming* 6 (1974), 352–359
- Hoffman, A.J., und Kruskal, J.B. [1956]: Integral boundary points of convex polyhedra. In: Linear Inequalities and Related Systems; *Annals of Mathematical Study* 38 (H.W. Kuhn, A.W. Tucker, Hrsg.), Princeton University Press, Princeton 1956, 223–246
- Lasserre, J.B. [2004]: The integer hull of a convex rational polytope. *Discrete & Computational Geometry* 32 (2004), 129–139
- Meyer, R.R. [1974]: On the existence of optimal solutions to integer and mixed-integer programming problems. *Mathematical Programming* 7 (1974), 223–235
- Pokutta, S., und Schulz, A.S. [2010]: On the rank of cutting-plane proof systems. In: Integer Programming and Combinatorial Optimization; Proceedings of the 14th International IPCO Conference; LNCS 6080 (F. Eisenbrand, F.B. Shepherd, Hrsg.), Springer, Berlin 2010, pp. 450–463
- Polyak, B.T. [1967]: A general method for solving extremal problems. *Doklady Akademii Nauk SSSR* 174 (1967), 33–36 [auf Russisch]. Englische Übersetzung: Soviet Mathematics Doklady 8 (1967), 593–597

- Rothvoß, T., und Sanità, L. [2017]: 0/1 polytopes with quadratic Chvátal rank. *Operations Research* 65 (2017), 212–220
- Schrijver, A. [1980]: On cutting planes. In: Combinatorics 79; Part II; Annals of Discrete Mathematics 9 (M. Deza, I.G. Rosenberg, Hrsg.), North-Holland, Amsterdam 1980, pp. 291–296
- Schrijver, A. [1981]: On total dual integrality. *Linear Algebra and its Applications* 38 (1981), 27–32
- Schrijver, A. [1983]: Packing and covering of crossing families of cuts. *Journal of Combinatorial Theory B* 35 (1983), 104–128
- Sebő, A. [1990]: Hilbert bases, Carathéodory's theorem and combinatorial optimization. In: Integer Programming and Combinatorial Optimization (R. Kannan und W.R. Pulleyblank, Hrsg.), University of Waterloo Press, 1990
- Seymour, P.D. [1980]: Decomposition of regular matroids. *Journal of Combinatorial Theory B* 28 (1980), 305–359
- Veinott, A.F., Jr., und Dantzig, G.B. [1968]. Integral extreme points. *SIAM Review* 10 (1968), 371–372
- Wolsey, L.A. [1981]: The b -hull of an integer program. *Discrete Applied Mathematics* 3 (1981), 193–201



6 Aufspannende Bäume und Arboreszenzen

Gegeben sei eine Telefongesellschaft, die eine Teilmenge einer vorhandenen Menge von Kabelverbindungen mieten möchte, wobei jede Kabelverbindung zwei Städte verbindet. Die zu mietenden Kabelverbindungen sollen alle Städte auf die billigste Weise miteinander verbinden. Dieses Problem stellt man natürlich durch einen Graphen dar: Die Knoten sind die Städte und die Kanten die Kabelverbindungen. Nach Satz 2.4 sind die minimalen zusammenhängenden aufspannenden Teilgraphen eines gegebenen Graphen seine aufspannenden Bäume. Also suchen wir hier einen aufspannenden Baum minimalen Gewichtes, wobei das Gewicht eines Teilgraphen T eines Graphen G mit den Gewichten $c : E(G) \rightarrow \mathbb{R}$ gleich $c(E(T)) = \sum_{e \in E(T)} c(e)$ ist. Der Wert $c(e)$ wird auch als die Kosten der Kante e bezeichnet.

Dies ist ein zwar einfaches aber sehr wichtiges kombinatorisches Optimierungsproblem. Es zählt auch zu den ältesten kombinatorischen Optimierungsproblemen, d. h. zu denen mit der längsten Geschichte. Der erste Algorithmus stammt von Borůvka [1926a, 1926b], siehe Nešetřil, Milková und Nešetřilová [2001].

Bei dem BOHRPUNKTPROBLEM suchen wir einen kürzesten alle Knoten eines vollständigen Graphen durchlaufenden Weg, hier suchen wir einen kürzesten aufspannenden Baum. Obwohl die Anzahl der aufspannenden Bäume noch viel größer als diejenige der Wege ist (K_n enthält $\frac{n!}{2}$ hamiltonsche Wege, aber n^{n-2} verschiedene aufspannende Bäume; siehe Satz 6.2), kann man dieses Problem viel leichter lösen. In der Tat genügt eine simple Greedy-Strategie, wie wir in Abschnitt 6.1 sehen werden.

Arboreszenzen kann man als die gerichtete Version von Bäumen betrachten; nach Satz 2.5 sind sie die minimalen aufspannenden Teilgraphen eines Digraphen mit der Eigenschaft, dass alle Knoten von einer Wurzel aus erreichbar sind. Die gerichtete Version des MINIMUM-SPANNING-TREE-PROBLEMS, das MINIMUM-WEIGHT-ARBORESCENCE-PROBLEM, macht einige Schwierigkeiten, da man mit einer Greedy-Strategie nicht zum Ziel kommt. In Abschnitt 6.2 beschreiben wir einen Lösungsweg.

Da es sehr effiziente kombinatorische Algorithmen für diese Probleme gibt, ist es nicht empfehlenswert, sie mit LINEARER OPTIMIERUNG zu lösen. Trotzdem ist es durchaus interessant, dass sich die zugehörigen Polytope (die konvexe Hülle der Inzidenzvektoren der aufspannenden Bäume bzw. der Arboreszenzen, siehe Korollar 3.33) auf schöne Weise beschreiben lassen, wie wir in Abschnitt 6.3

zeigen werden. In Abschnitt 6.4 werden wir einige klassische Resultate bezüglich des Packens von aufspannenden Bäumen und Arboreszenzen beweisen.

6.1 Aufspannende Bäume mit minimalem Gewicht

In diesem Abschnitt betrachten wir die folgenden zwei Probleme:

MAXIMUM-WEIGHT-FOREST-PROBLEM

Instanz: Ein ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Bestimme einen Wald in G mit maximalem Gewicht.

MINIMUM-SPANNING-TREE-PROBLEM

Instanz: Ein ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Bestimme einen aufspannenden Baum in G mit minimalem Gewicht, oder entscheide, dass G nicht zusammenhängend ist.

Wir behaupten, dass diese beiden Probleme äquivalent sind. Genauer: Wir sagen, dass ein Problem \mathcal{P} **linear reduzierbar** auf ein Problem \mathcal{Q} ist, wenn es zwei in linearer Zeit berechenbare Funktionen f und g gibt, so dass f eine Instanz x von \mathcal{P} in eine Instanz $f(x)$ von \mathcal{Q} transformiert und g eine Lösung von $f(x)$ in eine Lösung von x transformiert. Ist \mathcal{Q} linear reduzierbar von \mathcal{P} und \mathcal{P} linear reduzierbar von \mathcal{Q} , so heißen die Probleme \mathcal{P} und \mathcal{Q} **äquivalent**.

Proposition 6.1. *Das MAXIMUM-WEIGHT-FOREST-PROBLEM und das MINIMUM-SPANNING-TREE-PROBLEM sind äquivalent.*

Beweis: Sei (G, c) eine Instanz des MAXIMUM-WEIGHT-FOREST-PROBLEMS. Entferne alle negativ gewichteten Kanten und setze $c'(e) := -c(e)$ für alle $e \in E(G)$. Füge eine inklusionsminimale den Graphen zusammenhängend machende Menge F von Kanten mit beliebigen Gewichten hinzu; sei G' der resultierende Graph. Dann ist die Instanz (G', c') des MINIMUM-SPANNING-TREE-PROBLEMS in folgendem Sinne äquivalent: Entfernt man die in F liegenden Kanten eines aufspannenden Baumes minimalen Gewichtes in (G', c') , so resultiert ein Wald maximalen Gewichtes in (G, c) .

Zum Beweis der Umkehrung sei (G, c) eine Instanz des MINIMUM-SPANNING-TREE-PROBLEMS. Setze $c'(e) := K - c(e)$ für alle $e \in E(G)$, wobei $K = 1 + \max_{e \in E(G)} c(e)$. Dann ist die Instanz (G, c') des MAXIMUM-WEIGHT-FOREST-PROBLEMS äquivalent, da alle aufspannenden Bäume nach Satz 2.4 die gleiche Anzahl von Kanten haben. \square

In Kapitel 15 werden wir es nochmals mit Reduktionen von Problemen zu tun haben. In diesem Abschnitt werden wir uns nur noch mit dem MINIMUM-SPANNING-TREE-PROBLEM befassen.

Zunächst werden wir die Anzahl der zulässigen Lösungen bestimmen. Das folgende Resultat ist als Satz von Cayley bekannt:

Satz 6.2. (Sylvester [1857], Cayley [1889]) *Die Anzahl der aufspannenden Bäume in K_n mit $n \in \mathbb{N}$ ist n^{n-2} .*

Beweis: Sei t_n die Anzahl der aufspannenden Bäume in K_n . Sei ferner

$$\mathcal{B}_{n,k} := \{(B, f) : B \text{ Branching}, V(B) = \{1, \dots, n\}, |E(B)| = k, f : E(B) \rightarrow \{1, \dots, k\} \text{ bijektiv}\}.$$

Da jeder aufspannende Baum n Orientierungen als Arboreszenz hat und seine Kantenmenge auf $(n-1)!$ verschiedene Weisen geordnet werden kann, folgt $|\mathcal{B}_{n,n-1}| = (n-1)! \cdot n \cdot t_n$. Ferner gilt $|\mathcal{B}_{n,0}| = 1$. Wir werden nun beweisen, dass $|\mathcal{B}_{n,i+1}| = n(n-i-1)|\mathcal{B}_{n,i}|$, woraus $|\mathcal{B}_{n,n-1}| = n^{n-1}(n-1)!|\mathcal{B}_{n,0}|$ folgt. Somit ist $t_n = n^{n-2}$.

Für jedes Element $(B, f) \in \mathcal{B}_{n,i+1}$ definieren wir $g(B, f) \in \mathcal{B}_{n,i}$ durch Entfernen der Kante $f^{-1}(i+1)$. Nun ist jedes $(B', f) \in \mathcal{B}_{n,i}$ das Bild von genau $n(n-i-1)$ Elementen von $\mathcal{B}_{n,i+1}$: Wir fügen eine Kante $e = (v, w)$ hinzu und setzen $f(e) := i+1$; es gibt n verschiedene Wahlen für v (alle Knoten), und es gibt $n-i-1$ verschiedene Wahlen für w (die Wurzeln der v nicht enthaltenden Zusammenhangskomponenten von B'). \square

Als Nächstes leiten wir Optimalitätsbedingungen ab.

Satz 6.3. *Sei (G, c) eine Instanz des MINIMUM-SPANNING-TREE-PROBLEMS und T ein aufspannender Baum in G . Dann sind die folgenden vier Aussagen äquivalent:*

- (a) T ist optimal.
- (b) Für jedes $e = \{x, y\} \in E(G) \setminus E(T)$ gilt: Keine Kante des x - y -Weges in T hat höheres Gewicht als e .
- (c) Für jedes $e \in E(T)$ gilt: Ist C eine der beiden Zusammenhangskomponenten von $T - e$, so ist e eine Kante von $\delta(V(C))$ mit minimalem Gewicht.
- (d) Es kann $E(T) = \{e_1, \dots, e_{n-1}\}$ so geordnet werden, dass es für jedes $i \in \{1, \dots, n-1\}$ eine Menge $X \subseteq V(G)$ gibt, für die e_i eine Kante von $\delta(X)$ mit minimalem Gewicht ist und $e_j \notin \delta(X)$ für alle $j \in \{1, \dots, i-1\}$.

Beweis: (a) \Rightarrow (b): Angenommen, (b) wäre nicht erfüllt: Sei $e = \{x, y\} \in E(G) \setminus E(T)$ und f eine Kante des x - y -Weges in T mit $c(f) > c(e)$. Dann ist $(T - f) + e$ ein aufspannender Baum mit geringerem Gewicht.

(b) \Rightarrow (c): Angenommen, (c) wäre nicht erfüllt: Dann gibt es ein $e \in E(T)$, eine Zusammenhangskomponente C von $T - e$ und eine Kante $f = \{x, y\} \in \delta(V(C))$ mit $c(f) < c(e)$. Beachte, dass der x - y -Weg in T eine Kante aus $\delta(V(C))$ enthalten muss, dass aber e die einzige solche Kante ist. Dies verletzt jedoch (b).

(c) \Rightarrow (d): Ordne $E(T) = \{e_1, \dots, e_{n-1}\}$ beliebig und setze $X := V(C)$.

(d) \Rightarrow (a): Angenommen, $E(T) = \{e_1, \dots, e_{n-1}\}$ erfüllt (d). Sei T^* ein optimaler aufspannender Baum, so dass $i := \min\{h \in \{1, \dots, n-1\} : e_h \notin E(T^*)\}$ maximal

ist. Wir werden zeigen, dass $i = \infty$, d. h. $T = T^*$. Wäre dies nicht der Fall, so gäbe es ein $X \subseteq V(G)$ mit der Eigenschaft, dass e_i eine Kante von $\delta(X)$ mit minimalem Gewicht ist und $e_j \notin \delta(X)$ für alle $j \in \{1, \dots, i-1\}$. Dann enthält $T^* + e_i$ einen Kreis C . Da $e_i \in E(C) \cap \delta(X)$, gibt es mindestens eine weitere Kante f ($f \neq e_i$) von C , die in $\delta(X)$ liegt (siehe Aufgabe 13, Kapitel 2). Beachte, dass $(T^* + e_i) - f$ ein aufspannender Baum ist. Es gilt $c(e_i) \geq c(f)$, da T^* optimal ist. Da aber $f \in \delta(X)$, haben wir auch $c(f) \geq c(e_i)$. Ferner gilt $j > i$, falls $f = e_j \in E(T)$. Somit folgt $c(f) = c(e_i)$, also ist $(T^* + e_i) - f$ ein weiterer optimaler aufspannender Baum, im Widerspruch zur Maximalität von i . \square

Der folgende Greedy-Algorithmus für das MINIMUM-SPANNING-TREE-PROBLEM wurde von Kruskal [1956] vorgeschlagen. Er kann als Spezialfall eines viel allgemeineren Greedy-Algorithmus betrachtet werden, den wir in Abschnitt 13.4 besprechen werden. Sei nun $n := |V(G)|$ und $m := |E(G)|$.

KRUSKALS ALGORITHMUS

Input: Ein zusammenhängender ungerichteter Graph G ,
Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Output: Ein aufspannender Baum T mit minimalem Gewicht.

-
- ① Sortiere die Kanten, so dass $c(e_1) \leq c(e_2) \leq \dots \leq c(e_m)$.
 - ② Setze $T := (V(G), \emptyset)$.
 - ③ **For** $i := 1$ **to** m **do**:
- If** $T + e_i$ kreisfrei **then** setze $T := T + e_i$.
-

Satz 6.4. KRUSKALS ALGORITHMUS arbeitet korrekt.

Beweis: Offensichtlich konstruiert der Algorithmus schrittweise einen aufspannenden Baum T . Ferner garantiert er, dass die Aussage (b) von Satz 6.3 gilt, also ist T optimal. \square

Die Laufzeit von KRUSKALS ALGORITHMUS ist $O(mn)$: Nach Satz 1.5 können die Kanten in $O(m \log m)$ -Zeit sortiert werden, und das Testen auf Kreise in einem Graphen mit höchstens n Kanten kann in $O(n)$ -Zeit implementiert werden (man braucht nur DFS (oder BFS) anzuwenden und zu prüfen, ob es eine Kante gibt, die nicht zum DFS-Baum gehört). Da dies m mal wiederholt wird, ergibt sich eine Gesamlaufzeit von $O(m \log m + mn) = O(mn)$. Eine effizientere Implementierung ist jedoch möglich:

Satz 6.5. KRUSKALS ALGORITHMUS kann mit $O(m \log n)$ -Laufzeit implementiert werden.

Beweis: Zunächst können parallele Kanten eliminiert werden: Alle Kanten, außer den billigsten, sind redundant. Also können wir $m = O(n^2)$ annehmen. Da die Laufzeit von ① offensichtlich $O(m \log m) = O(m \log n)$ ist, betrachten wir nun ③ näher. Wir beschreiben eine Datenstruktur, welche die Zusammenhangskomponenten von T verwaltet. In ③ wird getestet, ob das Hinzufügen einer Kante

$e_i = \{v, w\}$ zu T einen Kreis erzeugt. Dies ist äquivalent dazu, ob v und w in derselben Zusammenhangskomponente liegen.

Unsere Implementierung hält laufend ein Branching B mit $V(B) = V(G)$ bereit. Zu jeder Zeit werden die Zusammenhangskomponenten von B von denselben Knotenmengen induziert wie diejenigen von T . (Beachte jedoch, dass B i. A. keine Orientierung von T ist.)

Bei der Prüfung einer Kante $e_i = \{v, w\}$ in ③ bestimmen wir die Wurzel r_v der v enthaltenden Arboreszenz in B und die Wurzel r_w der w enthaltenden Arboreszenz in B . Die dafür benötigte Zeit ist zur Summe der Längen des r_v - v -Wege und des r_w - w -Wege in B proportional. Wir werden später zeigen, dass diese Zeit immer höchstens $\log n$ ist.

Als Nächstes prüfen wir, ob $r_v = r_w$. Ist $r_v \neq r_w$, so kommt e_i zu T hinzu und wir müssen eine Kante zu B hinzufügen. Sei $h(r)$ die maximale Länge eines Weges von r aus in B . Ist $h(r_v) \geq h(r_w)$, so fügen wir eine Kante (r_v, r_w) zu B hinzu, anderenfalls fügen wir (r_w, r_v) zu B hinzu. Ist $h(r_v) = h(r_w)$, so erhöht diese Operation $h(r_v)$ um eins, anderenfalls hat die neue Wurzel denselben h -Wert wie vorher. Somit können die h -Werte der Wurzeln leicht gespeichert werden. Am Anfang ist natürlich $B := (V(G), \emptyset)$ und $h(v) := 0$ für alle $v \in V(G)$.

Wir behaupten nun, dass eine Arboreszenz von B mit Wurzel r mindestens $2^{h(r)}$ Knoten enthält. Daraus folgt dann $h(r) \leq \log n$, womit der Beweis beendet ist. Es ist klar, dass die Behauptung am Anfang gilt. Wir müssen zeigen, dass sie nach dem Hinzufügen der Kante (x, y) zu B weiterhin gilt. Ändert sich $h(x)$ nicht, so ist dies trivial. Sonst haben wir $h(x) = h(y)$ vor der Operation, also enthalten die beiden Arboreszenzen je mindestens $2^{h(x)}$ Knoten. Damit hat die neue Arboreszenz mit Wurzel x mindestens $2 \cdot 2^{h(x)} = 2^{h(x)+1}$ Knoten, wie erwünscht. \square

Die obige Implementierung kann mittels eines Tricks weiter verbessert werden: Jedes Mal wenn die Wurzel r_v der v enthaltenden Arboreszenz in B bestimmt worden ist, entferne man alle Kanten des r_v - v -Wege P und füge eine Kante (r_x, x) für jedes $x \in V(P) \setminus \{r_v\}$ hinzu. Eine recht komplizierte Analyse zeigt, dass diese sogenannte Weg-Kompressions-Heuristik die Laufzeit von ③ fast linear macht: Sie ist $O(m\alpha(m, n))$, wobei $\alpha(m, n)$ die Umkehrfunktion der Ackermann-Funktion ist (siehe Tarjan [1975, 1983]).

Als Nächstes erwähnen wir einen weiteren bekannten Algorithmus für das MINIMUM-SPANNING-TREE-PROBLEM, der von Jarník [1930] (siehe Korte und Nešetřil [2001]), Dijkstra [1959] und Prim [1957] stammt:

PRIMS ALGORITHMUS

Input: Ein zusammenhängender ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Output: Ein aufspannender Baum T mit minimalem Gewicht.

① Wähle $v \in V(G)$. Setze $T := (\{v\}, \emptyset)$.

② **While** $V(T) \neq V(G)$ **do:**

 Wähle eine Kante $e \in \delta_G(V(T))$ mit minimalem Gewicht. Setze
 $T := T + e$.

Satz 6.6. PRIMS ALGORITHMUS arbeitet korrekt. Seine Laufzeit ist $O(m + n^2)$.

Beweis: Die Korrektheit folgt daraus, dass die Aussage (d) von Satz 6.3 bei jedem Schritt gilt (ordne die Kanten von T in der vom Algorithmus gewählten Reihenfolge).

Wiederum entfernen wir zunächst parallele Kanten. Um $O(n^2)$ -Laufzeit zu erreichen, speichern wir für jeden Knoten $v \in V(G) \setminus V(T)$ eine billigste Kante $e \in E(V(T), \{v\})$. Diese Kanten nennen wir die Kandidaten. Die Initialisierung der Kandidaten nach ① braucht $O(n)$ -Zeit. Jede Wahl einer billigsten Kante aus der Kandidatenmenge braucht $O(n)$ -Zeit. Die Aktualisierung der Kandidatenliste kann durch das Scannen der mit dem zu $V(T)$ hinzugefügten Knoten inzidenten Kanten erreicht werden und braucht somit auch $O(n)$ -Zeit. Da die **While**-Schleife in ② $n - 1$ Iterationen hat, ist die $O(n^2)$ Schranke bewiesen. \square

Die Laufzeit kann durch effiziente Datenstrukturen verbessert werden. Sei $l_{T,v} := \min\{c(e) : e \in E(V(T), \{v\})\}$. Wir speichern die Menge $\{(v, l_{T,v}) : v \in V(G) \setminus V(T), l_{T,v} < \infty\}$ in einer Datenstruktur, die sogenannte Prioritätswarteschlange oder auch Heap, welche uns erlaubt, Elemente hinzuzufügen, ein Element (v, l) mit minimalem l zu finden und zu entfernen, und den sogenannten Schlüssel l eines Elementes (v, l) zu verringern. PRIMS ALGORITHMUS kann nun folgendermaßen formuliert werden:

- ① Wähle $v \in V(G)$. Setze $T := (\{v\}, \emptyset)$.
Sei $l_w := \infty$ für $w \in V(G) \setminus \{v\}$.
- ② **While** $V(T) \neq V(G)$ **do**:
For $e = \{v, w\} \in E(\{v\}, V(G) \setminus V(T))$ **do**:
If $c(e) < l_w < \infty$ **then** setze $l_w := c(e)$ und **DECREASEKEY**(w, l_w).
If $l_w = \infty$ **then** setze $l_w := c(e)$ und **INSERT**(w, l_w).
 $(v, l_v) := \text{DELETEMIN}$.
Sei $e \in E(V(T), \{v\})$ mit $c(e) = l_v$. Setze $T := T + e$.

Es gibt mehrere Möglichkeiten, einen Heap zu implementieren. Eine sehr effiziente ist der sogenannte Fibonacci-Heap, der von Fredman und Tarjan [1987] vorgeschlagen worden ist. Unsere Darstellung basiert auf Schrijver [2003]:

Satz 6.7. Es ist möglich, eine Datenstruktur für eine endliche Menge (anfänglich leer) zu führen, wobei jedem Element u eine reelle Zahl $d(u)$, sein sogenannter Schlüssel, zugeteilt wird, und in $O(m + p + n \log p)$ -Zeit jede beliebige Folge von

- p **INSERT**-Operationen (ein Element u mit Schlüssel $d(u)$ hinzufügen);
- n **DELETEMIN**-Operationen (ein Element u mit minimalem $d(u)$ finden und entfernen);
- m **DECREASEKEY**-Operationen ($d(u)$ auf einen bestimmten Wert für ein Element u verringern)

zu tätigen.

Beweis: Die endliche Menge, die wir U nennen, wird in einem sogenannten Fibonacci-Heap gespeichert, d. h. in einem Branching (U, E) mit einer Funktion $\varphi : U \rightarrow \{0, 1\}$ mit den folgenden Eigenschaften:

- (i) Ist $(u, v) \in E$, so folgt $d(u) \leq d(v)$. (Dies ist die sogenannte Heap-Ordnung.)
- (ii) Für jedes $u \in U$ können die Kinder von u mit den ganzen Zahlen 1 bis $|\delta^+(u)|$ durchnummieriert werden, so dass für jedes i das i -te Kind v die Ungleichung $|\delta^+(v)| + \varphi(v) \geq i - 1$ erfüllt.
- (iii) Sind u und v zwei verschiedene Wurzeln ($\delta^-(u) = \delta^-(v) = \emptyset$), so gilt $|\delta^+(u)| \neq |\delta^+(v)|$.

Aus Bedingung (ii) folgt:

- (iv) Ist der Ausgangsgrad eines Knotens u mindestens k , so sind mindestens $\sqrt{2}^k$ Knoten von u aus erreichbar.

Der Beweis von (iv) erfolgt mittels Induktion über k , wobei die Fälle $k = 0$ und $k = 1$ trivial sind. Sei also u ein Knoten mit $|\delta^+(u)| \geq k \geq 1$ und v ein Kind von u mit $|\delta^+(v)| \geq k - 2$ (wegen (ii) gibt es ein solches v). Wenden wir die Induktionsvoraussetzung auf v in (U, E) und auf u in $(U, E \setminus \{(u, v)\})$ an, so folgt, dass mindestens $\sqrt{2}^{k-2}$ bzw. $\sqrt{2}^{k-1}$ Knoten erreichbar sind. Nun folgt (iv) mittels der Ungleichung $\sqrt{2}^k \leq \sqrt{2}^{k-2} + \sqrt{2}^{k-1}$.

Insbesondere folgt aus (iv), dass $|\delta^+(u)| \leq 2 \log |U|$ für alle $u \in U$. Mit (iii) können wir somit die Wurzeln von (U, E) mittels einer Funktion $b : \{0, 1, \dots, \lfloor 2 \log |U| \rfloor\} \rightarrow U$ mit $b(|\delta^+(u)|) = u$ für jede Wurzel u speichern. Beachte: Aus $b(i) = u$ folgt weder $|\delta^+(u)| = i$ noch, dass u eine Wurzel ist.

Zusätzlich führen wir Buch über eine doppelt verkettete Liste der Kinder (in beliebiger Reihenfolge), über einen Zeiger zum Vorgänger (falls er existiert), und über den Ausgangsgrad eines jeden Knotens. Wir werden nun zeigen, wie die INSERT-, DELETEMIN- und DECREASEKEY-Operationen implementiert werden.

$\text{INSERT}(v, d(v))$ wird folgendermaßen implementiert: Setze $\varphi(v) := 0$ und wende an:

$\text{PLANT}(v)$:

- ① Setze $r := b(|\delta^+(v)|)$.
 - if** r ist eine Wurzel mit $r \neq v$ und $|\delta^+(r)| = |\delta^+(v)|$ **then**:
 - if** $d(r) \leq d(v)$ **then** füge (r, v) zu E hinzu und $\text{PLANT}(r)$.
 - if** $d(v) < d(r)$ **then** füge (v, r) zu E hinzu und $\text{PLANT}(v)$.
 - else** setze $b(|\delta^+(v)|) := v$.

Da (U, E) immer ein Branching ist, terminiert die Rekursion. Beachte auch, dass (i), (ii) und (iii) erhalten bleiben.

DELETEMIN wird implementiert, indem man $b(i)$ für $i = 0, \dots, \lfloor 2 \log |U| \rfloor$ scannnt, um ein Element u mit $\delta^-(u) = \emptyset$ und minimalem $d(u)$ zu finden, dann u und seine inzidenten Kanten entfernt und schließlich schrittweise $\text{PLANT}(v)$ für jedes (frühere) Kind v von u anwendet.

$\text{DECREASEKEY}(v, d(v))$ ist etwas komplizierter. Sei P der längste in v endende Weg in (U, E) mit der Eigenschaft $\varphi(u) = 1$ für jeden internen Knoten u . Setze nun $\varphi(u) := 1 - \varphi(u)$ für alle $u \in V(P) \setminus \{v\}$, entferne alle Kanten von P aus E und wende $\text{PLANT}(z)$ für jedes $z \in V(P)$ an, das eine Wurzel des neuen Waldes ist.

Um zu sehen, dass (ii) dabei erhalten bleibt, brauchen wir nur den Vorgänger des Anfangsknotens x von P zu betrachten, falls es existiert. Dann ist aber x keine Wurzel, also wird der Wert $\varphi(x)$ von 0 auf 1 erhöht, um das verlorene Kind zu kompensieren.

Zum Schluss schätzen wir die Laufzeit ab. Da φ höchstens m mal steigt (höchstens einmal pro DECREASEKEY), wird φ höchstens m mal verringert. Also ist die Summe der Längen der Wege P in allen DECREASEKEY-Operationen höchstens $m + m$. Somit werden insgesamt höchstens $2m + 2n \log p$ Kanten entfernt (da jede DELETEMIN-Operation bis zu $2 \log p$ Kanten entfernen kann). Es folgt, dass insgesamt höchstens $2m + 2n \log p + p - 1$ Kanten hinzugefügt werden, womit die Gesamlaufzeit $O(m + p + n \log p)$ ist. \square

Korollar 6.8. *Wird PRIMS ALGORITHMUS mit dem Fibonacci-Heap implementiert, so löst er das MINIMUM-SPANNING-TREE-PROBLEM in $O(m + n \log n)$ -Zeit.*

Beweis: Es werden höchstens $n - 1$ INSERT-, $n - 1$ DELETEMIN- und m DECREASEKEY-Operationen getätigten. \square

Mit einer ausgeklügelteren Implementierung kann man eine $O(m \log \beta(n, m))$ -Laufzeit erreichen, wobei $\beta(n, m) = \min\{i : \log^{(i)} n \leq \frac{m}{n}\}$, siehe Fredman und Tarjan [1987], Gabow, Galil und Spencer [1989] und Gabow et al. [1986]. Der schnellste deterministische Algorithmus stammt von Pettie und Ramachandran [2002]. Er ist mindestens so schnell wie der Algorithmus von Chazelle [2000] mit $O(m\alpha(m, n))$ -Laufzeit, wobei α die Umkehrfunktion der Ackermann-Funktion ist.

Mit einem anderen Rechenmodell erreichten Fredman und Willard [1994] sogar lineare Laufzeit. Auch gibt es einen randomisierten Algorithmus, welcher einen aufspannenden Baum minimalen Gewichtes mit linearer erwarteter Laufzeit findet (Karger, Klein und Tarjan [1995]; ein solcher Algorithmus, der immer eine optimale Lösung findet, heißt Las-Vegas-Algorithmus). Dieser Algorithmus benutzt ein (deterministisches) Verfahren zur Feststellung der Optimalität eines aufspannenden Baumes; ein Algorithmus für dieses Problem mit linearer Laufzeit wurde von Dixon, Rauch und Tarjan [1992] beschrieben; siehe auch King [1997].

Das MINIMUM-SPANNING-TREE-PROBLEM für planare Graphen kann mit linearer Laufzeit (deterministisch) gelöst werden (Cheriton und Tarjan [1976]). Das Problem, einen aufspannenden Baum mit minimalem Gewicht für eine n -punktige Menge in der Ebene zu finden, kann mit $O(n \log n)$ -Laufzeit gelöst werden (siehe Aufgabe 14). PRIMS ALGORITHMUS kann für solche Instanzen recht effizient sein, da man für eine effektive Bestimmung der nächsten Nachbarn in der Ebene geeignete Datenstrukturen einsetzen kann.

6.2 Arboreszenzen mit minimalem Gewicht

Nahe liegende gerichtete Versionen des MAXIMUM-WEIGHT-FOREST-PROBLEMS und des MINIMUM-SPANNING-TREE-PROBLEMS lauten wie folgt:

MAXIMUM-WEIGHT-BRANCHING-PROBLEM

Instanz: Ein Digraph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Bestimme ein Branching in G mit maximalem Gewicht.

MINIMUM-WEIGHT-ARBORESCENCE-PROBLEM

Instanz: Ein Digraph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Bestimme eine aufspannende Arboreszenz in G mit minimalem Gewicht, oder entscheide, dass es keine solche gibt.

Gelegentlich möchten wir die Wurzel vorab angeben:

MINIMUM-WEIGHT-ROOTED-ARBORESCENCE-PROBLEM

Instanz: Ein Digraph G , ein Knoten $r \in V(G)$, Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Bestimme eine aufspannende Arboreszenz in G mit Wurzel r und mit minimalem Gewicht, oder entscheide, dass es keine solche gibt.

Wie im ungerichteten Fall sind diese drei Probleme äquivalent:

Proposition 6.9. *Die folgenden drei Probleme: das MAXIMUM-WEIGHT-BRANCHING-PROBLEM, das MINIMUM-WEIGHT-ARBORESCENCE-PROBLEM und das MINIMUM-WEIGHT-ROOTED-ARBORESCENCE-PROBLEM sind äquivalent.*

Beweis: Gegeben sei eine Instanz (G, c) des MINIMUM-WEIGHT-ARBORESCENCE-PROBLEMS. Sei $c'(e) := K - c(e)$ für alle $e \in E(G)$, wobei $K = 1 + \sum_{e \in E(G)} |c(e)|$. Dann ist die Instanz (G, c') des MAXIMUM-WEIGHT-BRANCHING-PROBLEMS äquivalent, denn für je zwei Branchings B und B' mit $|E(B)| > |E(B')|$ haben wir $c'(E(B)) > c'(E(B'))$ (und die Branchings mit $n - 1$ Kanten sind gerade die aufspannenden Arboreszenzen).

Gegeben sei eine Instanz (G, c) des MAXIMUM-WEIGHT-BRANCHING-PROBLEMS. Sei $G' := (V(G) \cup \{r\}, E(G) \cup \{(r, v) : v \in V(G)\})$. Sei ferner $c'(e) := -c(e)$ für $e \in E(G)$ und $c(e) := 0$ für $e \in E(G') \setminus E(G)$. Dann ist die Instanz (G', r, c') des MINIMUM-WEIGHT-ROOTED-ARBORESCENCE-PROBLEMS äquivalent.

Gegeben sei schließlich eine Instanz (G, r, c) des MINIMUM-WEIGHT-ROOTED-ARBORESCENCE-PROBLEMS. Sei $G' := (V(G) \cup \{s\}, E(G) \cup \{(s, r)\})$ und $c((s, r)) := 0$. Dann ist die Instanz (G', c) des MINIMUM-WEIGHT-ARBORESCENCE-PROBLEMS äquivalent. \square

In diesem Abschnitt werden wir uns nur noch mit dem MAXIMUM-WEIGHT-BRANCHING-PROBLEM befassen. Dieses Problem ist nicht so einfach wie die ungerichtete Version, das MAXIMUM-WEIGHT-FOREST-PROBLEM. Es ist z. B. jeder inklusionsmaximale Wald auch ein Wald maximaler Kardinalität, die fetten Kanten in Abb. 6.1 bilden jedoch ein inklusionsmaximales Branching, dessen Kardinalität nicht maximal ist.

Zur Erinnerung: Ein Branching ist ein Graph B mit $|\delta_B^-(x)| \leq 1$ für alle $x \in V(B)$, dessen zugrunde liegender ungerichteter Graph ein Wald ist. Äquivalent

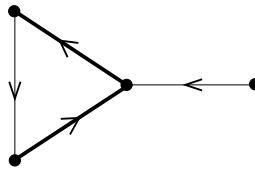


Abbildung 6.1.

ausgedrückt: Ein Branching ist ein azyklischer Digraph B mit $|\delta_B^-(x)| \leq 1$ für alle $x \in V(B)$, siehe Satz 2.5(g):

Proposition 6.10. *Sei B ein Digraph mit $|\delta_B^-(x)| \leq 1$ für alle $x \in V(B)$. Dann enthält B einen Kreis genau dann, wenn der zugrunde liegende ungerichtete Graph einen Kreis enthält.* \square

Nun sei G ein Digraph und $c : E(G) \rightarrow \mathbb{R}_+$. Wir können negative Gewichte ignorieren, da solche Kanten niemals in einem optimalen Branching auftreten werden. Eine Anfangsidee für einen möglichen Algorithmus wäre: Man nehme in jedem Knoten die beste ankommende Kante. Natürlich kann der resultierende Graph Kreise enthalten. Da Branchings aber kreisfrei sind, müssen wir mindestens eine Kante pro Kreis entfernen. Das folgende Lemma besagt, dass eine genügt.

Lemma 6.11. (Karp [1972]) *Sei B_0 ein Teilgraph von G mit maximalem Gewicht und $|\delta_{B_0}^-(v)| \leq 1$ für alle $v \in V(B_0)$. Dann gibt es ein optimales Branching B von G mit $|E(C) \setminus E(B)| = 1$ für jeden Kreis C in B_0 .*

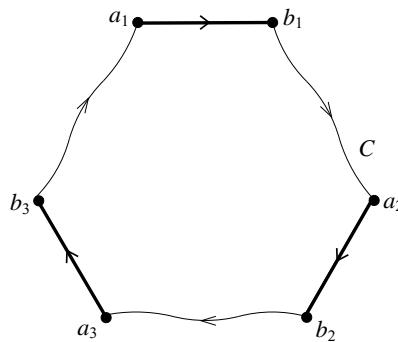


Abbildung 6.2.

Beweis: Sei B ein optimales Branching von G mit so vielen Kanten von B_0 wie möglich. Sei C ein Kreis in B_0 und $E(C) \setminus E(B) = \{(a_1, b_1), \dots, (a_k, b_k)\}$. Angenommen, $k \geq 2$ und $a_1, b_1, a_2, b_2, a_3, \dots, b_k$ liegen auf C in dieser Reihenfolge (siehe Abb. 6.2).

Wir behaupten nun, dass B einen b_i - b_{i-1} -Weg für jedes $i = 1, \dots, k$ ($b_0 := b_k$) enthält. Dies ist jedoch ein Widerspruch, da diese Wege eine geschlossene

Kantenfolge in B bilden, was es in einem Branching nicht geben kann. Also ist $k = 1$.

Sei $i \in \{1, \dots, k\}$. Es bleibt zu zeigen, dass B einen b_i - b_{i-1} -Weg enthält. Betrachte B' mit $V(B') = V(G)$ und $E(B') := \{(x, y) \in E(B) : y \neq b_i\} \cup \{(a_i, b_i)\}$.

Es kann B' kein Branching sein, da es optimal wäre und mehr Kanten von B_0 enthielte als B . Somit enthält B' nach Proposition 6.10 einen Kreis, d. h. B enthält einen b_i - a_i -Weg P . Da $k \geq 2$, liegt P nicht gänzlich auf C . Sei e die letzte nicht auf C liegende Kante von P . Offensichtlich ist $e = (x, b_{i-1})$ für ein bestimmtes x , somit enthält P (also auch B) einen b_i - b_{i-1} -Weg. \square

Die grundlegende Idee des Algorithmus von Edmonds [1967] ist, B_0 wie oben zu bestimmen und dann jeden Kreis von B_0 in G zu kontrahieren. Wählen wir die Gewichte für den resultierenden Graphen G_1 auf geeignete Weise, so wird jedes optimale Branching in G_1 einem optimalen Branching in G entsprechen.

EDMONDS' BRANCHING-ALGORITHMUS

Input: Ein Digraph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$.

Output: Ein Branching B von G mit maximalem Gewicht.

- ① Setze $i := 0$, $G_0 := G$ und $c_0 := c$.
- ② Sei B_i ein Teilgraph von G_i mit maximalem Gewicht und $|\delta_{B_i}^-(v)| \leq 1$ für alle $v \in V(B_i)$.
- ③ If B_i kreisfrei then setze $B := B_i$ und go to ⑤.
- ④ Sei C die Menge der Kreise in B_i . Kontrahiere diese Kreise:
Sei $V(G_{i+1}) := C \cup (V(G_i) \setminus \bigcup_{C \in C} V(C))$.
Für $e = (v, w) \in E(G_i)$ sei $e' = (v', w')$ und $\Phi_{i+1}(e') := e$, wobei
 $v' = C$ falls $v \in V(C)$ für $C \in C$, und $v' = v$ falls $v \notin \bigcup_{C \in C} V(C)$, und
 $w' = C$ falls $w \in V(C)$ für $C \in C$, und $w' = w$ falls $w \notin \bigcup_{C \in C} V(C)$.
Sei $E(G_{i+1}) := \{e' = (v', w') : e \in E(G_i), v' \neq w'\}$
(parallele Kanten können auftreten).
Für $e = (v, w) \in E(G_i)$ mit $e' = (v', w') \in E(G_{i+1})$ setze
 $c_{i+1}(e') := c_i(e)$ falls $w' \notin C$, und
 $c_{i+1}(e') := c_i(e) - c_i(\alpha(e, C)) + c_i(e_C)$ falls $w' = C \in C$, wobei
 $\alpha(e, C) \in \delta_C^-(w)$ und e_C eine billigste Kante von C ist.
Setze $i := i + 1$ und go to ②.
- ⑤ While $i > 0$ do:
 - Setze $B' := (V(G_{i-1}), \{\Phi_i(e) : e \in E(B)\})$.
 - For jeden Kreis C von B_{i-1} do:
 - If es gibt eine Kante $e \in \delta_{B'}^-(V(C))$
 - then setze $E(B') := E(B') \cup (E(C) \setminus \{\alpha(e, C)\})$
 - else setze $E(B') := E(B') \cup (E(C) \setminus \{e_C\})$.
 - Setze $B := B'$ und $i := i - 1$.

Dieser Algorithmus wurde unabhängig auch von Chu und Liu [1965] und von Bock [1971] entdeckt.

Satz 6.12. (Edmonds [1967]) EDMONDS' BRANCHING-ALGORITHMUS arbeitet korrekt.

Beweis: Wir werden zeigen, dass B in ⑤ immer ein optimales Branching von G_i ist. Dies ist trivial bei der ersten Ankunft an ⑤. Also müssen wir zeigen, dass eine Iteration von ⑤ ein optimales Branching B von G_i in ein optimales Branching B' von G_{i-1} transformiert.

Sei B_{i-1}^* ein Branching von G_{i-1} mit $|E(C) \setminus E(B_{i-1}^*)| = 1$ für jeden Kreis C von B_{i-1} . Es gehe B_i^* aus B_{i-1}^* durch Kontraktion der Kreise von B_{i-1} hervor. Dann ist B_i^* ein Branching von G_i . Ferner haben wir

$$c_{i-1}(B_{i-1}^*) \leq c_i(B_i^*) + \sum_{C: \text{Kreis von } B_{i-1}} (c_{i-1}(E(C)) - c_{i-1}(e_C)).$$

Nach der Induktionsvoraussetzung ist B ein optimales Branching von G_i , also gilt $c_i(B) \geq c_i(B_i^*)$. Daraus folgern wir, dass

$$\begin{aligned} c_{i-1}(B_{i-1}^*) &\leq c_i(B) + \sum_{C: \text{Kreis von } B_{i-1}} (c_{i-1}(E(C)) - c_{i-1}(e_C)) \\ &= c_{i-1}(B'). \end{aligned}$$

Mit Lemma 6.11 folgt, dass B' ein optimales Branching von G_{i-1} ist. \square

Dieser Beweis stammt von Karp [1972]. Der ursprüngliche Beweis von Edmonds basiert auf einer LP-Formulierung (siehe Korollar 6.15). Man sieht leicht, dass EDMONDS' BRANCHING-ALGORITHMUS $O(mn)$ -Laufzeit hat, wobei $m = |E(G)|$ und $n = |V(G)|$: Es gibt höchstens n Iterationen (d. h. es gilt $i \leq n$ zu jedem Zeitpunkt des Algorithmus), und jede Iteration kann in $O(m)$ -Zeit implementiert werden.

Die beste bisher gefundene Schranke stammt von Gabow et al. [1986] und wurde mit einem Fibonacci-Heap bestimmt; ihr Branching Algorithmus läuft in $O(m + n \log n)$ -Zeit.

6.3 Polyedrische Darstellungen

Eine polyedrische Darstellung des MINIMUM-SPANNING-TREE-PROBLEMS lautet folgendermaßen:

Satz 6.13. (Edmonds [1970]) Gegeben sei ein zusammenhängender ungerichteter Graph G mit $n := |V(G)|$. Dann ist das Polytop $P :=$

$$\left\{ x \in [0, 1]^{|E(G)|} : \sum_{e \in E(G)} x_e = n - 1, \sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ für } \emptyset \neq X \subset V(G) \right\}$$

ganzzahlig. Die Ecken dieses Polytops sind gerade die Inzidenzvektoren der aufspannenden Bäume von G . (P heißt das **Baumpolytop** von G .)

Beweis: Sei T ein aufspannender Baum von G und x der Inzidenzvektor von $E(T)$. Nach Satz 2.4 folgt sofort, dass $x \in P$. Ferner ist x eine Ecke von P , da $x \in \{0, 1\}^{E(G)}$.

Sei andererseits x eine ganzzahlige Ecke von P . Dann ist x der Inzidenzvektor der Kantenmenge eines kreisfreien Teilgraphen H mit $n - 1$ Kanten. Wiederum folgt mit Satz 2.4, dass H ein aufspannender Baum ist.

Also genügt es zu zeigen, dass P ganzzahlig ist (beachte Satz 5.14). Sei $c : E(G) \rightarrow \mathbb{R}$ und T der durch Anwendung von KRUSKALS ALGORITHMUS auf (G, c) hervorgehende Baum (gleichgewichtete Kanten werden willkürlich sortiert). Setze $E(T) = \{f_1, \dots, f_{n-1}\}$, wobei die f_i in der durch den Algorithmus gegebenen Reihenfolge genommen werden. Insbesondere gilt $c(f_1) \leq \dots \leq c(f_{n-1})$. Sei $X_k \subseteq V(G)$ diejenige Zusammenhangskomponente von $(V(G), \{f_1, \dots, f_k\})$, welche f_k ($k = 1, \dots, n - 1$) enthält.

Sei x^* der Inzidenzvektor von $E(T)$. Wir werden nun zeigen, dass x^* eine optimale Lösung des folgenden LP ist:

$$\begin{aligned} \min \quad & \sum_{e \in E(G)} c(e)x_e \\ \text{bzgl.} \quad & \sum_{e \in E(G)} x_e = n - 1 \\ & \sum_{e \in E(G[X])} x_e \leq |X| - 1 \quad (\emptyset \neq X \subset V(G)) \\ & x_e \geq 0 \quad (e \in E(G)). \end{aligned}$$

Hier definieren wir für jedes $\emptyset \neq X \subset V(G)$ eine duale Variable z_X und noch eine weitere duale Variable $z_{V(G)}$ für die Gleichungsnebenbedingung. Dann lautet das duale LP:

$$\begin{aligned} \max \quad & - \sum_{\emptyset \neq X \subseteq V(G)} (|X| - 1)z_X \\ \text{bzgl.} \quad & - \sum_{e \subseteq X \subseteq V(G)} z_X \leq c(e) \quad (e \in E(G)) \\ & z_X \geq 0 \quad (\emptyset \neq X \subset V(G)). \end{aligned}$$

Beachte, dass es keine Nebenbedingung $z_{V(G)} \geq 0$ gibt. Nun setzen wir $z_{X_k}^* := c(f_l) - c(f_k)$ für $k = 1, \dots, n - 2$, wobei l der erste Index größer als k ist, für den $f_l \cap X_k \neq \emptyset$. Sei $z_{V(G)}^* := -c(f_{n-1})$ und $z_X^* := 0$ für alle $X \notin \{X_1, \dots, X_{n-1}\}$.

Für jedes $e = \{v, w\}$ haben wir dann

$$- \sum_{e \subseteq X \subseteq V(G)} z_X^* = c(f_i),$$

wobei i der kleinste Index mit $v, w \in X_i$ ist. Ferner gilt $c(f_i) \leq c(e)$, da v und w in verschiedenen Zusammenhangskomponenten von $(V(G), \{f_1, \dots, f_{i-1}\})$ liegen. Somit ist z^* eine zulässige Lösung.

Für jedes $e \in E(G)$ mit $x_e^* > 0$ gilt $e \in E(T)$ und somit

$$-\sum_{e \subseteq X \subseteq V(G)} z_X^* = c(e),$$

also wird die entsprechende duale Nebenbedingung mit Gleichheit erfüllt. Schließlich gilt auch noch $z_X^* > 0$, also ist $T[X]$ zusammenhängend. Damit wird die entsprechende primale Nebenbedingung mit Gleichheit erfüllt. Anders ausgedrückt, die primalen und dualen Bedingungen des komplementären Schlupfes sind erfüllt, also sind x^* und z^* nach Korollar 3.23 optimale Lösungen des primalen bzw. dualen LP. \square

In der Tat haben wir sogar bewiesen, dass das Ungleichungssystem in Satz 6.13 TDI ist. Auch weisen wir darauf hin, dass der obige Beweis einen alternativen Beweis des korrekten Verlaufs von KRUSKALS ALGORITHMUS (Satz 6.4) liefert. Eine weitere Darstellung des Baumpolytops wird in Aufgabe 19 vorgestellt. Ferner gibt es auch eine polynomiale LP-Formulierung; siehe Aufgabe 20.

Ersetzen wir die Nebenbedingung $\sum_{e \in E(G)} x_e = n - 1$ durch $\sum_{e \in E(G)} x_e \leq n - 1$, so erhalten wir die konvexe Hülle der Inzidenzvektoren aller Wälder in G (siehe Aufgabe 21). Eine Verallgemeinerung dieser Resultate ist Edmonds' Charakterisierung des Matroid-Polytops (Satz 13.21).

Wir wenden uns nun der polyedrischen Darstellung des MINIMUM-WEIGHT-ROOTED-ARBORESCENCE-PROBLEMS zu. Zunächst beweisen wir ein klassisches Resultat von Fulkerson. Wir erinnern daran, dass ein r -Schnitt die Kantenmenge $\delta^+(S)$ für ein $S \subset V(G)$ mit $r \in S$ ist.

Satz 6.14. (Fulkerson [1974]) *Sei G ein Digraph mit den Gewichten $c : E(G) \rightarrow \mathbb{Z}_+$ und $r \in V(G)$ mit der Eigenschaft, dass G eine aufspannende Arboreszenz mit Wurzel r enthält. Dann ist das minimale Gewicht einer aufspannenden Arboreszenz mit Wurzel r gleich der maximalen Anzahl t der r -Schnitte C_1, \dots, C_t (Wiederholungen erlaubt) mit der Eigenschaft, dass keine Kante e in mehr als $c(e)$ dieser Schnitte enthalten ist.*

Beweis: Sei A die Matrix, deren Spalten den Kanten von G entsprechen und deren Zeilen die Inzidenzvektoren aller r -Schnitte sind. Betrachte das LP

$$\min\{cx : Ax \geq \mathbf{1}, x \geq 0\},$$

und sein duales LP

$$\max\{\mathbf{1}y : yA \leq c, y \geq 0\}.$$

Nach Teil (e) von Satz 2.5 müssen wir zeigen, dass diese beiden LPs für jedes nichtnegative ganzzahlige c ganzzahlige optimale Lösungen haben. Nach Korollar 5.16 genügt es zu zeigen, dass das System $Ax \geq \mathbf{1}$, $x \geq 0$ TDI ist. Dazu benutzen wir Lemma 5.24.

Da das duale LP genau dann zulässig ist, wenn c nichtnegativ ist, setzen wir $c : E(G) \rightarrow \mathbb{Z}_+$. Sei y eine optimale Lösung von $\max\{\mathbf{1}y : yA \leq c, y \geq 0\}$ mit

$$\sum_{\emptyset \neq X \subseteq V(G) \setminus \{r\}} y_{\delta^-(X)} |X|^2 \quad (6.1)$$

so groß wie möglich. Wir behaupten nun, dass $\mathcal{F} := \{X : y_{\delta^-(X)} > 0\}$ laminar ist. Angenommen, es gäbe $X, Y \in \mathcal{F}$ mit $X \cap Y \neq \emptyset$, $X \setminus Y \neq \emptyset$ und $Y \setminus X \neq \emptyset$ (Abb. 6.3). Sei ferner $\epsilon := \min\{y_{\delta^-(X)}, y_{\delta^-(Y)}\}$. Setze $y'_{\delta^-(X)} := y_{\delta^-(X)} - \epsilon$, $y'_{\delta^-(Y)} := y_{\delta^-(Y)} - \epsilon$, $y'_{\delta^-(X \cap Y)} := y_{\delta^-(X \cap Y)} + \epsilon$, $y'_{\delta^-(X \cup Y)} := y_{\delta^-(X \cup Y)} + \epsilon$ und $y'(S) := y(S)$ für alle weiteren r -Schnitte S . Beachte, dass $y'A \leq yA$, somit ist y' eine zulässige duale Lösung. Da $\|y = \|y'$, ist sie auch optimal. Dies widerspricht jedoch der Maximalitätseigenschaft von y , da (6.1) für y' größer ist. (Für Zahlen $a > b \geq c > d > 0$ mit $a + d = b + c$ gilt $a^2 + d^2 > b^2 + c^2$.)

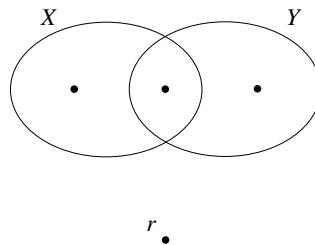


Abbildung 6.3.

Nun sei A' die Untermatrix von A , deren Zeilen den Elementen von \mathcal{F} entsprechen. Es ist A' die Einweg-Schnitt-Inzidenz-Matrix einer laminaren Familie (wir müssen, genauer gesagt, den aus G durch Umorientierung aller Kanten resultierenden Graphen betrachten). Nach Satz 5.29 ist A' dann vollständig-unimodular, wie erwünscht. \square

Der obige Beweis liefert auch die versprochene polyedrische Darstellung:

Korollar 6.15. (Edmonds [1967]) *Sei G ein Digraph mit den Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ und $r \in V(G)$ mit der Eigenschaft, dass G eine aufspannende Arboreszenz mit Wurzel r enthält. Dann hat das LP*

$$\min \left\{ cx : x \geq 0, \sum_{e \in \delta^+(X)} x_e \geq 1 \text{ für alle } X \subset V(G) \text{ mit } r \in X \right\}$$

eine ganzzahlige optimale Lösung (die der Inzidenzvektor einer aufspannenden Arboreszenz mit Wurzel r und minimalem Gewicht ist, eventuell mit zusätzlichen Kanten mit Gewicht Null). \square

Darstellungen der konvexen Hüllen der Inzidenzvektoren aller Branchings oder aufspannenden Arboreszenzen mit Wurzel r werden in den Aufgaben 22 und 23 besprochen.

6.4 Das Packen von aufspannenden Bäumen und Arboreszenzen

Suchen wir mehr als einen aufspannenden Baum bzw. mehr als eine aufspannende Arboreszenz, so ist es hilfreich, klassische Sätze von Tutte, Nash-Williams und Edmonds heranzuziehen. Zunächst beweisen wir Tuttes Satz zum Packen von aufspannenden Bäumen. Unser Beweis stammt im Wesentlichen von Mader (siehe Diestel [1996]) und benutzt das folgende Lemma:

Lemma 6.16. *Sei G ein ungerichteter Graph und $F = (F_1, \dots, F_k)$ ein k -Tupel paarweise kantendisjunkter Wälder in G mit $|E(F)|$ maximal, wobei $E(F) := \bigcup_{i=1}^k E(F_i)$. Sei $e \in E(G) \setminus E(F)$. Dann gibt es eine Menge $X \subseteq V(G)$ mit $e \subseteq X$, so dass $F_i[X]$ für jedes $i \in \{1, \dots, k\}$ zusammenhängend ist.*

Beweis: Gegeben seien zwei k -Tupel $F' = (F'_1, \dots, F'_k)$ und $F'' = (F''_1, \dots, F''_k)$ von kantendisjunkten Wäldern. Wir sagen, dass F'' aus F' durch Austausch von e' und e'' hervorgeht, wenn $F''_j = (F'_j \setminus e') \cup e''$ für ein bestimmtes j und $F''_i = F'_i$ für alle $i \neq j$. Sei \mathcal{F} die Menge derjenigen k -Tupel paarweise kantendisjunkter Wälder, die mittels Folgen solcher Austausche aus F hervorgehen. Sei $\bar{E} := E(G) \setminus (\bigcap_{F' \in \mathcal{F}} E(F'))$ und $\bar{G} := (V(G), \bar{E})$. Es ist $F \in \mathcal{F}$ und somit $e \in \bar{E}$. Sei X die Knotenmenge der e enthaltenden Zusammenhangskomponente von \bar{G} . Wir werden nun beweisen, dass $F_i[X]$ für jedes i zusammenhängend ist.

Behauptung: Für jedes $F' = (F'_1, \dots, F'_k) \in \mathcal{F}$ und jedes $\bar{e} = \{v, w\} \in E(\bar{G}[X]) \setminus E(F')$ gibt es einen $v-w$ -Weg in $F'_i[X]$ für alle $i \in \{1, \dots, k\}$.

Um dies zu beweisen, nehmen wir ein festes $i \in \{1, \dots, k\}$. Da $F' \in \mathcal{F}$ und $|E(F')| = |E(F)|$ maximal ist, enthält $F'_i + \bar{e}$ einen Kreis C . Nun gilt $F'_{e'} \in \mathcal{F}$ für alle $e' \in E(C) \setminus \{\bar{e}\}$, wobei $F'_{e'}$ aus F' durch Austausch von e' und \bar{e} hervorgeht. Damit ist $E(C) \subseteq \bar{E}$, also ist $C - \bar{e}$ ein $v-w$ -Weg in $F'_i[X]$. Damit ist die Behauptung bewiesen.

Da $\bar{G}[X]$ zusammenhängend ist, genügt es zu beweisen, dass es für jedes $\bar{e} = \{v, w\} \in E(\bar{G}[X])$ und jedes i einen $v-w$ -Weg in $F_i[X]$ gibt.

Sei also $\bar{e} = \{v, w\} \in E(\bar{G}[X])$. Da $\bar{e} \in \bar{E}$, gibt es ein k -Tupel $F' = (F'_1, \dots, F'_k) \in \mathcal{F}$ mit $\bar{e} \notin E(F')$. Mit der Behauptung folgt dann, es gibt einen $v-w$ -Weg in $F'_i[X]$ für jedes i .

Nun gibt es eine Folge $F = F^{(0)}, F^{(1)}, \dots, F^{(s)} = F'$ von Elementen aus \mathcal{F} mit der Eigenschaft, dass $F^{(r+1)}$ aus $F^{(r)}$ durch Austausch einer Kante hervorgeht ($r = 0, \dots, s-1$). Es genügt zu zeigen: Gibt es einen $v-w$ -Weg in $F_i^{(r+1)}[X]$, so auch in $F_i^{(r)}[X]$ ($r = 0, \dots, s-1$).

Nehmen wir also an, dass $F_i^{(r+1)}$ aus $F_i^{(r)}$ durch Austausch von e_r und e_{r+1} hervorgeht, und sei P ein $v-w$ -Weg in $F_i^{(r+1)}[X]$. Falls P die Kante $e_{r+1} = \{x, y\}$ nicht enthält, ist P auch ein Weg in $F_i^{(r)}[X]$. Andernfalls können wir e_{r+1} ersetzen: Betrachte den $x-y$ -Weg Q in $F_i^{(r)}[X]$, der wegen $e_{r+1} \in E(\bar{G}[X])$ nach der Behauptung existiert. Da $(E(P) \setminus \{e_{r+1}\}) \cup E(Q)$ einen $v-w$ -Weg in $F_i^{(r)}[X]$ enthält, ist der Beweis abgeschlossen. \square

Wir können nun Tuttes Satz über paarweise kantendisjunkte aufspannende Bäume beweisen. Ein **Multischnitt** in einem ungerichteten Graphen G ist eine Kantenmenge $\delta(X_1, \dots, X_p) := \delta(X_1) \cup \dots \cup \delta(X_p)$, wobei $V(G) = X_1 \dot{\cup} X_2 \dot{\cup} \dots \dot{\cup} X_p$ eine Partition der Knotenmenge in nichtleere Teilmengen ist. Für $p = 3$ sprechen wir von 3-Schnitten. Beachte, dass Multischnitte mit $p = 2$ Schnitte sind.

Satz 6.17. (Tutte [1961], Nash-Williams [1961]) *Ein ungerichteter Graph G enthält k paarweise kantendisjunkte aufspannende Bäume genau dann, wenn*

$$|\delta(X_1, \dots, X_p)| \geq k(p - 1)$$

für jeden Multischnitt $\delta(X_1, \dots, X_p)$.

Beweis: Um die Notwendigkeit zu beweisen, seien T_1, \dots, T_k paarweise kantendisjunkte aufspannende Bäume in G und $\delta(X_1, \dots, X_p)$ ein Multischnitt. Durch Kontraktion jeder der Knotenmengen X_1, \dots, X_p erhalten wir einen Graphen G' , dessen Knoten X_1, \dots, X_p sind und dessen Kanten den Kanten des Multischnitts entsprechen. Die T_1, \dots, T_k entsprechen paarweise kantendisjunkten zusammenhängenden Teilgraphen T'_1, \dots, T'_k in G' . Jeder der Teilgraphen T'_1, \dots, T'_k hat mindestens $p - 1$ Kanten, somit hat G' (also auch der Multischnitt) mindestens $k(p - 1)$ Kanten.

Nun beweisen wir mittels Induktion über $|V(G)|$, dass die Bedingung auch hinreichend ist. Für $n := |V(G)| \leq 2$ ist die Aussage klar. Sei nun $n > 2$ und angenommen, $|\delta(X_1, \dots, X_p)| \geq k(p - 1)$ für jeden Multischnitt $\delta(X_1, \dots, X_p)$. Insbesondere (betrachte die Partition in einelementige Knotenmengen) hat G mindestens $k(n - 1)$ Kanten. Ferner bleibt diese Eigenschaft bei der Kontraktion von Knotenmengen erhalten, also folgt mit der Induktionsvoraussetzung: G/X enthält k paarweise kantendisjunkte aufspannende Bäume für jedes $X \subset V(G)$ mit $|X| \geq 2$.

Sei $F = (F_1, \dots, F_k)$ ein k -Tupel paarweise kantendisjunkter Wälder in G mit $|E(F)|$ maximal, wobei wiederum $E(F) := \bigcup_{i=1}^k E(F_i)$. Ist jedes F_i ein aufspannender Baum, so sind wir fertig. Falls nicht, so ist $|E(F)| < k(n - 1)$, also gibt es eine Kante $e \in E(G) \setminus E(F)$. Nach Lemma 6.16 gibt es dann eine Knotenmenge $X \subseteq V(G)$ mit $e \subseteq X$, so dass $F_i[X]$ für jedes i zusammenhängend ist. Aus $|X| \geq 2$ folgt: G/X enthält k paarweise kantendisjunkte aufspannende Bäume F'_1, \dots, F'_k . Dann bildet F'_i zusammen mit $F_i[X]$ für jedes i einen aufspannenden Baum in G und diese k aufspannenden Bäume sind paarweise kantendisjunkt. \square

Nun wenden wir uns dem entsprechenden Problem für Digraphen, d. h. dem Packen von Arboreszenzen, zu:

Satz 6.18. (Edmonds [1973]) *Sei G ein Digraph und $r \in V(G)$. Dann ist die maximale Anzahl von paarweise kantendisjunkten aufspannenden Arboreszenzen mit Wurzel r gleich der minimalen Kardinalität eines r -Schnitts.*

Beweis: Sei k die minimale Kardinalität eines r -Schnitts. Offensichtlich gibt es höchstens k paarweise kantendisjunkte aufspannende Arboreszenzen mit Wurzel r .

Wir werden mittels Induktion über k beweisen, dass es k paarweise kantendisjunkte aufspannende Arboreszenzen mit Wurzel r gibt. Der Fall $k = 0$ ist trivial.

Haben wir eine aufspannende Arboreszenz A mit Wurzel r und mit

$$\min_{r \in S \subset V(G)} |\delta_G^+(S) \setminus E(A)| \geq k - 1 \quad (6.2)$$

gefunden, so sind wir mittels Induktion fertig. Angenommen, wir haben bereits eine (nicht notwendigerweise aufspannende) Arboreszenz A mit Wurzel r gefunden, für die (6.2) gilt. Sei $R \subseteq V(G)$ die von A überdeckte Knotenmenge. Am Anfang ist $R = \{r\}$; ist $R = V(G)$, so sind wir fertig.

Ist $R \neq V(G)$, so nennen wir eine Knotenmenge $X \subseteq V(G)$ kritisch, falls

- (a) $r \in X$;
- (b) $X \cup R \neq V(G)$;
- (c) $|\delta_G^+(X) \setminus E(A)| = k - 1$.

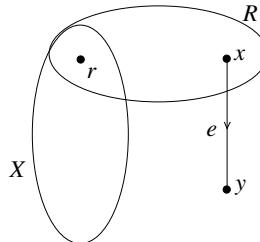


Abbildung 6.4.

Gibt es keine kritischen Knotenmengen, so können wir irgendeine in R beginnende Kante zu A hinzufügen. Andernfalls sei X eine inklusionsmaximale kritische Knotenmenge und $e = (x, y)$ eine Kante mit $x \in R \setminus X$ und $y \in V(G) \setminus (R \cup X)$ (siehe Abb. 6.4). Es gibt eine solche Kante, da

$$|\delta_{G-E(A)}^+(R \cup X)| = |\delta_G^+(R \cup X)| \geq k > k - 1 = |\delta_{G-E(A)}^+(X)|.$$

Nun fügen wir e zu A hinzu. Offensichtlich ist $A + e$ eine Arboreszenz mit Wurzel r . Wir müssen zeigen, dass (6.2) nach wie vor gilt.

Angenommen, es gäbe ein Y mit $r \in Y \subset V(G)$ und $|\delta_G^+(Y) \setminus E(A+e)| < k - 1$. Dann folgt $x \in Y$, $y \notin Y$ und $|\delta_G^+(Y) \setminus E(A)| = k - 1$. Mit Lemma 2.1(a) haben wir

$$\begin{aligned} k - 1 + k - 1 &= |\delta_{G-E(A)}^+(X)| + |\delta_{G-E(A)}^+(Y)| \\ &\geq |\delta_{G-E(A)}^+(X \cup Y)| + |\delta_{G-E(A)}^+(X \cap Y)| \\ &\geq k - 1 + k - 1, \end{aligned}$$

da $r \in X \cap Y$ und $y \in V(G) \setminus (X \cup Y)$. Also werden die beiden obigen Ungleichungen mit Gleichheit erfüllt, insbesondere gilt $|\delta_{G-E(A)}^+(X \cup Y)| = k - 1$. Da $y \in V(G) \setminus (X \cup Y \cup R)$, folgt, dass $X \cup Y$ kritisch ist. Da aber $x \in Y \setminus X$, widerspricht dies der Maximalität von X . \square

Dieser Beweis stammt von Lovász [1976]. Fujishige [2010] hat dieses Resultat auf das Packen von Arboreszenzen verallgemeinert, die gegebene (aber nicht notwendigerweise identische) Wurzeln haben und die für gewisse gegebene Teilmengen der Knotenmenge aufspannend sind. Eine gemeinsame Verallgemeinerung der Sätze 6.17 und 6.18 wurde von Frank [1981] gefunden. Eine gute Charakterisierung (siehe Kapitel 15 zur Erklärung dieses Begriffs) des Packproblems von aufspannenden Arboreszenzen mit beliebigen Wurzeln wird in folgendem Satz beschrieben, den wir hier nicht beweisen:

Satz 6.19. (Frank [1979]) *Ein Digraph G enthält k paarweise kantendisjunkte aufspannende Arboreszenzen genau dann, wenn*

$$\sum_{i=1}^p |\delta^-(X_i)| \geq k(p-1)$$

für jede Familie paarweise disjunkter nichtleerer Teilmengen $X_1, \dots, X_p \subseteq V(G)$.

Eine weitere relevante Frage ist, wie viele Wälder man zur Überdeckung eines Graphen benötigt. Die Antwort ist in folgendem Satz enthalten:

Satz 6.20. (Nash-Williams [1964]) *Die Kantenmenge eines ungerichteten Graphen G ist genau dann die Vereinigung von k Wäldern, wenn $|E(G[X])| \leq k(|X| - 1)$ für alle $\emptyset \neq X \subseteq V(G)$.*

Beweis: Die Notwendigkeit ist klar: Bei einer Gesamtknotenmenge X kann kein Wald mehr als $|X| - 1$ Kanten enthalten. Um zu beweisen, dass die Bedingung hinreichend ist, sei $F = (F_1, \dots, F_k)$ ein k -Tupel paarweise kantendisjunkter Wälder in G mit $|E(F)| = \left| \bigcup_{i=1}^k E(F_i) \right|$ maximal. Wir behaupten, dass $E(F) = E(G)$. Angenommen, es gäbe eine Kante $e \in E(G) \setminus E(F)$. Dann gibt es nach Lemma 6.16 eine Knotenmenge $X \subseteq V(G)$ mit $e \subseteq X$, so dass $F_i[X]$ für jedes i zusammenhängend ist. Insbesondere gilt

$$|E(G[X])| \geq \left| \{e\} \dot{\cup} \bigcup_{i=1}^k E(F_i[X]) \right| = 1 + k(|X| - 1),$$

im Widerspruch zur Annahme. \square

Aufgabe 29 enthält eine gerichtete Version. Verallgemeinerungen der Sätze 6.17 und 6.20 auf Matroide werden in Aufgabe 20, Kapitel 13, gegeben.

Aufgaben

1. Man beweise den Satz von Cayley 6.2 indem man zeigt, dass die folgende rekursive Definition eine Bijektion zwischen der Menge der aufspannenden Bäume in K_n und der Menge der Vektoren in $\{1, \dots, n\}^{n-2}$ herstellt: Für einen Baum T mit $V(T) = \{1, \dots, n\}$, $n \geq 3$, sei v das Blatt mit dem kleinsten Index und a_1 der Nachbar von v . Man setze rekursiv $a(T) := (a_1, \dots, a_{n-2})$, wobei $(a_2, \dots, a_{n-2}) = a(T - v)$.
Prüfer [1918]
2. Man beweise, dass es genau $(n+1)^{n-1}$ Branchings B mit $V(B) = \{1, \dots, n\}$ gibt.
3. Sei p_n die Wahrscheinlichkeit dass Knoten 1 ein Blatt in T ist, wobei T aus allen Bäumen mit Knotenmenge $\{1, \dots, n\}$ zufällig gewählt wird (mit gleichmäßiger Verteilung). Was ist $\lim_{n \rightarrow \infty} p_n$?
4. Seien (V, T_1) und (V, T_2) zwei Bäume mit derselben Knotenmenge V . Man beweise, dass es für jede Kante $e \in T_1$ eine Kante $f \in T_2$ gibt, so dass sowohl $(V, (T_1 \setminus \{e\}) \cup \{f\})$ als auch $(V, (T_2 \setminus \{f\}) \cup \{e\})$ ein Baum ist.
5. Sei (G, c) eine Instanz des MINIMUM-SPANNING-TREE-PROBLEMS, wobei G zusammenhängend ist und $c(e) \neq c(e')$ für je zwei verschiedene Kanten e und e' . Man beweise, dass es dann genau eine optimale Lösung gibt.
6. Für einen gegebenen ungerichteten Graphen G mit den Gewichten $c : E(G) \rightarrow \mathbb{R}$ und einem Knoten $v \in V(G)$ soll ein aufspannender Baum minimalen Gewichtes in G bestimmt werden, für den v kein Blatt ist. Kann man dieses Problem in polynomieller Zeit lösen?
7. Man möchte die Menge derjenigen Kanten e in einem ungerichteten Graphen G mit den Gewichten $c : E(G) \rightarrow \mathbb{R}$ bestimmen, für welche es einen e enthaltenden aufspannenden Baum minimalen Gewichtes in G gibt (oder anders ausgedrückt, man möchte die Vereinigung aller aufspannenden Bäume minimalen Gewichtes in G bestimmen). Man zeige, wie dieses Problem mit $O(mn)$ -Laufzeit gelöst werden kann.
8. Für einen gegebenen ungerichteten Graphen G mit beliebigen Gewichten $c : E(G) \rightarrow \mathbb{R}$ soll ein zusammenhängender aufspannender Teilgraph minimalen Gewichtes bestimmt werden. Kann man dieses Problem effizient lösen?
9. Man betrachte den folgenden Algorithmus (gelegentlich WORST-OUT-GREEDY-ALGORITHMUS genannt, siehe Abschnitt 13.4). Man untersuche die Kanten in der Reihenfolge nichtsteigender Gewichte und entferne jede Kante, die keine Brücke ist. Kann dieser Algorithmus zur Lösung des MINIMUM-SPANNING-TREE-PROBLEMS herangezogen werden?
10. Man betrachte den folgenden Färbungsalgorithmus. Zu Beginn sind alle Kanten ungefärbt. Man wende die zwei folgenden Schritte in beliebiger Reihenfolge an, bis alle Kanten gefärbt sind.
Blaue Regel: Man wähle einen Schnitt ohne blaue Kanten. Nun wähle man eine ungefärbte Kante minimalen Gewichtes in diesem Schnitt und färbe sie blau.
Rote Regel: Man wähle einen Kreis ohne rote Kanten. Nun wähle man eine

ungefärbte Kante maximalen Gewichtes in diesem Kreis und färbe sie rot.
 Man zeige, dass immer eine der beiden Regeln angewendet werden kann, solange es noch ungefärbte Kanten gibt (siehe Lemma 2.6). Man zeige ferner, dass die „Färbungsvariante“ während des Algorithmus erhalten bleibt: Es gibt immer einen optimalen aufspannenden Baum, welcher alle blauen aber gar keine roten Kanten enthält. (Also löst dieser Algorithmus das MINIMUM-SPANNING-TREE-PROBLEM optimal.) Man beachte, dass sowohl KRUSKALS ALGORITHMUS als auch PRIMS ALGORITHMUS Spezialfälle sind.

(Tarjan [1983])

11. Angenommen, man möchte in einem ungerichteten Graphen einen aufspannenden Baum T finden, mit der Eigenschaft, dass das Gewicht der schwersten Kante in T so gering wie möglich ist. Wie kann man dies bewerkstelligen?
12. Stimmt es, dass die maximale Länge eines Weges in einem Branching das einen Fibonacci-Heap implementiert $O(\log n)$ ist, wobei n die Anzahl der Elemente ist?
13. Man beweise, dass zwei Fibonacci-Heaps mit n_1 und n_2 Elementen in $O(\log(n_1 + n_2))$ Zeit zusammengefügt ("merged") werden können. Der so entstehende Fibonacci-Heap soll alle $n_1 + n_2$ Elemente enthalten.
14. Das Voronoï-Diagramm einer endlichen Menge $V \subset \mathbb{R}^2$ besteht aus den Gebieten

$$P_v := \left\{ x \in \mathbb{R}^2 : \|x - v\|_2 = \min_{w \in V} \|x - w\|_2 \right\}$$

für alle $v \in V$. Die Delaunay-Triangulierung von V ist der Graph

$$(V, \{\{v, w\} \subseteq V, v \neq w, |P_v \cap P_w| > 1\}).$$

Ein aufspannender Baum mit minimalem Gewicht für V ist ein Baum T mit $V(T) = V$ und minimaler Länge $\sum_{\{v,w\} \in E(T)} \|v - w\|_2$. Man beweise, dass jeder aufspannende Baum mit minimalem Gewicht ein Teilgraph der Delaunay-Triangulierung ist.

Bemerkung: Aus der Tatsache, dass die Delaunay-Triangulierung mit $O(n \log n)$ -Laufzeit berechnet werden kann (wobei $n = |V|$; siehe z. B. Fortune [1987], Knuth [1992]), folgt, dass es einen $O(n \log n)$ Algorithmus für das MINIMUM-SPANNING-TREE-PROBLEM für Punktmengen in der Ebene gibt.

(Shamos und Hoey [1975]; siehe auch Zhou, Shenoy und Nicholls [2002])

15. Kann man in linearer Zeit entscheiden, ob ein Digraph eine aufspannende Arboreszenz enthält?

Hinweis: Um eine mögliche Wurzel zu bestimmen, fange man bei einem beliebigen Knoten an und gehe so lange wie möglich Kanten rückwärts entlang. Erreicht man einen Kreis, so kontrahiere man ihn.

16. Kann man in einem gegebenen Digraphen ein Branching maximaler Kardinalität in linearer Zeit finden?

Hinweis: Zunächst bestimme man die starken Zusammenhangskomponenten.

17. Das MINIMUM-WEIGHT-ROOTED-ARBORESCENCE-PROBLEM kann mittels Proposition 6.9 auf das MAXIMUM-WEIGHT-BRANCHING-PROBLEM redu-

ziert werden. Es kann aber auch mittels einer modifizierten Version des EDMONDS' BRANCHING-ALGORITHMUS direkt gelöst werden. Man zeige dies.

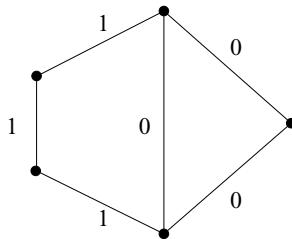


Abbildung 6.5.

18. Man beweise, dass das Baumpolytop eines ungerichteten Graphen G mit $n := |V(G)|$ (siehe Satz 6.13) im Allgemeinen eine echte Teilmenge des folgenden Polytops ist:

$$\left\{ x \in [0, 1]^{E(G)} : \sum_{e \in E(G)} x_e = n - 1, \sum_{e \in \delta(X)} x_e \geq 1 \text{ für } \emptyset \subset X \subset V(G) \right\}.$$

Hinweis: Um zu beweisen, dass dieses Polytop nicht ganzzahlig ist, betrachte man den Graphen in Abb. 6.5 (die Zahlen sind die Kantengewichte).
(Magnanti und Wolsey [1995])

19. Aus Aufgabe 18 folgt, dass Schnittbedingungen für die Beschreibung des Baumpolytops nicht ausreichen. Betrachten wir jedoch stattdessen Multischnitte, so gelingt dies: Man beweise, dass das Baumpolytop eines ungerichteten Graphen G mit $n := |V(G)|$ aus allen Vektoren $x \in [0, 1]^{E(G)}$ mit

$$\sum_{e \in E(G)} x_e = n - 1 \text{ und } \sum_{e \in C} x_e \geq k - 1 \text{ für alle Multischnitte } C = \delta(X_1, \dots, X_k)$$

besteht. (Magnanti und Wolsey [1995])

- * 20. Sei G ein ungerichteter Graph und $n := |V(G)|$. Man beweise, dass das folgende lineare Ungleichungssystem mit $O(n^3)$ Variablen und Ungleichungen ein Polytop beschreibt, dessen Orthogonalprojektion auf die x -Variablen das Baumpolytop von G ist: $x_e \geq 0$ ($e \in E(G)$), $z_{u,v,w} \geq 0$ ($\{u, v\} \in E(G)$, $w \in V(G) \setminus \{u, v\}$), $\sum_{e \in E(G)} x_e = n - 1$, $x_e = z_{u,v,w} + z_{v,u,w}$ ($e = \{u, v\} \in E(G)$, $w \in V(G) \setminus \{u, v\}$), und $x_e + \sum_{\{u,v\} \in \delta(v) \setminus \{e\}} z_{u,v,w} = 1$ ($v \in V(G)$, $e = \{v, w\} \in \delta(v)$).

Bemerkung: In Conforti, Cornuéjols und Zambelli [2010] geben diese Autoren einen Überblick über solche erweiterten Formulierungen von kombinatorischen LPs.

21. Man beweise, dass die konvexe Hülle der Inzidenzvektoren aller Wälder in einem ungerichteten Graphen G das folgende Polytop (Wald-Polytop) ist:

$$P := \left\{ x \in [0, 1]^{E(G)} : \sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ für } \emptyset \neq X \subseteq V(G) \right\}.$$

Bemerkung: Aus diesem Resultat folgt Satz 6.13, da $\sum_{e \in E(G[X])} x_e = |V(G)| - 1$ eine Stützhyperebene ist. Ferner ist es ein Spezialfall von Satz 13.21.

- * 22. Man beweise, dass die konvexe Hülle der Inzidenzvektoren aller Branchings in einem Digraphen G die Menge aller Vektoren $x \in [0, 1]^{E(G)}$ mit

$$\sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ für } \emptyset \neq X \subseteq V(G) \text{ und } \sum_{e \in \delta^-(v)} x_e \leq 1 \text{ für } v \in V(G)$$

ist.

Bemerkung: Dieses Resultat ist ein Spezialfall von Satz 14.13.

- * 23. Sei G ein Digraph und $r \in V(G)$. Man beweise, dass die Polytope

$$\left\{ x \in [0, 1]^{E(G)} : x_e = 0 \text{ (} e \in \delta^-(r) \text{)}, \sum_{e \in \delta^-(v)} x_e = 1 \text{ (} v \in V(G) \setminus \{r\} \text{)}, \sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ für } \emptyset \neq X \subseteq V(G) \right\}$$

und

$$\left\{ x \in [0, 1]^{E(G)} : x_e = 0 \text{ (} e \in \delta^-(r) \text{)}, \sum_{e \in \delta^-(v)} x_e = 1 \text{ (} v \in V(G) \setminus \{r\} \text{)}, \sum_{e \in \delta^+(X)} x_e \geq 1 \text{ für } r \in X \subset V(G) \right\}$$

beide die konvexe Hülle der Inzidenzvektoren aller aufspannenden Arboreszenzen mit Wurzel r sind.

- 24. Man beweise, dass jeder $2k$ -fach kantenzusammenhängende Graph k paarweise kantendisjunkte aufspannende Bäume enthält.
 - 25. Sei G ein Digraph und $r \in V(G)$. Man beweise, dass G genau dann die disjunkte Vereinigung von k aufspannenden Arboreszenzen mit Wurzel r ist, wenn der zugrunde liegende ungerichtete Graph die disjunkte Vereinigung von k aufspannenden Bäumen ist und $|\delta^-(x)| = k$ für alle $x \in V(G) \setminus \{r\}$ gilt. (Edmonds)
 - 26. Sei G ein Digraph und $r \in V(G)$. Angenommen, G enthält k paarweise kantendisjunkte Wege von r zu jedem anderen Knoten, dass aber das Entfernen einer beliebigen Kante diese Eigenschaft zerstört. Man beweise, dass jeder Knoten von G außer r genau k ankommende Kanten hat.
- Hinweis:* Man benutze Satz 6.18.
- * 27. Man beweise die Aussage von Aufgabe 26 ohne Satz 6.18 zu gebrauchen. Ferner formuliere und beweise man eine knotendisjunkte Version.

Hinweis: Hat ein Knoten v mehr als k ankommende Kanten, so nehme man k paarweise kantendisjunkte r - v -Wege. Man zeige, dass eine in v ankommende Kante, die auf keinem dieser Wege liegt, entfernt werden kann.

28. Man beschreibe einen polynomiellen Algorithmus zur Bestimmung einer kardinalitätsmaximalen Menge paarweise kantendisjunkter aufspannender Arboreszenzen (mit Wurzel r) in einem Digraphen G .

Bemerkung: Der effizienteste Algorithmus für dieses Problem stammt von Gabow [1995]; siehe auch (Gabow und Manu [1998]).

29. Man beweise, dass die Kanten eines Digraphen G genau dann durch k Branchings überdeckt werden können, wenn die folgenden beiden Bedingungen gelten:

- (a) $|\delta^-(v)| \leq k$ für alle $v \in V(G)$;
- (b) $|E(G[X])| \leq k(|X| - 1)$ für alle $X \subseteq V(G)$.

Hinweis: Man benutze Satz 6.18.

(Frank [1979])

Literatur

Allgemeine Literatur:

- Ahuja, R.K., Magnanti, T.L., und Orlin, J.B. [1993]: Network Flows. Prentice-Hall, Englewood Cliffs 1993, Kapitel 13
- Balakrishnan, V.K. [1995]: Network Optimization. Chapman and Hall, London 1995, Kapitel 1
- Cormen, T.H., Leiserson, C.E., Rivest, R.L., und Stein, C. [2009]: Introduction to Algorithms. 3. Aufl. MIT Press, Cambridge 2009, Kapitel 23
- Gondran, M., und Minoux, M. [1984]: Graphs and Algorithms. Wiley, Chichester 1984, Kapitel 4
- Magnanti, T.L., und Wolsey, L.A. [1995]: Optimal trees. In: Handbooks in Operations Research and Management Science; Volume 7: Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, Hrsg.), Elsevier, Amsterdam 1995, pp. 503–616
- Schrijver, A. [2003]: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin 2003, Chapters 50–53
- Tarjan, R.E. [1983]: Data Structures and Network Algorithms. SIAM, Philadelphia 1983, Kapitel 6
- Wu, B.Y., und Chao, K.-M. [2004]: Spanning Trees and Optimization Problems. Chapman & Hall/CRC, Boca Raton 2004

Zitierte Literatur:

- Bock, F.C. [1971]: An algorithm to construct a minimum directed spanning tree in a directed network. In: Avi-Itzhak, B. (Hrsg.): Developments in Operations Research, Volume 1. Gordon and Breach, New York 1971, pp. 29–44
- Borůvka, O. [1926a]: O jistém problému minimálním. Práca Moravské Přírodovědecké Společnosti 3 (1926), 37–58 [auf Tschechisch]
- Borůvka, O. [1926b]: Příspěvěk k řešení otázky ekonomické stavby. Elektrovodních sítí. Elektrotechnicky Obzor 15 (1926), 153–154 [auf Tschechisch]

- Cayley, A. [1889]: A theorem on trees. *Quarterly Journal on Mathematics* 23 (1889), 376–378
- Chazelle, B. [2000]: A minimum spanning tree algorithm with inverse-Ackermann type complexity. *Journal of the ACM* 47 (2000), 1028–1047
- Cheriton, D., und Tarjan, R.E. [1976]: Finding minimum spanning trees. *SIAM Journal on Computing* 5 (1976), 724–742
- Chu, Y., und Liu, T. [1965]: On the shortest arborescence of a directed graph. *Scientia Sinica* 4 (1965), 1396–1400; *Mathematical Review* 33, # 1245
- Conforti, M., Cornuéjols, G., und Zambelli, G. [2009]: Extended formulations in combinatorial optimization. *4OR* 8 (2010), 1–48
- Diestel, R. [1996]: *Graphentheorie*. Springer, Berlin 1996
- Dijkstra, E.W. [1959]: A note on two problems in connexion with graphs. *Numerische Mathematik* 1 (1959), 269–271
- Dixon, B., Rauch, M., und Tarjan, R.E. [1992]: Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM Journal on Computing* 21 (1992), 1184–1192
- Edmonds, J. [1967]: Optimum branchings. *Journal of Research of the National Bureau of Standards B* 71 (1967), 233–240
- Edmonds, J. [1970]: Submodular functions, matroids and certain polyhedra. In: *Combinatorial Structures and Their Applications; Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications 1969* (R. Guy, H. Hanani, N. Sauer, J. Schönheim, Hrsg.), Gordon and Breach, New York 1970, pp. 69–87
- Edmonds, J. [1973]: Edge-disjoint branchings. In: *Combinatorial Algorithms* (R. Rustin, Hrsg.), Algorithmic Press, New York 1973, pp. 91–96
- Fortune, S. [1987]: A sweepline algorithm for Voronoi diagrams. *Algorithmica* 2 (1987), 153–174
- Frank, A. [1981]: On disjoint trees and arborescences. In: *Algebraic Methods in Graph Theory; Colloquia Mathematica Societatis János Bolyai* 25 (L. Lovász, V.T. Sós, Hrsg.), North-Holland, Amsterdam 1981, pp. 159–169
- Frank, A. [1979]: Covering branchings. *Acta Scientiarum Mathematicarum (Szeged)* 41 (1979), 77–82
- Fredman, M.L., und Tarjan, R.E. [1987]: Fibonacci heaps and their uses in improved network optimization problems. *Journal of the ACM* 34 (1987), 596–615
- Fredman, M.L., und Willard, D.E. [1994]: Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *Journal of Computer and System Sciences* 48 (1994), 533–551
- Fujishige, S. [2010]: A note on disjoint arborescences. *Combinatorica* 30 (2010), 247–252
- Fulkerson, D.R. [1974]: Packing rooted directed cuts in a weighted directed graph. *Mathematical Programming* 6 (1974), 1–13
- Gabow, H.N. [1995]: A matroid approach to finding edge connectivity and packing arborescences. *Journal of Computer and System Sciences* 50 (1995), 259–273
- Gabow, H.N., Galil, Z., und Spencer, T. [1989]: Efficient implementation of graph algorithms using contraction. *Journal of the ACM* 36 (1989), 540–572
- Gabow, H.N., Galil, Z., Spencer, T., und Tarjan, R.E. [1986]: Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica* 6 (1986), 109–122
- Gabow, H.N., und Manu, K.S. [1998]: Packing algorithms for arborescences (and spanning trees) in capacitated graphs. *Mathematical Programming B* 82 (1998), 83–109

- Jarník, V. [1930]: O jistém problému minimálním. Práca Moravské Přírodovědecké Společnosti 6 (1930), 57–63
- Karger, D., Klein, P.N., und Tarjan, R.E. [1995]: A randomized linear-time algorithm to find minimum spanning trees. Journal of the ACM 42 (1995), 321–328
- Karp, R.M. [1972]: A simple derivation of Edmonds' algorithm for optimum branchings. Networks 1 (1972), 265–272
- King, V. [1997]: A simpler minimum spanning tree verification algorithm. Algorithmica 18 (1997), 263–270
- Knuth, D.E. [1992]: Axioms and hulls; LNCS 606. Springer, Berlin 1992
- Korte, B., und Nešetřil, J. [2001]: Vojtěch Jarník's work in combinatorial optimization. Discrete Mathematics 235 (2001), 1–17
- Kruskal, J.B. [1956]: On the shortest spanning subtree of a graph and the travelling salesman problem. Proceedings of the AMS 7 (1956), 48–50
- Lovász, L. [1976]: On two minimax theorems in graph. Journal of Combinatorial Theory B 21 (1976), 96–103
- Nash-Williams, C.S.J.A. [1961]: Edge-disjoint spanning trees of finite graphs. Journal of the London Mathematical Society 36 (1961), 445–450
- Nash-Williams, C.S.J.A. [1964]: Decompositions of finite graphs into forests. Journal of the London Mathematical Society 39 (1964), 12
- Nešetřil, J., Milková, E., und Nešetřilová, H. [2001]: Otakar Borůvka on minimum spanning tree problem. Translation of both the 1926 papers, comments, history. Discrete Mathematics 233 (2001), 3–36
- Pettie, S., und Ramachandran, V. [2002]: An optimal minimum spanning tree algorithm. Journal of the ACM 49 (2002), 16–34
- Prim, R.C. [1957]: Shortest connection networks and some generalizations. Bell System Technical Journal 36 (1957), 1389–1401
- Prüfer, H. [1918]: Neuer Beweis eines Satzes über Permutationen. Arch. Math. Phys. 27 (1918), 742–744
- Shamos, M.I., und Hoey, D. [1975]: Closest-point problems. Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science (1975), 151–162
- Sylvester, J.J. [1857]: On the change of systems of independent variables. Quarterly Journal of Mathematics 1 (1857), 42–56
- Tarjan, R.E. [1975]: Efficiency of a good but not linear set union algorithm. Journal of the ACM 22 (1975), 215–225
- Tutte, W.T. [1961]: On the problem of decomposing a graph into n connected factors. Journal of the London Mathematical Society 36 (1961), 221–230
- Zhou, H., Shenoy, N., und Nicholls, W. [2002]: Efficient minimum spanning tree construction without Delaunay triangulation. Information Processing Letters 81 (2002), 271–276



7 Kürzeste Wege

Eines der bekanntesten kombinatorischen Optimierungsprobleme ist, einen kürzesten Weg zwischen zwei bestimmten Knoten eines gerichteten oder ungerichteten Graphen zu finden:

KÜRZESTE-WEGE-PROBLEM

Instanz: Ein Graph G (gerichtet oder ungerichtet), Gewichte $c : E(G) \rightarrow \mathbb{R}$ und zwei Knoten $s, t \in V(G)$.

Aufgabe: Finde einen kürzesten s - t -Weg P , d. h. einen mit minimalem Gewicht $c(E(P))$, oder entscheide, dass t von s aus nicht erreichbar ist.

Offensichtlich hat dieses Problem viele praktische Anwendungen. Wie das MINIMUM-SPANNING-TREE-PROBLEM, kommt es oft als Teilproblem eines viel schwierigeren kombinatorischen Optimierungsproblems vor.

Erlauben wir beliebige Gewichte, so ist dieses Problem aber nicht einfach zu lösen. Sind z.B. alle Gewichte gleich -1 , so sind die s - t -Wege mit Gewicht $1 - |V(G)|$ genau die hamiltonschen s - t -Wege, und zu entscheiden, ob es einen solchen überhaupt gibt, ist ein schwieriges Problem (siehe Aufgabe 17(b), Kapitel 15).

Das Problem ist aber sehr viel einfacher zu lösen, wenn wir nur nichtnegative Gewichte zulassen, oder aber wenigstens Kreise negativen Gewichtes ausschließen:

Definition 7.1. Sei G ein (gerichteter oder ungerichteter) Graph mit Gewichten $c : E(G) \rightarrow \mathbb{R}$. Dann heißt c **konservativ** wenn es keinen Kreis mit negativem Gewicht gibt.

In Abschnitt 7.1 werden wir Algorithmen zur Lösung des KÜRZESTE-WEGE-PROBLEMS in Digraphen angeben. Der erste Algorithmus funktioniert nur für nichtnegative Gewichte, während der zweite für beliebige konservative Gewichte geeignet ist.

Beide Algorithmen in Abschnitt 7.1 berechnen in der Tat einen kürzesten s - v -Weg für alle $v \in V(G)$ ohne bedeutend mehr Laufzeit zu benötigen. Öfters möchte man einen kürzesten Weg für jedes Knotenpaar berechnen; in Abschnitt 7.2 werden wir sehen, wie man dieses Problem bewältigt.

Da uns Kreise negativen Gewichtes Probleme bereiten, werden wir auch zeigen, wie man diese aufspürt. Gibt es keine, so ist es recht einfach, einen Kreis minimalen

Gewichtes zu finden. Ein weiteres interessantes Problem ist die Bestimmung eines Kreises mit minimalem durchschnittlichem Kantengewicht. Wie wir in Abschnitt 7.3 sehen werden, kann die gerichtete Version dieses Problems ebenfalls effizient gelöst werden.

In Abschnitt 7.4 werden wir zeigen, dass ungerichtete Graphen mit nichtnegativen Gewichten stets Bäume enthalten, die nicht sehr viel teurer als aufspannende Bäume minimalen Gewichtes sind, aber auch kurze Wege von einer Quelle s zu allen anderen Knoten enthalten.

Kürzeste Wege in ungerichteten Graphen zu finden ist schwieriger, außer wenn die Kantengewichte nichtnegativ sind. Ungerichtete nichtnegativ gewichtete Kanten kann man durch ein Paar entgegengesetzter gerichteter und gleichgewichteter Kanten ersetzen; dies transformiert ein ungerichtetes Problem in ein äquivalentes gerichtetes. Dieser Trick funktioniert jedoch nicht für negativ gewichtete Kanten, weil dann Kreise negativen Gewichtes entstehen. Wir werden uns später in Abschnitt 12.2 (Korollar 12.13) mit dem Problem der Bestimmung kürzester Wege in ungerichteten Graphen mit konservativen Gewichten befassen.

Von nun an setzen wir einen Digraphen G voraus. O. B. d. A. können wir annehmen, dass G zusammenhängend und einfach ist; bei parallelen Kanten brauchen wir nur die leichteste zu berücksichtigen.

7.1 Kürzeste Wege von einer Quelle aus

Alle Kürzeste-Wege-Algorithmen in diesem Buch basieren auf dem folgenden Resultat, auch Bellmans Optimalitätsprinzip genannt, welches in der Tat den Kern der Dynamischen Optimierung beinhaltet:

Proposition 7.2. *Sei G ein Digraph mit konservativen Gewichten $c : E(G) \rightarrow \mathbb{R}$, seien s und w zwei Knoten und $k \in \mathbb{N}$. Sei ferner P ein kürzester Weg unter allen $s-w$ -Wegen mit höchstens k Kanten und sei $e = (v, w)$ die letzte Kante von P . Dann ist $P_{[s,v]}$ (d. h. P ohne Kante e) ein kürzester Weg unter allen $s-v$ -Wegen mit höchstens $k - 1$ Kanten.*

Beweis: Angenommen, es gäbe einen $s-v$ -Weg Q mit $|E(Q)| \leq k - 1$, der kürzer als $P_{[s,v]}$ ist. Dann gilt $c(E(Q)) + c(e) < c(E(P))$. Ist w nicht in Q enthalten, dann ist $Q + e$ ein kürzerer $s-w$ -Weg als P . Ist andererseits w in Q enthalten, so hat $Q_{[s,w]}$ die Länge $c(E(Q_{[s,w]})) = c(E(Q)) + c(e) - c(E(Q_{[w,v]} + e)) < c(E(P)) - c(E(Q_{[w,v]} + e)) \leq c(E(P))$, da $Q_{[w,v]} + e$ ein Kreis und c konservativ ist. In beiden Fällen haben wir somit einen Widerspruch zu der Annahme, dass P ein kürzester $s-w$ -Weg mit höchstens k Kanten ist. \square

Ein entsprechendes Resultat gilt für ungerichtete Graphen mit nichtnegativen Gewichten und auch für azyklische Digraphen mit beliebigen Gewichten. Wir bekommen die Rekursionsformeln $\text{dist}(s, s) = 0$ und $\text{dist}(s, w) = \min\{\text{dist}(s, v) + c((v, w)) : (v, w) \in E(G)\}$ für $w \in V(G) \setminus \{s\}$, welche sofort eine Lösung für das KÜRZESTE-WEGE-PROBLEM in azyklischen Digraphen ermöglichen (Aufgabe 7).

Proposition 7.2 liefert auch den Grund dafür, dass die meisten Algorithmen kürzeste Wege von s zu allen anderen Knoten berechnen. Bei der Berechnung eines kürzesten s - t -Weges P hat man bereits einen kürzesten s - v -Weg für jeden Knoten v auf P berechnet. Da man nicht im Voraus weiß, welche Knoten auf P liegen, ist es ganz natürlich, dass man kürzeste s - v -Wege für alle v berechnet. Diese s - v -Wege können wir sehr effizient speichern, indem wir nur die jeweils letzte Kante speichern.

Zunächst setzen wir nichtnegative Gewichte voraus, d.h. $c : E(G) \rightarrow \mathbb{R}_+$. Sind alle Gewichte gleich 1, so kann das KÜRZESTE-WEGE-PROBLEM mittels BFS gelöst werden (Proposition 2.18). Sind alle Gewichte natürliche Zahlen, d.h. $c : E(G) \rightarrow \mathbb{N}$, so könnte man jede Kante e durch einen Weg der Länge $c(e)$ ersetzen und wiederum BFS benutzen. Dies könnte jedoch zu einer exponentiellen Anzahl von Kanten führen; beachte, dass die Inputgröße

$\Theta(n \log m + m \log n + \sum_{e \in E(G)} \log c(e))$ ist, wobei $n = |V(G)|$ und $m = |E(G)|$.

Eine viel bessere Idee ist es, den folgenden von Dijkstra [1959] stammenden Algorithmus zu benutzen. Er hat große Ähnlichkeit mit PRIMS ALGORITHMUS für das MINIMUM-SPANNING-TREE-PROBLEM (Abschnitt 6.1).

DIJKSTRAS ALGORITHMUS

Input: Ein Digraph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$ und ein Knoten $s \in V(G)$.

Output: Kürzeste Wege von s zu allen $v \in V(G)$ und deren Längen.

Genauer, wir bekommen die Outputs $l(v)$ und $p(v)$ für alle $v \in V(G) \setminus \{s\}$. Es ist $l(v)$ die Länge eines kürzesten s - v -Weges, und dieser besteht aus einem kürzesten s - $p(v)$ -Weg plus der Kante $(p(v), v)$. Ist v nicht von s aus erreichbar, so ist $l(v) = \infty$ und $p(v)$ ist nicht definiert.

- ① Setze $l(s) := 0$. Setze $l(v) := \infty$ für alle $v \in V(G) \setminus \{s\}$.
Setze $R := \emptyset$.
- ② Finde einen Knoten $v \in V(G) \setminus R$ mit $l(v) = \min_{w \in V(G) \setminus R} l(w)$.
- ③ Setze $R := R \cup \{v\}$.
- ④ **For** alle $w \in V(G) \setminus R$ mit $(v, w) \in E(G)$ **do**:
If $l(w) > l(v) + c((v, w))$ **then**
setze $l(w) := l(v) + c((v, w))$ und $p(w) := v$.
- ⑤ **If** $R \neq V(G)$ **then go to** ②.

Satz 7.3. (Dijkstra [1959]) DIJKSTRAS ALGORITHMUS arbeitet korrekt.

Beweis: Wir beweisen, dass die folgenden zwei Aussagen zu jedem Zeitpunkt des Algorithmus gelten:

- (a) Für jedes $v \in V(G) \setminus \{s\}$ mit $l(v) < \infty$ gilt $p(v) \in R$ und $l(p(v)) + c((p(v), v)) = l(v)$, und die Folge $v, p(v), p(p(v)), \dots$ enthält s .
- (b) Für jedes $v \in R$ gilt $l(v) = \text{dist}_{(G, c)}(s, v)$.

Nach Beendigung von ① sind die beiden Aussagen trivialerweise erfüllt. In ④ wird nur dann $l(w)$ auf $l(v) + c((v, w))$ reduziert und $p(w)$ auf v gesetzt, wenn $v \in R$ und $w \notin R$. Da die Folge $v, p(v), p(p(v)), \dots$ den Knoten s enthält, aber keinen Knoten außerhalb R , insbesondere auch nicht den Knoten w , bleibt (a) in ④ erhalten.

Es ist (b) trivial für $v = s$. Nun benutzen wir Induktion über $|V(R)|$. Ange- nommen, der Knoten $v \in V(G) \setminus \{s\}$ wird in ③ zu R hinzugefügt, und es gäbe einen s - v -Weg P in G mit Länge weniger als $l(v)$. Sei y der erste auf P liegende und zu $(V(G) \setminus R) \cup \{v\}$ gehörende Knoten und x der Vorgänger von y auf P . Da $x \in R$, folgt mit ④ und der Induktionsvoraussetzung:

$$\begin{aligned} l(y) &\leq l(x) + c((x, y)) = \text{dist}_{(G,c)}(s, x) + c((x, y)) \\ &\leq c(E(P_{[s,y]})) \leq c(E(P)) < l(v), \end{aligned}$$

im Widerspruch zu der Wahl von v in ②. \square

Die Laufzeit ist offensichtlich $O(n^2)$. Diese können wir aber durch Benutzung eines Fibonacci-Heaps verbessern:

Satz 7.4. (Fredman und Tarjan [1987]) *Implementiert man DIJKSTRAS ALGORITHMUS mit einem Fibonacci-Heap, so läuft er in $O(m + n \log n)$ -Zeit, wobei $n = |V(G)|$ und $m = |E(G)|$.*

Beweis: Wir benutzen Satz 6.7 um die Menge $\{(v, l(v)) : v \in V(G) \setminus R, l(v) < \infty\}$ aufrechtzuerhalten. Dann bilden ② und ③ eine DELETEMIN-Operation, während die Aktualisierung von $l(w)$ in ④ eine INSERT-Operation ist, falls $l(w)$ unendlich war, anderenfalls ist sie eine DECREASEKEY-Operation. \square

Dies ist die beste bekannte streng polynomielle Laufzeit für das KÜRZESTE-WEGE-PROBLEM mit nichtnegativen Gewichten. (Mit einem anderen Rechenmodell erreichten Fredman und Willard [1994], Thorup [2000] und Raman [1997] etwas bessere Laufzeiten.)

Sind die Gewichte natürliche Zahlen aus einem festen endlichen Intervall, so gibt es einen einfachen Algorithmus mit linearer Laufzeit (Aufgabe 3). Im Allgemeinen sind $O(m \log \log c_{\max})$ -Laufzeiten (Johnson [1982]) und $O(m + n \sqrt{\log c_{\max}})$ -Laufzeiten (Ahuja et al. [1990]) möglich für Gewichte $c : E(G) \rightarrow \{0, \dots, c_{\max}\}$. Diese wurden von Thorup [2004] auf $O(m + n \log \log c_{\max})$ und $O(m + n \log \log n)$ verbessert, aber sogar die letztere Schranke gilt nur für ganzzahlige Kantengewichte und der Algorithmus ist nicht streng polynomiell. Für allgemeine nichtnegative Gewichte haben Orlin et al. [2010] einen $O(m \log(2 + nk/m))$ Algorithmus beschrieben, wobei k die Anzahl der verschiedenen Kantengewichte ist.

Für planare Digraphen gibt es einen Algorithmus mit linearer Laufzeit von Henzinger et al. [1997]. Schließlich erwähnen wir noch, dass Thorup [1999] einen Algorithmus mit linearer Laufzeit für die Bestimmung eines kürzesten Weges in einem ungerichteten Graphen mit nichtnegativen ganzzahligen Gewichten gefunden

hat. Siehe auch Pettie und Ramachandran [2005], sowie weitere zitierte Literatur in dieser Arbeit.

Wir wenden uns nun einem Algorithmus für allgemeine konservative Gewichte zu:

MOORE-BELLMAN-FORD-ALGORITHMUS

Input: Ein Digraph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$ und ein Knoten $s \in V(G)$.

Output: Ein negativer Kreis C in G , oder kürzeste Wege von s zu allen $v \in V(G)$ und deren Längen.

Genauer: Im zweiten Fall bekommen wir die Outputs $l(v)$ und $p(v)$ für alle $v \in V(G) \setminus \{s\}$. Es ist $l(v)$ die Länge eines kürzesten $s-v$ -Weges, und dieser besteht aus einem kürzesten $s-p(v)$ -Weg plus der Kante $(p(v), v)$. Ist v nicht von s aus erreichbar, so ist $l(v) = \infty$ und $p(v)$ ist nicht definiert.

-
- ① Setze $l(s) := 0$ und $l(v) := \infty$ für alle $v \in V(G) \setminus \{s\}$. Sei $n := |V(G)|$.
 - ② **For** $i := 1$ **to** $n - 1$ **do:**
 - For** jede Kante $(v, w) \in E(G)$ **do:**
 - If** $l(w) > l(v) + c((v, w))$ **then**
 - setze $l(w) := l(v) + c((v, w))$ und $p(w) := v$.
 - If** es gibt eine Kante $(v, w) \in E(G)$ mit $l(w) > l(v) + c((v, w))$ **then**
 - setze $x_n := w$, $x_{n-1} := v$, und $x_{n-i-1} := p(x_{n-i})$ für $i = 1, \dots, n - 1$, und liefere irgendeinen Kreis C in $(V(G), \{(x_{i-1}, x_i) : i = 1, \dots, n\})$ als Output.
-

Satz 7.5. (Moore [1959], Bellman [1958], Ford [1956]) *Der MOORE-BELLMAN-FORD-ALGORITHMUS arbeitet korrekt und hat $O(nm)$ -Laufzeit.*

Beweis: Die $O(nm)$ -Laufzeit ist klar. Zu irgendeinem Zeitpunkt des Algorithmus und für $v \in V(G)$, sei $k(v)$ diejenige Iteration in der $l(v)$ auf seinen momentanen Wert reduziert wurde, und setze $k(v) := 0$ falls $l(v)$ nach ① nicht mehr verändert wurde. Seien $F := \{(p(y), y) : y \in V(G), k(y) > 0\}$, und $F' := \{(v, w) \in E(G) : l(w) > l(v) + c((v, w))\}$. Wir behaupten nun, dass die folgenden zwei Aussagen immer gelten:

- (a) $l(y) \geq l(x) + c((x, y))$ und $k(x) \geq k(y) - 1$ für alle $(x, y) \in F$;
- (b) Enthält $(V(G), F \cup F')$ einen Kreis C , so hat C ein negatives Gesamtgewicht.

Zum Beweis von (a) beachten wir, dass $l(y) = l(x) + c((x, y))$ und $k(x) \geq k(y) - 1$ wenn $p(y)$ auf x gesetzt wird, und dass $l(x)$ niemals zunimmt und $k(x)$ niemals abnimmt.

Zum Beweis von (b) betrachten wir einen Kreis C in $(V(G), F \cup F')$. Nach (a) folgt $\sum_{(v,w) \in E(C)} c((v, w)) = \sum_{(v,w) \in E(C)} (c((v, w)) + l(v) - l(w)) \leq 0$. Damit ist (b) bis auf den Fall bewiesen, dass $E(C) \subseteq F$. Wird zu irgendeinem Zeitpunkt

ein Kreis C in $(V(G), F)$ erzeugt, indem man $p(y) := x$ setzt, dann hatte man $(x, y) \in F'$ unmittelbar vor diesem Schritt. Somit ist C ein negativer Kreis.

Findet der Algorithmus in ③ eine Kante $(v, w) \in F'$, so gilt $k(v) = n - 1$, also ist $k(x_i) \geq i$ für $i = n - 2, \dots, 1$. Somit ist die Folge x_0, \dots, x_n wohldefiniert und sie enthält eine Wiederholung. Also findet der Algorithmus in ③ tatsächlich einen Kreis C der nach (b) ein negatives Gesamtgewicht hat.

Bricht der Algorithmus ab mit $l(w) \leq l(v) + c((v, w))$ für alle $(v, w) \in E(G)$, so haben wir $\sum_{(v,w) \in E(C)} c((v, w)) = \sum_{(v,w) \in E(C)} (c((v, w)) + l(v) - l(w)) \geq 0$ für jeden Kreis C in $G[R]$, wobei $R := \{v \in V(G) : l(v) < \infty\}$. Also enthält $G[R]$ keinen negativen Kreis. Somit folgt aus (b), dass (R, F) azyklisch ist. Ferner folgt aus $x \in R \setminus \{s\}$, dass $p(x) \in R$, also ist (R, F) eine Arboreszenz mit Wurzel s .

Nach (a) ist $l(x)$ für jedes $x \in R$ (zu jedem beliebigen Zeitpunkt des Algorithmus) mindestens gleich der Länge des s - x -Weges in (R, F) .

Nun behaupten wir: Nach k Iterationen des Algorithmus ist für jedes $x \in R$ die Länge eines kürzesten s - x -Weges mit höchstens k Kanten mindestens gleich $l(x)$. Diese Aussage lässt sich leicht mittels Induktion beweisen. Sei P ein kürzester s - x -Weg mit höchstens k Kanten und sei (w, x) die letzte Kante von P . Wendet man nun Proposition 7.2 auf $G[R]$ an, so folgt, dass $P_{[s,w]}$ ein kürzester s - w -Weg mit höchstens $k - 1$ Kanten ist, und mit der Induktionsvoraussetzung folgt $l(w) \leq c(E(P_{[s,w]}))$ nach $k - 1$ Iterationen. In der k -ten Iteration wird aber die Kante (w, x) auch untersucht, demnach gilt $l(x) \leq l(w) + c((w, x)) \leq c(E(P))$.

Da es keine Wege mit mehr als $n - 1$ Kanten gibt, folgt mit der obigen Behauptung, dass der Algorithmus korrekt arbeitet. \square

Beachte: Ist c konservativ, so ist (R, F) eine Arboreszenz die einen kürzesten Weg von s zu jedem von s aus erreichbaren Knoten enthält. Eine solche Arboreszenz wird manchmal als *Kürzester-Wege-Baum* bezeichnet.

Dieser Algorithmus ist immer noch der schnellste bekannte streng polynomielle Algorithmus für das KÜRZESTE-WEGE-PROBLEM in Digraphen mit konservativen Gewichten. Ein von Goldberg [1995] stammender Skalierungsalgorithmus hat eine $O(\sqrt{nm} \log(C + 2))$ -Laufzeit, falls die Kantengewichte ganzzahlig sind mit Betrag höchstens C . Für planare Digraphen haben Mozes und Wulff-Nilsen [2010] einen $O(n \log^2 n / \log \log n)$ -Algorithmus beschrieben.

Für den Fall, dass G negative Kreise enthält, ist kein polynomieller Algorithmus für das KÜRZESTE-WEGE-PROBLEM bekannt (dieses Problem ist NP -schwer; siehe Aufgabe 17(b), Kapitel 15). Das Hauptproblem liegt darin, dass Proposition 7.2 nicht für allgemeine Gewichte gilt. Es ist nicht klar, wie man einen Weg anstatt einer beliebigen Kantenfolge konstruieren soll. Liegen keine negativen Kreise vor, so ist jede kürzeste Kantenfolge ein Weg, eventuell zuzüglich einiger Kreise mit Gewicht Null, die jedoch entfernt werden können. Nicht zuletzt in dieser Hinsicht ist es also eine wichtige Frage, wie man beweist, dass es keine negativen Kreise gibt. Der folgende nützliche Begriff stammt von Edmonds und Karp [1972]:

Definition 7.6. Sei G ein Digraph mit Gewichten $c : E(G) \rightarrow \mathbb{R}$ und $\pi : V(G) \rightarrow \mathbb{R}$. Dann definieren wir für jede Kante $(x, y) \in E(G)$ die **reduzierten Kosten** von (x, y) bezüglich π durch $c_\pi((x, y)) := c((x, y)) + \pi(x) - \pi(y)$. Ist $c_\pi(e) \geq 0$ für alle $e \in E(G)$, so heißt π ein **zulässiges Potenzial**.

Satz 7.7. Sei G ein Digraph mit Gewichten $c : E(G) \rightarrow \mathbb{R}$. Es gibt ein zulässiges Potenzial von (G, c) genau dann, wenn c konservativ ist. Für einen gegebenen Digraphen G mit Gewichten $c : E(G) \rightarrow \mathbb{R}$ können wir in $O(nm)$ Laufzeit entweder ein zulässiges Potenzial oder einen negativen Kreis finden.

Beweis: Ist π ein zulässiges Potenzial, so gilt für jeden Kreis C :

$$0 \leq \sum_{e \in E(C)} c_\pi(e) = \sum_{e=(x,y) \in E(C)} (c(e) + \pi(x) - \pi(y)) = \sum_{e \in E(C)} c(e),$$

wobei sich die Potenziale aufheben. Folglich ist c konservativ.

Um entweder einen negativen Kreis oder ein zulässiges Potenzial zu finden, fügen wir einen neuen Knoten s und mit Null gewichtete Kanten (s, v) für jedes $v \in V(G)$ hinzu. Nun wenden wir den MOORE-BELLMAN-FORD-ALGORITHMUS an und erhalten entweder einen negativen Kreis (der natürlich s nicht enthalten kann), oder Zahlen $l(v) < \infty$ für alle $v \in V(G)$ die ein zulässiges Potenzial l bilden. \square

Dies kann als eine besondere Form von LP-Dualität betrachtet werden; siehe Aufgabe 9.

In der Praxis werden effizientere Methoden zum Auffinden von negativen Kreisen gebraucht; siehe Cherkassky und Goldberg [1999].

7.2 Kürzeste Wege zwischen allen Knotenpaaren

Angenommen, wir wollen nun einen kürzesten s - t -Weg für alle geordneten Knotenpaare (s, t) in einem Digraphen finden:

KÜRZESTE-WEGE-PROBLEM FÜR ALLE PAARE

Instanz: Ein Digraph G und konservative Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Finde Zahlen l_{st} und Knoten p_{st} für alle $s, t \in V(G)$ mit $s \neq t$, so dass l_{st} die Länge eines kürzesten s - t -Weges und (p_{st}, t) die letzte Kante eines solchen Weges ist, falls es einen solchen gibt.

Natürlich könnten wir den MOORE-BELLMAN-FORD ALGORITHMUS n -mal anwenden, einmal für jede Wahl von s . Wir hätten sofort einen $O(n^2m)$ -Algorithmus. Es haben aber Bazaraa und Langley [1974] und Johnson [1977] gezeigt, dass es besser geht:

Satz 7.8. Das KÜRZESTE-WEGE-PROBLEM FÜR ALLE PAARE kann mit $O(mn + n^2 \log n)$ -Laufzeit gelöst werden, wobei $n = |V(G)|$ und $m = |E(G)|$.

Beweis: Sei (G, c) eine Instanz. Zunächst berechnen wir ein zulässiges Potenzial π , was nach Korollar 7.7 in $O(nm)$ -Zeit möglich ist. Danach berechnen wir für jedes einzelne $s \in V(G)$ kürzeste Wege von s aus, unter Benutzung der reduzierten Kosten c_π anstatt c . Für jeden Knoten t ist der resultierende $s-t$ -Weg dann auch ein kürzester Weg bezüglich c , da sich die Länge eines jeden $s-t$ -Weges um die konstante Größe $\pi(s) - \pi(t)$ ändert. Da die reduzierten Kosten nichtnegativ sind, können wir jedes Mal DIJKSTRAS ALGORITHMUS verwenden. Somit folgt nach Satz 7.4, dass die Gesamlaufzeit $O(mn + n(m + n \log n))$ ist. \square

Dieselbe Idee werden wir später in Kapitel 9 wieder benutzen (für den Beweis von Satz 9.14).

Pettie [2004] hat gezeigt, wie man die Laufzeit weiter auf $O(mn + n^2 \log \log n)$ verbessern kann; das ist die beste bekannte Schranke. Für dichte Graphen mit nichtnegativen Gewichten ist die $O(n^3 \log \log n / \log^2 n)$ -Schranke von Han und Takaoka [2016] etwas besser. Sind alle Kantengewichte kleine natürliche Zahlen, so kann man diese mittels schneller Matrizenmultiplikation noch weiter verbessern; siehe z. B. Zwick [2002].

Die Lösung des KÜRZESTE-WEGE-PROBLEMS FÜR ALLE PAARE erlaubt uns auch den metrischen Abschluss zu berechnen:

Definition 7.9. Gegeben sei ein Graph G (gerichtet oder ungerichtet) mit konservativen Gewichten $c : E(G) \rightarrow \mathbb{R}$. Der **metrische Abschluss** von (G, c) ist das Paar (\bar{G}, \bar{c}) , wobei \bar{G} der einfache Graph mit $V(\bar{G}) = V(G)$ ist, welcher für je zwei Knoten $x, y \in V(G)$ mit $x \neq y$ genau dann eine Kante $e = \{x, y\}$ (oder $e = (x, y)$ für G gerichtet) mit Gewicht $\bar{c}(e) = \text{dist}_{(G,c)}(x, y)$ enthält, wenn y von x aus in G erreichbar ist.

Korollar 7.10. Sei G ein Digraph mit konservativen Gewichten $c : E(G) \rightarrow \mathbb{R}$ oder ein ungerichteter Graph mit nichtnegativen Gewichten $c : E(G) \rightarrow \mathbb{R}_+$. Dann kann der metrische Abschluss von (G, c) in $O(mn + n^2 \log n)$ -Zeit berechnet werden.

Beweis: Ist G ungerichtet, so ersetzen wir jede Kante durch ein Paar entgegengesetzt gerichteter Kanten. Dann lösen wir die resultierende Instanz des KÜRZESTE-WEGE-PROBLEMS FÜR ALLE PAARE. \square

Der Rest dieses Abschnitts ist dem FLOYD-WARSHALL-ALGORITHMUS gewidmet, einem weiteren $O(n^3)$ -Algorithmus für das KÜRZESTE-WEGE-PROBLEM FÜR ALLE PAARE. Der wesentliche Vorteil des FLOYD-WARSHALL-ALGORITHMUS ist seine Einfachheit. Wir können o. B. d. A. annehmen, dass die Knoten mit $1, \dots, n$ nummeriert sind.

FLOYD-WARSHALL-ALGORITHMUS

Input: Ein Digraph G mit $V(G) = \{1, \dots, n\}$ und konservativen Gewichten $c : E(G) \rightarrow \mathbb{R}$.

Output: Matrizen $(l_{ij})_{1 \leq i, j \leq n}$ und $(p_{ij})_{1 \leq i, j \leq n}$, wobei l_{ij} die Länge eines kürzesten Weges von i nach j ist und (p_{ij}, j) die letzte Kante eines solchen Weges (falls es einen solchen gibt).

-
- ① Setze $l_{ij} := c((i, j))$ für alle $(i, j) \in E(G)$.
Setze $l_{ij} := \infty$ für alle $(i, j) \in (V(G) \times V(G)) \setminus E(G)$ mit $i \neq j$.
Setze $l_{ii} := 0$ für alle i .
Setze $p_{ij} := i$ für alle $i, j \in V(G)$.
- ② **For** $j := 1$ **to** n **do:**
For $i := 1$ **to** n **do:** **If** $i \neq j$ **then:**
For $k := 1$ **to** n **do:** **If** $k \neq j$ **then:**
If $l_{ik} > l_{ij} + l_{jk}$ **then** setze $l_{ik} := l_{ij} + l_{jk}$ und $p_{ik} := p_{jk}$.
-

Satz 7.11. (Floyd [1962], Warshall [1962]) *Der FLOYD-WARSHALL-ALGORITHMUS arbeitet korrekt und hat $O(n^3)$ -Laufzeit.*

Beweis: Die Laufzeit ist klar.

Behauptung: Nachdem der Algorithmus die äußere Schleife für $j = 1, 2, \dots, j_0$ durchlaufen hat, ist der Wert der Variable l_{ik} (für alle i und k) gleich der Länge eines kürzesten i - k -Weges, dessen Zwischenknoten v alle $\in \{1, \dots, j_0\}$ sind, und (p_{ik}, k) ist dessen letzte Kante.

Diese Behauptung werden wir für $j_0 = 0, \dots, n$ mittels Induktion beweisen. Für $j_0 = 0$ gilt sie nach ①, und für $j_0 = n$ ergibt sie die Korrektheit des Algorithmus.

Angenommen, die Behauptung gilt für ein $j_0 \in \{0, \dots, n-1\}$. Dann müssen wir zeigen, dass sie auch für j_0+1 gilt. Während des Ablaufs der äußeren Schleife für $j = j_0+1$ wird l_{ik} für jedes i und k durch $l_{i,j_0+1} + l_{j_0+1,k}$ ersetzt, falls dieser Wert kleiner als l_{ik} ist. Beachte dabei, dass der Wert von l_{ik} nach Induktionsvoraussetzung gleich der Länge eines kürzesten i - k -Weges ist, dessen Zwischenknoten v alle $\in \{1, \dots, j_0\}$ sind. Es bleibt zu zeigen, dass der entsprechende i - (j_0+1) -Weg P und der (j_0+1) - k -Weg Q keinen gemeinsamen inneren Knoten haben.

Angenommen, es gäbe einen sowohl zu P als auch zu Q gehörenden inneren Knoten. Indem wir einen maximalen geschlossenen Spaziergang in $P + Q$ (der aufgrund unserer Annahme ein nichtnegatives Gesamtgewicht hat, da er die Vereinigung von Kreisen ist) auslassen, also eine Abkürzung nehmen, erhalten wir einen i - k -Weg R , dessen Zwischenknoten v alle $\in \{1, \dots, j_0\}$ sind. Es ist R nicht länger als $l_{i,j_0+1} + l_{j_0+1,k}$ (und insbesondere kürzer als der Wert von l_{ik} vor dem Ablauf der äußeren Schleife für $j = j_0+1$).

Dies widerspricht jedoch der Induktionsvoraussetzung, da R nur Zwischenknoten $v \in \{1, \dots, j_0\}$ hat. \square

Wie der MOORE-BELLMAN-FORD-ALGORITHMUS, kann der FLOYD-WARSHALL-ALGORITHMUS auch zum Aufspüren von negativen Kreisen benutzt werden (siehe Aufgabe 13 und Hougardy [2010]).

Das KÜRZESTE-WEGE-PROBLEM FÜR ALLE PAARE in ungerichteten Graphen mit beliebigen konservativen Gewichten ist schwieriger; siehe Satz 12.14.

7.3 Kreise mit minimalem durchschnittlichem Kantengewicht

Mit den oben besprochenen Kürzeste-Wege-Algorithmen können wir leicht einen Kreis mit minimalem Gesamtgewicht in einem Digraphen mit konservativen Gewichten finden (siehe Aufgabe 14). Eine andere Frage ist: Wie findet man einen Kreis mit minimalem durchschnittlichem Kantengewicht?

GERICHTETES MINIMUM-MEAN-CYCLE-PROBLEM

Instanz: Ein Digraph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Finde einen Kreis C , dessen durchschnittliches Kantengewicht $\frac{c(E(C))}{|E(C)|}$ minimal ist, oder entscheide, dass G azyklisch ist.

In diesem Abschnitt werden wir zeigen, wie man dieses Problem mittels dynamischer Optimierung löst, ähnlich wie bei den Kürzeste-Wege-Algorithmen. Wir könnten G als stark zusammenhängend voraussetzen, da wir die starken Zusammenhangskomponenten in linearer Zeit bestimmen (Satz 2.19) und dann das Problem für jede einzelne starke Zusammenhangskomponente lösen können. In dem folgenden Min-Max-Satz genügt es jedoch anzunehmen, dass es einen Knoten s gibt, von dem aus alle Knoten erreichbar sind. Hier betrachten wir nicht nur Wege, sondern auch beliebige Kantenfolgen, eventuell mit Knoten- und Kantenwiederholungen.

Satz 7.12. (Karp [1978]) *Sei G ein Digraph mit Gewichten $c : E(G) \rightarrow \mathbb{R}$. Sei ferner $s \in V(G)$ mit der Eigenschaft, dass jeder Knoten von s aus erreichbar ist. Für $x \in V(G)$ und $k \in \mathbb{Z}_+$ sei*

$$F_k(x) := \min \left\{ \sum_{i=1}^k c((v_{i-1}, v_i)) : v_0 = s, v_k = x, (v_{i-1}, v_i) \in E(G) \text{ für alle } i \right\}$$

das minimale Gesamtgewicht einer Kantenfolge der Länge k von s nach x (und ∞ , falls es keine gibt). Sei $\mu(G, c)$ das minimale durchschnittliche Kantengewicht der Kreise in G (und $\mu(G, c) = \infty$, falls G azyklisch ist). Dann ist

$$\mu(G, c) = \min_{x \in V(G)} \max_{\substack{0 \leq k \leq n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n - k}.$$

Beweis: Ist G azyklisch, so ist $F_n(x) = \infty$ für alle $x \in V(G)$. Somit gilt der Satz. Nun nehmen wir an, dass $\mu(G, c) < \infty$.

Zunächst beweisen wir: Ist $\mu(G, c) = 0$, so gilt

$$\min_{x \in V(G)} \max_{\substack{0 \leq k \leq n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n - k} = 0.$$

Sei G ein Digraph mit $\mu(G, c) = 0$. Dann enthält G keinen negativen Kreis. Da c konservativ ist, haben wir $F_n(x) \geq \text{dist}_{(G,c)}(s, x) = \min_{0 \leq k \leq n-1} F_k(x)$, also ist

$$\max_{\substack{0 \leq k \leq n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n - k} \geq 0$$

für alle $x \in V(G)$. Wir zeigen nun, dass es einen Knoten x gibt, für den diese Ungleichung mit Gleichheit erfüllt ist, d. h. $F_n(x) = \text{dist}_{(G,c)}(s, x)$. Sei C irgendein Kreis in G mit Gewicht Null und $w \in V(C)$. Sei ferner P ein kürzester $s-w$ -Weg gefolgt von n Wiederholungen von C . Sei P' die Kantenfolge bestehend aus den ersten n Kanten von P und x der Endknoten von P' . Da P eine Kantenfolge mit minimalem Gesamtgewicht von s nach w ist, muss jedes Anfangsstück, insbesondere P' , eine Kantenfolge mit minimalem Gesamtgewicht sein. Somit ist $F_n(x) = c(E(P')) = \text{dist}_{(G,c)}(s, x)$.

Damit haben wir den Satz für den Fall $\mu(G, c) = 0$ bewiesen und wenden uns nun dem allgemeinen Fall zu. Beachte, dass die Addition einer Konstante zu allen Kantengewichten sowohl $\mu(G, c)$ als auch

$$\min_{x \in V(G)} \max_{\substack{0 \leq k \leq n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n - k}$$

um denselben Betrag verändert, nämlich um gerade diese Konstante. Wählt man Letztere gleich $-\mu(G, c)$, so haben wir den allgemeinen Fall auf den Fall $\mu(G, c) = 0$ zurückgeführt. \square

Dieser Satz legt den folgenden Algorithmus nahe:

MINIMUM-MEAN-CYCLE-ALGORITHMUS

Input: Ein Digraph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Output: Ein Kreis C mit minimalem durchschnittlichem Kantengewicht oder die Information, dass G azyklisch ist.

- ① Füge zu G einen Knoten s und Kanten (s, x) mit $c((s, x)) := 0$ für alle $x \in V(G)$ hinzu.
 - ② Setze $n := |V(G)|$, $F_0(s) := 0$, und $F_0(x) := \infty$ für alle $x \in V(G) \setminus \{s\}$.
 - ③ **For** $k := 1$ **to** n **do:**
 - For** alle $x \in V(G)$ **do:**
 - Setze $F_k(x) := \infty$.
 - For** alle $(w, x) \in \delta^-(x)$ **do:**
 - If** $F_{k-1}(w) + c((w, x)) < F_k(x)$ **then:**
 - Setze $F_k(x) := F_{k-1}(w) + c((w, x))$ und $p_k(x) := w$.
- ④ **If** $F_n(x) = \infty$ für alle $x \in V(G)$ **then stop** (G ist azyklisch).
- ⑤ Sei x ein Knoten, für den $\max_{\substack{0 \leq k \leq n-1 \\ F_k(x) < \infty}} \frac{F_n(x) - F_k(x)}{n - k}$ minimal ist.
- ⑥ Sei C irgendein Kreis in der durch $s = p_1(p_2(\dots(p_n(x))\dots)), \dots, p_{n-1}(p_n(x)), p_n(x), x$ gegebenen Kantenfolge.

Korollar 7.13. (Karp [1978]) *Der MINIMUM-MEAN-CYCLE-ALGORITHMUS arbeitet korrekt und hat $O(nm)$ Laufzeit.*

Beweis: In ① kommen keine neuen Kreise in G zustande, aber es wird Satz 7.12 anwendbar. Es ist klar, dass ② und ③ die Zahlen $F_k(x)$ korrekt berechnen. Terminiert der Algorithmus in ④, so ist G tatsächlich azyklisch.

Betrachte die Instanz (G, c') , wobei $c'(e) := c(e) - \mu(G, c)$ für alle $e \in E(G)$. Auf dieser Instanz läuft der Algorithmus genauso ab wie auf der Instanz (G, c) , mit dem einzigen Unterschied, dass die F -Werte verändert werden: von $F_k(x)$ auf $F'_k(x) = F_k(x) - k\mu(G, c)$. Mit der in ⑤ getroffenen Wahl von x , mit Satz 7.12 und da $\mu(G, c') = 0$, folgt $F'_n(x) = \min_{0 \leq k \leq n-1} F'_k(x)$. Somit besteht jede Kantenfolge von s nach x mit n Kanten und der Länge $F'_n(x)$ in (G, c') aus einem kürzesten s - x -Weg und einem oder mehreren Kreisen mit Gewicht Null. Diese Kreise haben das durchschnittliche Gesamtgewicht $\mu(G, c)$ in (G, c) .

Also ist jeder Kreis auf einer Kantenfolge mit minimalem Gewicht und der Länge n von s zu dem in ⑤ gewählten Knoten x ein Kreis mit minimalem durchschnittlichem Kantengewicht. In ⑥ wird ein solcher Kreis gewählt.

Die Laufzeit wird von ③ dominiert und ③ benötigt offensichtlich $O(nm)$ -Laufzeit. Beachte, dass ⑤ nur $O(n^2)$ -Laufzeit benötigt. \square

Dieser Algorithmus kann nicht zur Bestimmung eines Kreises mit minimalem durchschnittlichem Kantengewicht in einem ungerichteten Graphen mit Kantengewichten benutzt werden; siehe Aufgaben 10 und 18, Kapitel 12.

Algorithmen für Probleme zur Minimierung allgemeinerer Verhältnisse als dem von Gesamtgewicht zur Kantenanzahl in Kreisen wurden von Megiddo [1979,1983] und Radzik [1993] vorgeschlagen. Siehe auch Aufgabe 10, Kapitel 12.

7.4 Seichte leichte Bäume

Wir haben bereits effiziente Algorithmen für die Berechnung von aufspannenden Bäumen minimalen Gewichtes und von Kürzeste-Wege-Bäumen kennengelernt. Diese Bäume können sehr unterschiedlich aussehen. In ungerichteten Graphen können wir jedoch einen vernünftigen Kompromiss erreichen, indem wir einen leichten Baum mit kurzen Wegen von einer Quelle s zu allen anderen Knoten berechnen:

Satz 7.14. (Khuller, Raghavachari und Young [1995]) *Sei G ein ungerichteter Graph mit den Kantengewichten $c : E(G) \rightarrow \mathbb{R}_+$ und einer Quelle $s \in V(G)$. Sei $\epsilon > 0$. Dann gibt es einen aufspannenden Baum S in G , für den Folgendes gilt: $\text{dist}_{(S,c)}(s, v) \leq (1 + \epsilon) \text{dist}_{(G,c)}(s, v)$ für alle $v \in V(G)$, und $\sum_{e \in E(S)} c(e)$ ist höchstens gleich $(1 + \frac{2}{\epsilon})$ mal dem Gewicht eines aufspannenden Baumes minimalen Gewichtes in (G, c) . Eine solche Arboreszenz kann man in $O(m + n \log n)$ Laufzeit berechnen, wobei $n = |V(G)|$ und $m = |E(G)|$.*

Beweis: Sei S_0 ein aufspannender Baum minimalen Gewichtes in (G, c) und P ein Kürzeste-Wege-Baum, d. h. ein Baum, der für jedes $v \in V(G)$ einen kürzesten $s-v$ -Weg in (G, c) enthält. Die Bäume S_0 und P können mit PRIMS ALGORITHMUS bzw. DIJKSTRAS ALGORITHMUS berechnet werden, beide mit $O(m + n \log n)$ Laufzeit (Korollar 6.8 und Satz 7.4).

Man beginnt mit $S = S_0$. Dann verdoppelt man alle Kanten in S_0 und bestimmt einen eulerschen Spaziergang W (siehe Satz 2.25). Nun folge man diesem Spaziergang beginnend in s . Jedesmal wenn man erstmalig zu einem Knoten v kommt, für den $\text{dist}_{(S,c)}(s, v) > (1 + \epsilon) \text{dist}_{(G,c)}(s, v)$ gilt, füge man $E(P_{[s,v]})$ zu $E(S)$ hinzu, wobei $P_{[s,v]}$ der $s-v$ -Weg in P ist.

Dieses Verfahren liefert offensichtlich einen Graph S mit der Eigenschaft $\text{dist}_{(S,c)}(s, v) \leq (1 + \epsilon) \text{dist}_{(G,c)}(s, v)$ für alle $v \in V(G)$. Als Nächstes beschränken wir nun das gesamte Kantengewicht.

Seien v_1, \dots, v_k diejenigen Knoten v , für welche wir einen kürzesten $s-v$ -Weg zu S hinzufügen, und sei $v_0 := s$. Sei ferner $c_W(v_{i-1}, v_i)$ die Gesamtlänge der Kanten des Teilspaziergangs von W zwischen dem ersten Auftreten von v_{i-1} und dem ersten Auftreten von v_i . Wenn der Knotens v_i betrachtet wird, enthält der Graph S einen kürzesten $s-v_{i-1}$ -Weg, und es gilt

$$\text{dist}_{(G,c)}(s, v_{i-1}) + c_W(v_{i-1}, v_i) \geq \text{dist}_{(S,c)}(s, v_i) > (1 + \epsilon) \text{dist}_{(G,c)}(s, v_i).$$

Addieren wir diese Ungleichungen, so bekommen wir

$$\sum_{i=1}^k \text{dist}_{(G,c)}(s, v_{i-1}) + 2 \sum_{e \in E(S_0)} c(e) > (1 + \epsilon) \sum_{i=1}^k \text{dist}_{(G,c)}(s, v_i).$$

Also gilt

$$\epsilon \sum_{i=1}^k \text{dist}_{(G,c)}(s, v_i) \leq \epsilon \sum_{i=1}^{k-1} \text{dist}_{(G,c)}(s, v_i) + (1 + \epsilon) \text{dist}_{(G,c)}(s, v_k) < 2 \sum_{e \in E(S_0)} c(e),$$

und somit folgt

$$\sum_{e \in E(S)} c(e) = \sum_{e \in E(S_0)} c(e) + \sum_{i=1}^k \text{dist}_{(G,c)}(s, v_i) \leq \left(1 + \frac{2}{\epsilon}\right) \sum_{e \in E(S_0)} c(e)$$

wie gewünscht. Nun ist aber S normalerweise noch kein Baum, mit der Berechnung eines Kürzeste-Wege-Baumes in S von s aus erreichen wir jedoch das Ziel.

Zur Verifizierung der angegebenen Laufzeit beachte man, dass man nicht sämtliche Kanten von $P_{[s,v]}$ hinzufügen muss, sondern nur diejenigen, die nicht bereits früher hinzugefügt wurden. Speichert man bei jedem Knoten v , ob $P_{[s,v]}$ bereits in S enthalten ist oder nicht, so vermeidet man die wiederholte Betrachtung von Kanten in P .

Es verbleibt das Problem, wie man schnell genug bei der Betrachtung eines Knotens v entscheiden kann, ob $\text{dist}_{(S,c)}(s, v) > (1 + \epsilon) \text{dist}_{(G,c)}(s, v)$. Der

obige Beweis zeigt jedoch, dass man diese Bedingung durch die Bedingung $\text{dist}_{(G,c)}(s, v_{i-1}) + c_W(v_{i-1}, v_i) > (1 + \epsilon) \text{dist}_{(G,c)}(s, v)$ ersetzen kann, wobei v_{i-1} der letzte Knoten u ist, für den man einen kürzesten $s-u$ -Weg zu S hinzugefügt hat (oder s , wenn dies noch nie passiert ist). \square

Solche Bäume heißen seichte leichte Bäume; im Englischen heißen sie *shallow-light trees*. Der oben erwähnte Kompromiss kann nicht verbessert werden; siehe Aufgabe 19.

Aufgaben

1. Sei G ein Graph (gerichtet oder ungerichtet) mit Gewichten $c : E(G) \rightarrow \mathbb{Z}_+$ und $s, t \in V(G)$ mit t erreichbar von s aus. Man zeige: Die minimale Länge eines $s-t$ -Weges ist gleich der maximalen Anzahl von s und t trennenden Schnitten mit der Eigenschaft, dass jede Kante e in höchstens $c(e)$ von ihnen enthalten ist.
2. Man betrachte DIJKSTRAS ALGORITHMUS und nehme an, dass es nur um einen kürzesten $s-t$ -Weg für einen gegebenen Knoten t geht. Man zeige, dass man den Algorithmus terminieren kann, sobald $v = t$ oder $l(v) = \infty$ in ②.
3. Angenommen, alle Kantengewichte sind ganzzahlig und liegen zwischen 0 und einer Konstante C . Kann man DIJKSTRAS ALGORITHMUS für diesen besonderen Fall so implementieren, dass er lineare Laufzeit hat? (Dial [1969])
4. Gegeben seien ein Digraph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$ und zwei Knoten $s, t \in V(G)$. Angenommen, es gibt nur einen kürzesten $s-t$ -Weg P . Kann man dann den kürzesten $s-t$ -Weg abgesehen von P in polynomieller Zeit finden?
5. Man ändere DIJKSTRAS ALGORITHMUS ab, um das Bottleneck-Wege-Problem zu lösen: Gegeben sei ein Digraph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$ und zwei Knoten $s, t \in V(G)$; man finde einen $s-t$ -Weg, dessen längste Kante so kurz wie möglich ist.
6. Sei G ein Digraph mit $s, t \in V(G)$. Jeder Kante $e \in E(G)$ wird eine Zahl $r(e)$ (die Kantenzuverlässigkeit) zugeteilt, wobei $0 \leq r(e) \leq 1$. Die Zuverlässigkeit eines Weges P ist dann das Produkt der Kantenzuverlässigkeiten all seiner Kanten. Das Problem besteht darin, einen $s-t$ -Weg maximaler Zuverlässigkeit zu finden.
 - (a) Man zeige, dass man dieses Problem durch Logarithmierung auf ein KÜRZESTE-WEGE-PROBLEM zurückführen kann.
 - (b) Man zeige, wie man dieses Problem ohne Logarithmierung (in polynomieller Zeit) lösen kann.
7. Gegeben sei ein azyklischer Digraph G , $c : E(G) \rightarrow \mathbb{R}$ und $s, t \in V(G)$. Man zeige, wie man einen kürzesten $s-t$ -Weg in G in linearer Zeit finden kann.
8. Gegeben sei ein azyklischer Digraph G , $c : E(G) \rightarrow \mathbb{R}$ und $s, t \in V(G)$. Man zeige, wie man die Vereinigung von allen längsten $s-t$ -Wegen in G in linearer Zeit finden kann.

9. Man beweise Satz 7.7 mittels LP-Dualität, insbesondere Satz 3.24.
10. Sei G ein Digraph mit konservativen Gewichten $c : E(G) \rightarrow \mathbb{R}$. Seien $s, t \in V(G)$ mit t erreichbar von s aus. Man beweise: Die minimale Länge eines $s-t$ -Weges in G ist gleich dem maximalen Wert von $\pi(t) - \pi(s)$, wobei π ein zulässiges Potenzial von (G, c) ist.
11. Sei G ein stark zusammenhängender Digraph mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$. Sei $\emptyset \neq L \subseteq V(G)$ so dass jedes Element von L von allen Knoten aus erreichbar ist. Seien $s, t \in V(G)$ und $\pi(v) := \min\{0, \min_{l \in L}(\text{dist}_{(G,c)}(t, l) - \text{dist}_{(G,c)}(v, l))\}$ für $v \in V(G)$. Man beweise:
 - (a) Es ist π ein zulässiges Potenzial.
 - (b) Jeder kürzeste $s-t$ -Weg in (G, c_π) ist ein kürzester $s-t$ -Weg in (G, c) .
 - (c) Es gilt $\{v \in V(G) : \text{dist}_{(G,c_\pi)}(s, v) < \text{dist}_{(G,c_\pi)}(s, t)\} \subseteq \{v \in V(G) : \text{dist}_{(G,c)}(s, v) < \text{dist}_{(G,c)}(s, t)\}$.

Bemerkung: Muss eine große Anzahl kürzester Wege in einem großen Graphen berechnet werden, so könnte es von Vorteil sein, zunächst die Entfernung zu einer geeignet gewählten Menge $L \subseteq V(G)$ (von Orientierungspunkten, im Englischen **landmarks**) zu berechnen. Mittels (a), (b), (c), und Aufgabe 2 kann man dann die Entfernung zu diesen 'landmarks' dazu benutzen, die Suche nach kürzesten $s-t$ -Wegen für viele Paare s und t zu beschleunigen, indem man DIJKSTRAS ALGORITHMUS auf (G, c_π) anwendet.
 (Goldberg und Harrelson [2005])
12. Sei G ein Digraph, $V(G) = A \dot{\cup} B$ und $E(G[B]) = \emptyset$. Angenommen, es ist $|\delta(v)| \leq k$ für alle $v \in B$. Seien $s, t \in V(G)$ und $c : E(G) \rightarrow \mathbb{R}$ konservativ. Man beweise, dass man einen kürzesten $s-t$ -Weg in $O(|A|k|E(G)|)$ -Zeit finden kann und, falls c nichtnegativ ist, in $O(|A|^2)$ -Zeit.
 (Orlin [1993])
13. Man nehme an, der FLOYD-WARSHALL-ALGORITHMUS wird auf eine Instanz (G, c) mit beliebigen Gewichten $c : E(G) \rightarrow \mathbb{R}$ angewendet. Man beweise, dass alle l_{ii} ($i = 1, \dots, n$) genau dann nichtnegativ bleiben, wenn c konservativ ist.
14. Gegeben sei ein Digraph mit konservativen Gewichten. Man zeige, wie man einen Kreis mit minimalem Gesamtgewicht in polynomieller Zeit findet. Kann man eine $O(n^3)$ Laufzeit erreichen?

Hinweis: Man modifiziere den FLOYD-WARSHALL-ALGORITHMUS etwas.
Bemerkung: Orlin und Sedefño-Noda [2017] haben einen Algorithmus mit $O(nm)$ Laufzeit gefunden. Für allgemeine Gewichte enthält das Problem die Entscheidung, ob ein gegebener Digraph hamiltonsch ist (und ist somit NP-schwer; siehe Kapitel 15). In Abschnitt 12.2 wird beschrieben, wie man den Kreis mit minimalem Gewicht in einem ungerichteten Graphen (mit konservativen Gewichten) findet.
15. Sei G ein vollständiger (ungerichteter) Graph und $c : E(G) \rightarrow \mathbb{R}_+$. Man zeige, dass (G, c) genau dann sein eigener metrischer Abschluss ist, wenn die Dreiecksungleichung gilt: $c(\{x, y\}) + c(\{y, z\}) \geq c(\{x, z\})$ für je drei paarweise verschiedene Knoten $x, y, z \in V(G)$.

16. Die Zeitbedingungen („timing constraints“) eines Logikchips lassen sich durch einen Digraphen G mit Kantengewichten $c : E(G) \rightarrow \mathbb{R}_+$ darstellen. Dabei entsprechen die Knoten den Speicherelementen und die Kanten gewissen durch die kombinatorische Logik definierten Wegen, während die Gewichte den „worst-case“ Schätzungen der Signallaufzeiten entsprechen. Eine wichtige Aufgabe des VLSI-Chip-Designs (VLSI bedeutet „very large scale integration“) ist es, einen optimalen Takt-Zeitplan zu finden, d.h. eine Abbildung $a : V(G) \rightarrow \mathbb{R}$ mit der Eigenschaft, dass $a(v) + c((v, w)) \leq a(w) + T$ für alle $(v, w) \in E(G)$ und eine möglichst kleine Zahl T . (Es ist T die Zykluszeit des Chips und $a(v)$ bzw. $a(v) + T$ sind die „Abfahrtszeit“ bzw. die späteste zulässige „Ankunftszeit“ des Signals in v .)
- Man reduziere das Problem, das optimale T zu finden, auf das GERICHTE-MINIMUM-MEAN-CYCLE-PROBLEM.
 - Man zeige, wie man die Zahlen $a(v)$ einer optimalen Lösung effizient bestimmen kann.
 - Es ist typisch, einige der Zahlen $a(v)$ vorab festzulegen. Man zeige, wie man in diesem Fall das Problem lösen würde.
- (Albrecht et al. [2002])
17. Man zeige, dass die folgende Variante des MINIMUM-MEAN-CYCLE-ALGORITHMUS auch korrekt arbeitet: Kein weiterer Knoten s wird hinzugefügt, und statt ① und ② setzen wir nur $F_0(x) := 0$ für alle $x \in V(G)$.
18. Man finde einen polynomiellen Algorithmus für das folgende Problem: Für einen gegebenen ungerichteten Graph G , zwei seiner Knoten s, t und eine natürliche Zahl k bestimme man, ob es eine Kantenfolge der Länge k von s nach t in G gibt.
Hinweis: k kann viel größer als n sein. Man betrachte den Fall $k > 2n$ für sich.
19. Man zeige, dass es für jedes $\epsilon > 0$ und $\delta < 1 + \frac{2}{\epsilon}$ einen gewichteten Graphen (G, c) gibt, in dem jeder aufspannende Baum S , für den $\text{dist}_{(S,c)}(s, v) \leq (1 + \epsilon) \text{dist}_{(G,c)}(s, v)$ für alle $v \in V(G)$ gilt, ein Gesamtgewicht größer als δ mal dem Gewicht eines aufspannenden Baumes minimalen Gewichtes hat.
(Khuller, Raghavachari und Young [1995])

Literatur

Allgemeine Literatur:

- Ahuja, R.K., Magnanti, T.L., und Orlin, J.B. [1993]: Network Flows. Prentice-Hall, Englewood Cliffs 1993, Kapitel 4 und 5
- Cormen, T.H., Leiserson, C.E., Rivest, R.L., und Stein, C. [2009]: Introduction to Algorithms. 3. Aufl. MIT Press, Cambridge 2009, Kapitel 24 und 25
- Dreyfus, S.E. [1969]: An appraisal of some shortest path algorithms. Operations Research 17 (1969), 395–412
- Gallo, G., und Pallottino, S. [1988]: Shortest paths algorithms. Annals of Operations Research 13 (1988), 3–79

- Gondran, M., und Minoux, M. [1984]: Graphs and Algorithms. Wiley, Chichester 1984, Kapitel 2
- Lawler, E.L. [1976]: Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston, New York 1976, Kapitel 3
- Schrijver, A. [2003]: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin 2003, Kapitel 6–8
- Tarjan, R.E. [1983]: Data Structures and Network Algorithms. SIAM, Philadelphia 1983, Kapitel 7

Zitierte Literatur:

- Ahuja, R.K., Mehlhorn, K., Orlin, J.B., und Tarjan, R.E. [1990]: Faster algorithms for the shortest path problem. *Journal of the ACM* 37 (1990), 213–223
- Albrecht, C., Korte, B., Schietke, J., und Vygen, J. [2002]: Maximum mean weight cycle in a digraph and minimizing cycle time of a logic chip. *Discrete Applied Mathematics* 123 (2002), 103–127
- Bazaraa, M.S., und Langley, R.W. [1974]: A dual shortest path algorithm. *SIAM Journal on Applied Mathematics* 26 (1974), 496–501
- Bellman, R.E. [1958]: On a routing problem. *Quarterly of Applied Mathematics* 16 (1958), 87–90
- Cherkassky, B.V., und Goldberg, A.V. [1999]: Negative-cycle detection algorithms. *Mathematical Programming A* 85 (1999), 277–311
- Dial, R.B. [1969]: Algorithm 360: shortest path forest with topological order. *Communications of the ACM* 12 (1969), 632–633
- Dijkstra, E.W. [1959]: A note on two problems in connexion with graphs. *Numerische Mathematik* 1 (1959), 269–271
- Edmonds, J., und Karp, R.M. [1972]: Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM* 19 (1972), 248–264
- Floyd, R.W. [1962]: Algorithm 97 – shortest path. *Communications of the ACM* 5 (1962), 345
- Ford, L.R. [1956]: Network flow theory. Paper P-923, The Rand Corporation, Santa Monica 1956
- Fredman, M.L., und Tarjan, R.E. [1987]: Fibonacci heaps and their uses in improved network optimization problems. *Journal of the ACM* 34 (1987), 596–615
- Fredman, M.L., und Willard, D.E. [1994]: Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *Journal of Computer and System Sciences* 48 (1994), 533–551
- Goldberg, A.V. [1995]: Scaling algorithms for the shortest paths problem. *SIAM Journal on Computing* 24 (1995), 494–504
- Goldberg, A.V., und Harrelson, C. [2005]: Computing the shortest path: *A** search meets graph theory. *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms* (2005), 156–165
- Han, Y., und Takaoka, T. [2016]: An $O(n^3 \log \log n / \log^2 n)$ time algorithm for all pairs shortest paths. *Journal of Discrete Algorithms* 38–41 (2016), 9–19
- Henzinger, M.R., Klein, P., Rao, S., und Subramanian, S. [1997]: Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences* 55 (1997), 3–23
- Hougardy, S. [2010]: The Floyd-Warshall algorithm on graphs with negative cycles. *Information Processing Letters* 110 (2010), 279–281

- Johnson, D.B. [1977]: Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM* 24 (1977), 1–13
- Johnson, D.B. [1982]: A priority queue in which initialization and queue operations take $O(\log \log D)$ time. *Mathematical Systems Theory* 15 (1982), 295–309
- Karp, R.M. [1978]: A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics* 23 (1978), 309–311
- Khuller, S., Raghavachari, B., und Young, N. [1995]: Balancing minimum spanning trees and shortest-path trees. *Algorithmica* 14 (1995), 305–321
- Megiddo, N. [1979]: Combinatorial optimization with rational objective functions. *Mathematics of Operations Research* 4 (1979), 414–424
- Megiddo, N. [1983]: Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM* 30 (1983), 852–865
- Moore, E.F. [1959]: The shortest path through a maze. *Proceedings of the International Symposium on the Theory of Switching, Part II*, Harvard University Press, 1959, 285–292
- Mozes, S., und Wulff-Nilsen, C. [2010]: Shortest paths in planar graphs with real lengths in $O(n \log^2 n / \log \log n)$ time. In: *Algorithms – Proceedings of the 18th European Symposium on Algorithms (ESA), Part II; LNCS 6347* (M. de Berg, U. Meyer, Hrsg.), Springer, Berlin 2010, pp. 206–217
- Orlin, J.B. [1993]: A faster strongly polynomial minimum cost flow algorithm. *Operations Research* 41 (1993), 338–350
- Orlin, J.B., Madduri, K., Subramani, K., und Williamson, M. [2010]: A faster algorithm for the single source shortest path problem with few distinct positive lengths. *Journal of Discrete Algorithms* 8 (2010), 189–198
- Orlin, J.B., und Sedeño-Noda, A. [2017]: An $O(nm)$ time algorithm for finding the min length directed cycle in a graph. *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms* (2017), 1866–1879
- Pettie, S. [2004]: A new approach to all-pairs shortest paths on real-weighted graphs. *Theoretical Computer Science* 312 (2004), 47–74
- Pettie, S., und Ramachandran, V. [2005]: Computing shortest paths with comparisons and additions. *SIAM Journal on Computing* 34 (2005), 1398–1431
- Radzik, T. [1993]: Parametric flows, weighted means of cuts, and fractional combinatorial optimization. In: *Complexity in Numerical Optimization* (P.M. Pardalos, Hrsg.), World Scientific, Singapore 1993
- Raman, R. [1997]: Recent results on the single-source shortest paths problem. *ACM SIGACT News* 28 (1997), 81–87
- Thorup, M. [1999]: Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM* 46 (1999), 362–394
- Thorup, M. [2000]: On RAM priority queues. *SIAM Journal on Computing* 30 (2000), 86–109
- Thorup, M. [2004]: Integer priority queues with decrease key in constant time and the single source shortest paths problem. *Journal of Computer and System Sciences* 69 (2004), 330–353
- Warshall, S. [1962]: A theorem on boolean matrices. *Journal of the ACM* 9 (1962), 11–12
- Zwick, U. [2002]: All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM* 49 (2002), 289–317



8 Netzwerkflüsse

In diesem und dem nächsten Kapitel werden wir Flüsse in Netzwerken betrachten. Wir gehen aus von einem Digraphen G mit Kantenkapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und zwei vorgegebenen Knoten, s (die **Quelle**) und t (die **Senke**). Das Quadrupel (G, u, s, t) wird auch als **Netzwerk** bezeichnet.

Es geht hier hauptsächlich darum, so viele Einheiten wie möglich gleichzeitig von s nach t zu befördern. Eine Lösung dieses Problems heißt ein maximaler Fluss. Genauer:

Definition 8.1. Gegeben sei ein Digraph G mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$. Ein **Fluss** ist eine Funktion $f : E(G) \rightarrow \mathbb{R}_+$ mit $f(e) \leq u(e)$ für alle $e \in E(G)$. Der **Überschuss** (im Englischen: **excess**) eines Flusses f in $v \in V(G)$ ist

$$\text{ex}_f(v) := \sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e).$$

Man sagt, dass f die **Flusserhaltungsregel** im Knoten v erfüllt, wenn $\text{ex}_f(v) = 0$. Ein Fluss, der die Flusserhaltungsregel in jedem Knoten erfüllt, ist eine **Zirkulation**.

Sei nun ein Netzwerk (G, u, s, t) gegeben. Ein **s-t-Fluss** ist ein Fluss f mit $\text{ex}_f(s) \leq 0$ und $\text{ex}_f(v) = 0$ für alle $v \in V(G) \setminus \{s, t\}$. Der **Wert** eines **s-t-Flusses** f ist $\text{value}(f) := -\text{ex}_f(s)$.

Nun können wir das grundlegende Problem dieses Kapitels formulieren:

MAXIMUM-FLOW-PROBLEM

Instanz: Ein Netzwerk (G, u, s, t) .

Aufgabe: Bestimme einen **s-t-Fluss** mit maximalem Wert.

O. B. d. A. können wir annehmen, dass G einfach ist, da parallele Kanten vorher zusammengelegt werden können.

Dieses Problem hat zahlreiche Anwendungen. Betrachten wir z. B. das **JOB-ZUORDNUNGSPROBLEM**: Gegeben seien n Arbeiten (Jobs) mit den für ihre Erledigung benötigten Zeiten $t_1, \dots, t_n \in \mathbb{R}_+$, m Arbeiter und für jede Arbeit $i \in \{1, \dots, n\}$ eine nichtleere Teilmenge $S_i \subseteq \{1, \dots, m\}$ derjenigen Arbeiter, die an der Arbeit i tätig sein können. Wir fordern Zahlen $x_{ij} \in \mathbb{R}_+$ für alle $i = 1, \dots, n$ und $j \in S_i$ (nämlich die vom Arbeiter j an der Arbeit i verbrachte Zeit), so dass alle Arbeiten erledigt werden, d. h. $\sum_{j \in S_i} x_{ij} = t_i$ für alle $i = 1, \dots, n$. Das Ziel ist

es, alle Arbeiten in minimaler Zeit zu erledigen, d. h. $T(x) := \max_{j=1}^m \sum_{i:j \in S_i} x_{ij}$ zu minimieren. Anstatt dieses Problem mit LINEARER OPTIMIERUNG zu lösen, möchten wir hier einen kombinatorischen Algorithmus finden.

Wir werden das optimale $T(x)$ mittels binärer Suche finden. Dazu müssen wir für einen festen Wert T Zahlen $x_{ij} \in \mathbb{R}_+$ mit $\sum_{j \in S_i} x_{ij} = t_i$ für alle i und $\sum_{i:j \in S_i} x_{ij} \leq T$ für alle j finden. Die Mengen S_i stellen wir durch einen (bipartiten) Digraphen dar, mit Knoten v_i für die Jobs i , Knoten w_j für die Arbeiter j und Kanten (v_i, w_j) , falls $j \in S_i$. Ferner fügen wir zwei weitere Knoten s und t und Kanten (s, v_i) für alle i und (w_j, t) für alle j hinzu. Sei G dieser Graph, für den wir Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ mittels $u((s, v_i)) := t_i$ und $u(e) := T$ für alle anderen Kanten definieren. Dann entsprechen den zulässigen Lösungen x mit $T(x) \leq T$ offensichtlich gerade die $s-t$ -Flüsse mit Wert $\sum_{i=1}^n t_i$ in (G, u) . Diese sind in der Tat maximale Flüsse.

In Abschnitt 8.1 werden wir einen grundlegenden Algorithmus für das MAXIMUM-FLOW-PROBLEM angeben und ihn benutzen, um das Max-Flow-Min-Cut-Theorem zu beweisen. Dieser Satz ist eines der bekanntesten Resultate in der kombinatorischen Optimierung und zeigt die Verbindung zum Problem der Bestimmung eines $s-t$ -Schnittes minimaler Kapazität auf. Ferner werden wir zeigen, dass es für ganzzahlige Kapazitäten immer einen optimalen Fluss gibt, der ganzzahlig ist. Aus diesen beiden Resultaten zusammen ergibt sich der Satz von Menger über disjunkte Wege, wie wir in Abschnitt 8.2 sehen werden.

Die Abschnitte 8.3, 8.4 und 8.5 beinhalten effiziente Algorithmen für das MAXIMUM-FLOW-PROBLEM. Danach werden wir unsere Aufmerksamkeit auf das Problem der Bestimmung von Schnitten minimaler Kapazität richten. In Abschnitt 8.6 beschreiben wir eine elegante Methode zur Speicherung der minimalen Kapazität eines $s-t$ -Schnittes (welche gleich dem maximalen Wert eines $s-t$ -Flusses ist) für alle Knotenpaare s und t . In Abschnitt 8.7 zeigen wir, wie man den Kantenzusammenhang oder auch einen Schnitt minimaler Kapazität in einem ungerichteten Graphen auf effizientere Weise bestimmen kann als durch die Anwendung mehrerer Netzwerkflussberechnungen.

8.1 Das Max-Flow-Min-Cut-Theorem

Die Definition des MAXIMUM-FLOW-PROBLEMS legt die folgende LP-Formulierung nahe:

$$\begin{aligned} \max \quad & \sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e \\ \text{bzgl.} \quad & \sum_{e \in \delta^-(v)} x_e = \sum_{e \in \delta^+(v)} x_e \quad (v \in V(G) \setminus \{s, t\}) \\ & x_e \leq u(e) \quad (e \in E(G)) \\ & x_e \geq 0 \quad (e \in E(G)). \end{aligned}$$

Da dieses LP offensichtlich beschränkt und der Null-Fluss $f \equiv 0$ immer zulässig ist, haben wir die folgende Tatsache:

Proposition 8.2. *Das MAXIMUM-FLOW-PROBLEM hat stets eine optimale Lösung.* □

Ferner folgt mittels Satz 4.18, dass es einen polynomiellen Algorithmus gibt. Wir sind mit diesem Resultat jedoch noch nicht zufrieden, sondern möchten einen kombinatorischen Algorithmus haben, der ohne LINEARE OPTIMIERUNG auskommt.

Wir erinnern daran, dass ein s - t -Schnitt in G eine Kantenmenge $\delta^+(X)$ mit $s \in X$ und $t \in V(G) \setminus X$ ist. Die **Kapazität** eines s - t -Schnittes ist die Summe der Kapazitäten seiner Kanten. Ein minimaler s - t -Schnitt in (G, u) ist ein s - t -Schnitt minimaler Kapazität (bezüglich u) in G .

Lemma 8.3. *Für jede Knotenmenge $A \subseteq V(G)$ mit $s \in A$ und $t \notin A$ und für jeden s - t -Fluss f gilt:*

- (a) $\text{value}(f) = \sum_{e \in \delta^+(A)} f(e) - \sum_{e \in \delta^-(A)} f(e),$
- (b) $\text{value}(f) \leq \sum_{e \in \delta^+(A)} u(e).$

Beweis: (a): Da die Flusserhaltungsregel für $v \in A \setminus \{s\}$ erfüllt ist, gilt:

$$\begin{aligned} \text{value}(f) &= \sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e) \\ &= \sum_{v \in A} \left(\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) \right) \\ &= \sum_{e \in \delta^+(A)} f(e) - \sum_{e \in \delta^-(A)} f(e). \end{aligned}$$

(b): Diese Aussage folgt aus (a), da $0 \leq f(e) \leq u(e)$ für alle $e \in E(G)$. \square

Anders ausgedrückt: Der Wert eines maximalen Flusses kann nicht größer als die minimale Kapazität eines s - t -Schnittes sein. In der Tat sind die beiden Größen gleich. Um dies zu beweisen, benötigen wir den Begriff des augmentierenden Weges, der auch in einigen weiteren Kapiteln vorkommen wird.

Definition 8.4. Sei G ein Digraph und $e = (v, w) \in E(G)$. Dann sei \overleftarrow{e} eine neue Kante von w nach v , die sogenannte **gegenläufige Kante** von e . Ebenso ist e die **gegenläufige Kante** von \overleftarrow{e} . Nun definieren wir den Graphen $\overset{\leftrightarrow}{G} := (V(G), E(G) \dot{\cup} \{\overleftarrow{e} : e \in E(G)\})$. Beachte: Sind $e = (v, w)$, $e' = (w, v) \in E(G)$, dann sind \overleftarrow{e} und e' zwei verschiedene parallele Kanten in $\overset{\leftrightarrow}{G}$.

Gegeben sei ein Digraph G mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und ein Fluss f . Dann definieren wir die **Residualkapazitäten** $u_f : E(G) \xrightarrow{\leftrightarrow} \mathbb{R}_+$ durch $u_f(e) := u(e) - f(e)$ und $u_f(\overleftarrow{e}) := f(e)$ für alle $e \in E(G)$. Der **Residualgraph** G_f ist der Graph $(V(G), \{e \in E(\overset{\leftrightarrow}{G}) : u_f(e) > 0\})$.

Gegeben sei ein Fluss f und ein Weg (oder Kreis) P in G_f . Den Fluss f entlang P um γ zu **augmentieren** bedeutet, man erhöhe $f(e)$ um γ für jedes $e \in E(P)$ mit $e \in E(G)$, und man verringere $f(e_0)$ um γ für jedes $e \in E(P)$ mit $e = \overleftarrow{e_0}$ für ein $e_0 \in E(G)$.

Gegeben sei ein Netzwerk (G, u, s, t) und ein s - t -Fluss f . Dann ist ein f -**augmentierender Weg** ein s - t -Weg in dem Residualgraphen G_f .

Mit diesem Begriff können wir nun den folgenden Algorithmus für das MAXIMUM-FLOW-PROBLEM, der von Ford und Fulkerson [1957] stammt, auf natürliche Weise formulieren. Zunächst beschränken wir uns auf ganzzahlige Kapazitäten.

FORD-FULKERSON-ALGORITHMUS

Input: Ein Netzwerk (G, u, s, t) mit $u : E(G) \rightarrow \mathbb{Z}_+$.

Output: Ein $s-t$ -Fluss f mit maximalem Wert.

- ① Setze $f(e) := 0$ für alle $e \in E(G)$.
- ② Bestimme einen f -augmentierenden Weg P . **If** es gibt keinen **then stop**.
- ③ Berechne $\gamma := \min_{e \in E(P)} u_f(e)$. Augmentiere f entlang P um γ und **go to** ②.

Kanten, auf denen das Minimum in ③ angenommen wird, nennt man auch Bottleneck-Kanten. Die Wahl von γ gewährleistet, dass f weiterhin ein Fluss ist. Da P ein $s-t$ -Weg ist, ist die Flusserhaltungsregel in allen Knoten außer s und t erfüllt.

Die Bestimmung eines augmentierenden Weges ist einfach: Man braucht nur einen $s-t$ -Weg in G_f zu finden. Dabei sollte man aber Acht geben. Erlaubt man nämlich irrationale Kapazitäten (und hat bei der Wahl der augmentierenden Wege Pech), so kann es passieren, dass der Algorithmus überhaupt nicht terminiert (Aufgabe 2).

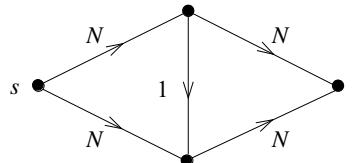


Abbildung 8.1.

Aber auch bei ganzzahligen Kapazitäten kann es exponentiell viele Augmentierungen geben. Ein Beispiel hierfür ist das einfache in Abb. 8.1 gezeigte Netzwerk, wobei die Zahlen die Kantenkapazitäten sind ($N \in \mathbb{N}$). Wählen wir bei jeder Iteration einen Weg der Länge 3, so können wir den Fluss jedes Mal nur um 1 augmentieren, also benötigen wir $2N$ Iterationen. Beachte, dass die Inputlänge $O(\log N)$ ist, da Kapazitäten selbstverständlich binär codiert werden. Diese Probleme werden wir in Abschnitt 8.3 aus dem Weg räumen.

Wir behaupten nun: Wenn der Algorithmus terminiert, ist f ein maximaler Fluss.

Satz 8.5. Ein s - t -Fluss f hat genau dann maximalen Wert, wenn es keinen f -augmentierenden Weg gibt.

Beweis: Gibt es einen augmentierenden Weg P , so wird in Schritt ③ des FORD-FULKERSON-ALGORITHMUS ein Fluss mit höherem Wert berechnet, also ist f kein Fluss mit maximalem Wert. Gibt es andererseits keinen augmentierenden Weg, so bedeutet dies, dass t nicht von s aus in G_f erreichbar ist. Sei R die Menge der von s aus in G_f erreichbaren Knoten. Nach der Definition von G_f folgt $f(e) = u(e)$ für alle $e \in \delta_G^+(R)$ und $f(e) = 0$ für alle $e \in \delta_G^-(R)$.

Nach Lemma 8.3 (a) folgt aber

$$\text{value}(f) = \sum_{e \in \delta_G^+(R)} u(e),$$

und mit Lemma 8.3 (b) folgt dann, dass f ein Fluss mit maximalem Wert ist. \square

Insbesondere haben wir bewiesen, dass man für jeden s - t -Fluss mit maximalem Wert einen s - t -Schnitt hat, dessen Kapazität gleich dem Wert des Flusses ist. Zusammen mit Lemma 8.3 (b) bekommen wir das zentrale Resultat der Theorie der Netzwerkflüsse, nämlich das Max-Flow-Min-Cut-Theorem:

Satz 8.6. (Ford und Fulkerson [1956], Dantzig und Fulkerson [1956]) In einem Netzwerk ist der maximale Wert eines s - t -Flusses gleich der minimalen Kapazität eines s - t -Schnittes. \square

Ein anderer Beweis ist von Elias, Feinstein und Shannon [1956] gegeben worden. Das Max-Flow-Min-Cut-Theorem folgt auch problemlos aus der LP-Dualität, siehe Aufgabe 9, Kapitel 3.

Sind alle Kapazitäten ganzzahlig, so ist das γ in ③ des FORD-FULKERSON-ALGORITHMUS immer ganzzahlig. Da es einen maximalen Fluss mit endlichem Wert gibt (Proposition 8.2), terminiert der Algorithmus nach endlich vielen Schritten. Somit haben wir die folgende wichtige Aussage:

Korollar 8.7. (Dantzig und Fulkerson [1956]) Sind die Kapazitäten eines Netzwerkes ganzzahlig, so gibt es einen ganzzahligen maximalen Fluss. \square

Dieses Korollar, welches manchmal der Satz über ganzzahlige Flüsse genannt wird, kann auch leicht mittels der vollständigen Unimodularität der Inzidenzmatrix eines Digraphen bewiesen werden (Aufgabe 3).

Wir schließen diesen Abschnitt mit einem weiteren einfachen aber nützlichen Resultat, nämlich dem Flussdekompositionssatz:

Satz 8.8. (Gallai [1958], Ford und Fulkerson [1962]) Sei (G, u, s, t) ein Netzwerk und f ein s - t -Fluss in G . Dann gibt es eine Familie \mathcal{P} von s - t -Wegen in G , eine Familie \mathcal{C} von Kreisen in G und Gewichte $w : \mathcal{P} \cup \mathcal{C} \rightarrow \mathbb{R}_+$, so dass $f(e) = \sum_{P \in \mathcal{P} \cup \mathcal{C}, e \in E(P)} w(P)$ für alle $e \in E(G)$, $\sum_{P \in \mathcal{P}} w(P) = \text{value}(f)$ und $|\mathcal{P}| + |\mathcal{C}| \leq |E(G)|$.

Ferner gilt: Ist f ganzzahlig, so können die Gewichte w ganzzahlig gewählt werden.

Beweis: Wir konstruieren \mathcal{P} , \mathcal{C} und w mittels Induktion über die Anzahl der Kanten mit nichtverschwindendem Fluss. Wir nehmen an, es gebe eine Kante e^+ mit $f(e^+) > 0$; sonst ist die Aussage trivial. Betrachte einen maximalen e^+ enthaltenden Spaziergang W mit den folgenden zwei Eigenschaften: jede Kante hat positiven Fluss und kein Knoten kommt zweimal vor, wenn er nicht möglicherweise einer der Endknoten ist. Es enthält W höchstens n Kanten. Ferner enthält W einen Kreis P , oder aber W ist ein Weg P . Gilt Letzteres, so beginnt P in s , endet in t , und $f(\delta^-(s)) = f(\delta^+(t)) = 0$ (wegen der Maximalität von W und der Flusserhaltungsregel).

Setze $w(P) := \min_{e \in E(P)} f(e)$, $f'(e) := f(e) - w(P)$ für $e \in E(P)$ und $f'(e) := f(e)$ für $e \notin E(P)$. Mit Anwendung der Induktionsvoraussetzung auf f' ist der Beweis abgeschlossen. \square

Dieser Beweis liefert auch einen $O(mn)$ -Algorithmus zur Berechnung einer solchen Flussdekomposition.

8.2 Der Satz von Menger

Betrachten wir Korollar 8.7 und Satz 8.8 mit sämtlichen Kapazitäten gleich 1, so können wir ganzzahlige s - t -Flüsse als Familien von paarweise kantendisjunkten s - t -Wegen und Kreisen betrachten und bekommen somit den folgenden wichtigen Satz:

Satz 8.9. (Menger [1927]) *Sei G ein Graph (gerichtet oder ungerichtet), seien s und t zwei Knoten und $k \in \mathbb{N}$. Es gibt k paarweise kantendisjunkte s - t -Wege genau dann, wenn t nach dem Entfernen von $k - 1$ beliebigen Kanten weiterhin von s aus erreichbar ist.*

Beweis: Die Notwendigkeit ist klar. Um zu beweisen, dass die Bedingung im gerichteten Fall hinreichend ist, sei (G, u, s, t) ein Netzwerk mit allen Kapazitäten gleich 1 und der Eigenschaft, dass t nach dem Entfernen von $k - 1$ beliebigen Kanten weiterhin von s aus erreichbar ist. Damit ist die minimale Kapazität eines s - t -Schnittes mindestens k . Mit dem Max-Flow-Min-Cut-Theorem (Satz 8.6) und Korollar 8.7 gibt es dann einen ganzzahligen s - t -Fluss mit Wert mindestens k . Nach Satz 8.8 kann dieser Fluss in ganzzahlige Flüsse auf s - t -Wegen (und eventuell auf einigen Kreisen) zerlegt werden. Da alle Kapazitäten gleich 1 sind, muss es es mindestens k paarweise kantendisjunkte s - t -Wege geben.

Um zu beweisen, dass die Bedingung im ungerichteten Fall hinreichend ist, sei G ein ungerichteter Graph und s und t zwei Knoten von G mit der Eigenschaft, dass t nach dem Entfernen von $k - 1$ beliebigen Kanten weiterhin von s aus erreichbar ist. Diese Eigenschaft bleibt offensichtlich bei der folgenden Operation erhalten: Man ersetze jede ungerichtete Kante $e = \{v, w\}$ durch fünf gerichtete Kanten: (v, x_e) , (w, x_e) , (x_e, y_e) , (y_e, v) und (y_e, w) , wobei x_e und y_e neue Knoten sind (siehe Abb. 8.2). Es sei G' der neue Digraph, der nach dem ersten Teil des

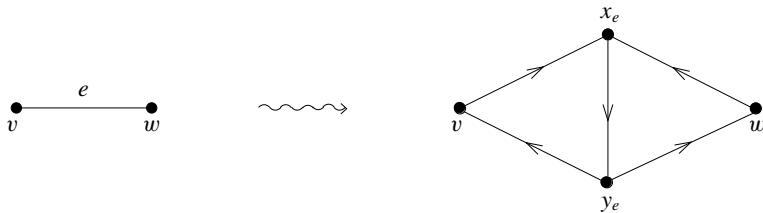


Abbildung 8.2.

Beweises k paarweise kantendisjunkte $s-t$ -Wege hat. Diese können nun leicht in k paarweise kantendisjunkte $s-t$ -Wege in G transformiert werden. \square

Andererseits lässt sich das Max-Flow-Min-Cut-Theorem leicht aus dem Satz von Menger ableiten (wenigstens für rationale Kapazitäten). Nun werden wir die knotendisjunkte Version des Satzes von Menger betrachten. Wir nennen eine Menge von Wegen **intern disjunkt**, falls keine zwei Wege der Menge eine Kante oder einen inneren Knoten gemeinsam haben. Obwohl sie einen oder beide Endknoten gemeinsam haben können, werden intern disjunkte Wege gelegentlich auch knotendisjunkt genannt (bei vorgegebenen Endknoten).

Satz 8.10. (Menger [1927]) *Sei G ein Graph (gerichtet oder ungerichtet), seien s und t zwei nicht benachbarte Knoten und $k \in \mathbb{N}$. Es gibt k intern disjunkte $s-t$ -Wege genau dann, wenn t nach dem Entfernen von $k-1$ beliebigen Knoten (abgesehen von s und t) weiterhin von s aus erreichbar ist.*

Beweis: Die Notwendigkeit ist wiederum klar. Dass die Bedingung im gerichteten Fall hinreichend ist, folgt aus dem gerichteten Teil von Satz 8.9 mittels der folgenden elementaren Konstruktion: Wir ersetzen jeden Knoten v von G durch zwei Knoten v' und v'' und eine Kante (v', v'') , und jede Kante (v, w) von G durch die Kante (v'', w) . Jede $(k-1)$ -elementige Kantenmenge in dem neuen Graphen G' , deren Entfernen t' von s'' aus unerreichbar macht, bedeutet die Existenz einer höchstens $(k-1)$ -elementigen weder s noch t enthaltenden Knotenmenge in G , deren Entfernen t von s aus unerreichbar macht. Ferner entsprechen den paarweise kantendisjunkten $s''-t'$ -Wegen in G' intern disjunkte $s-t$ -Wege in G .

Der Beweis im ungerichteten Fall folgt aus demjenigen für den gerichteten Fall mittels derselben Konstruktion wie im Beweis von Satz 8.9 (Abb. 8.2). \square

Das folgende Korollar ist eine wichtige Folgerung aus dem Satz von Menger:

Korollar 8.11. (Whitney [1932]) *Ein ungerichteter Graph G mit mindestens zwei Knoten ist genau dann k -fach kantenzusammenhängend, wenn es für jedes Knotenpaar $s, t \in V(G)$ mit $s \neq t$ k paarweise kantendisjunkte $s-t$ -Wege gibt.*

Ein ungerichteter Graph G mit mehr als k Knoten ist genau dann k -fach zusammenhängend, wenn es für jedes Knotenpaar $s, t \in V(G)$ mit $s \neq t$ k intern disjunkte $s-t$ -Wege gibt.

Beweis: Die erste Aussage folgt direkt aus Satz 8.9.

Zum Beweis der zweiten Aussage, sei G ein ungerichteter Graph mit mehr als k Knoten. Gibt es in G eine $(k - 1)$ -elementige Knotenmenge, deren Entfernen G unzusammenhängend macht, dann kann G nicht k intern disjunkte $s-t$ -Wege für jedes Paar $s, t \in V(G)$ haben.

Zur umgekehrten Richtung: Hat G für irgendein Paar $s, t \in V(G)$ keine k intern disjunkten $s-t$ -Wege, dann betrachten wir zwei Fälle. Sind s und t nicht benachbart, so folgt aus Satz 8.10, dass es in G eine $(k - 1)$ -elementige Knotenmenge gibt, deren Entfernen s und t trennt.

Sind s und t durch eine Menge F paralleler Kanten mit $|F| \geq 1$ verbunden, so hat $G - F$ keine $k - |F|$ intern disjunkten $s-t$ -Wege, somit folgt nach Satz 8.10, dass $G - F$ eine Menge X von $k - |F| - 1$ Knoten hat, deren Entfernen s und t trennt. Sei $v \in V(G) \setminus (X \cup \{s, t\})$. Dann kann v nicht sowohl von s als auch von t aus in $(G - F) - X$ erreichbar sein. Ist etwa v von s aus unerreichbar, dann liegen v und s in verschiedenen Zusammenhangskomponenten von $G - (X \cup \{t\})$.

□

In vielen Anwendungen sucht man paarweise kanten- oder knotendisjunkte (oder auch intern disjunkte) Wege zwischen mehreren Knotenpaaren. Die vier Versionen des Satzes von Menger (gerichtet oder ungerichtet, kanten- oder knotendisjunkt) entsprechen den vier Versionen des DISJUNKTE-WEGE-PROBLEMS:

GERICHTETES / UNGERICHTETES KANTENDISJUNKTE-WEGE- / KNOTENDISJUNKTE-WEGE-PROBLEM

Instanz: Zwei gerichtete / ungerichtete Graphen (G, H) auf derselben Knotenmenge.

Aufgabe: Bestimme eine Familie $(P_f)_{f \in E(H)}$ von paarweise kantendisjunkten / intern disjunkten Wegen in G , so dass P_f für jedes $f = (t, s)$ oder $f = \{t, s\}$ in H ein $s-t$ -Weg ist.

Eine solche Familie heißt eine **Lösung** von (G, H) . Man sagt: P_f realisiert f . Die Kanten von G heißen **Angebotskanten**, diejenigen von H **Nachfragekanten**. Ein mit einer Nachfragekante inzidenter Knoten heißt **Terminal**.

Oben haben wir den Spezialfall $E(H)$ gleich einer Menge von k parallelen Kanten betrachtet. Das allgemeine DISJUNKTE-WEGE-PROBLEM werden wir in Kapitel 19 behandeln. Hier erwähnen wir nur den folgenden nützlichen Spezialfall des Satzes von Menger:

Proposition 8.12. Sei (G, H) eine Instanz des GERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, wobei $E(H)$ nur eine Menge paralleler Kanten und $G + H$ eulersch ist. Dann hat (G, H) eine Lösung.

Beweis: Da $G + H$ eulersch ist, gehört jede Kante, insbesondere auch jedes $f \in E(H)$, einem Kreis C an. Dann nehmen wir $C - f$ als den ersten Weg unserer Lösung, entfernen C und wenden Induktion an. □

8.3 Der Edmonds-Karp-Algorithmus

In Aufgabe 2 wird gezeigt, dass man ② des FORD-FULKERSON-ALGORITHMUS genauer spezifizieren muss. Anstatt einen beliebigen augmentierenden Weg zu wählen, sollte man einen kürzesten suchen, d.h. einen augmentierenden Weg mit einer minimalen Anzahl von Kanten. Mit dieser einfachen Idee ist es Edmonds und Karp [1972] gelungen, den ersten polynomiellen Algorithmus für das MAXIMUM-FLOW-PROBLEM zu finden.

EDMONDS-KARP-ALGORITHMUS

Input: Ein Netzwerk (G, u, s, t) .

Output: Ein $s-t$ -Fluss f mit maximalem Wert.

- ① Setze $f(e) := 0$ für alle $e \in E(G)$.
- ② Bestimme einen kürzesten f -augmentierenden Weg P . **If** es gibt keinen **then stop**.
- ③ Berechne $\gamma := \min_{e \in E(P)} u_f(e)$. Augmentiere f entlang P um γ und **go to** ②.

Dies bedeutet, dass man ② im FORD-FULKERSON-ALGORITHMUS mit BFS implementieren sollte (siehe Abschnitt 2.3).

Lemma 8.13. Sei f_1, f_2, \dots eine Folge von Flüssen, wobei f_{i+1} aus f_i hervorgeht, indem man entlang einem kürzesten f_i -augmentierenden Weg P_i augmentiert. Dann gilt:

- (a) $|E(P_k)| \leq |E(P_{k+1})|$ für alle k .
- (b) $|E(P_k)| + 2 \leq |E(P_l)|$ für alle $k < l$, so dass $P_k \cup P_l$ ein Paar gegenläufiger Kanten enthält.

Beweis: (a): Betrachte den aus $P_k \cup P_{k+1}$ durch Entfernen aller Paare gegenläufiger Kanten resultierenden Graphen G_1 . (Kanten, die sowohl in P_k als auch in P_{k+1} vorkommen, erscheinen doppelt in G_1 .) Jeder einfache Teilgraph von G_1 ist ein Teilgraph von G_{f_k} , da jede Kante in $E(G_{f_{k+1}}) \setminus E(G_{f_k})$ die gegenläufige Kante einer Kante in P_k ist.

Sei H_1 der aus nur zwei Kopien von (t, s) bestehende Graph. Offensichtlich ist $G_1 + H_1$ eulersch. Nach Proposition 8.12 gibt es dann zwei kantendisjunkte $s-t$ -Wege Q_1 und Q_2 . Da $E(G_1) \subseteq E(G_{f_k})$, sind Q_1 und Q_2 beide f_k -augmentierende Wege. Da P_k ein kürzester f_k -augmentierender Weg war, folgt $|E(P_k)| \leq |E(Q_1)|$ und $|E(P_k)| \leq |E(Q_2)|$. Damit haben wir

$$2|E(P_k)| \leq |E(Q_1)| + |E(Q_2)| \leq |E(G_1)| \leq |E(P_k)| + |E(P_{k+1})|$$

und somit $|E(P_k)| \leq |E(P_{k+1})|$.

(b): Nach (a) genügt es, die Aussage nur für diejenigen k, l mit der Eigenschaft, dass $P_i \cup P_l$ für $k < i < l$ keine Paare gegenläufiger Kanten enthält, zu zeigen.

Wie oben, betrachten wir den aus $P_k \cup P_l$ durch Entfernen aller Paare gegenläufiger Kanten resultierenden Graphen G_1 . Wiederum behaupten wir, dass jeder einfache Teilgraph von G_1 ein Teilgraph von G_{f_k} ist. Beachte dazu, dass $E(P_k) \subseteq E(G_{f_k})$ und $E(P_l) \subseteq E(G_{f_l})$, und dass jede Kante von $E(G_{f_l}) \setminus E(G_{f_k})$ die gegenläufige Kante einer Kante auf einem der Wege $P_k, P_{k+1}, \dots, P_{l-1}$ ist. Unter diesen Wegen enthält aber – wegen der Wahl von k und l – nur P_k die gegenläufige Kante einer Kante auf P_l .

Sei H_1 wiederum der aus zwei Kopien von (t, s) bestehende Graph. Da $G_1 + H_1$ eulersch ist, gibt es nach Proposition 8.12 zwei kantendisjunkte $s-t$ -Wege Q_1 und Q_2 . Wiederum sind Q_1 und Q_2 beide f_k -augmentierend. Da P_k ein kürzester f_k -augmentierender Weg war, folgt $|E(P_k)| \leq |E(Q_1)|$ und $|E(P_k)| \leq |E(Q_2)|$. Damit haben wir

$$2|E(P_k)| \leq |E(Q_1)| + |E(Q_2)| \leq |E(P_k)| + |E(P_l)| - 2$$

(da wir mindestens zwei Kanten entfernt haben), womit der Beweis abgeschlossen ist. \square

Satz 8.14. (Edmonds und Karp [1972]) *Unabhängig von den Kantenkapazitäten terminiert der EDMONDS-KARP-ALGORITHMUS nach höchstens $\frac{mn}{2}$ Augmentierungen, wobei m bzw. n die Anzahl der Kanten bzw. Knoten ist.*

Beweis: Seien P_1, P_2, \dots die während des EDMONDS-KARP-ALGORITHMUS gewählten augmentierenden Wege. Wegen der Wahl von γ in ③ des Algorithmus enthält jeder augmentierende Weg wenigstens eine Bottleneck-Kante.

Für irgendeine Kante e sei P_{i_1}, P_{i_2}, \dots die Teilfolge der e als Bottleneck-Kante enthaltenden augmentierenden Wege. Offensichtlich muss zwischen P_{i_j} und $P_{i_{j+1}}$ ein $\overset{\leftarrow}{e}$ enthaltender augmentierender Weg P_k mit $i_j < k < i_{j+1}$ liegen. Nach Lemma 8.13 (b) gilt $|E(P_{i_j})| + 4 \leq |E(P_k)| + 2 \leq |E(P_{i_{j+1}})|$ für alle j . Hat e weder s noch t als Endknoten, so folgt $3 \leq |E(P_{i_j})| \leq n-1$ für alle j , also können höchstens $\frac{n}{4}$ augmentierende Wege e als Bottleneck-Kante enthalten. Andernfalls enthält höchstens einer der augmentierenden Wege e oder $\overset{\leftrightarrow}{e}$ als Bottleneck-Kante.

Da jeder augmentierende Weg mindestens eine Kante von $\overset{\leftrightarrow}{G}$ als Bottleneck-Kante enthält, kann es höchstens $|E(\overset{\leftrightarrow}{G})| \frac{n}{4} = \frac{mn}{2}$ augmentierende Wege geben. \square

Korollar 8.15. *Der EDMONDS-KARP-ALGORITHMUS löst das MAXIMUM-FLOW-PROBLEM mit $O(m^2n)$ -Laufzeit.*

Beweis: Nach Satz 8.14 gibt es höchstens $\frac{mn}{2}$ Augmentierungen. Jede Augmentierung benutzt BFS und benötigt somit $O(m)$ -Laufzeit. \square

8.4 Dinic', Karzanovs, und Fujishiges Algorithmus

Ungefähr zu der Zeit, wo Edmonds und Karp die Möglichkeit eines polynomiellen Algorithmus für das MAXIMUM-FLOW-PROBLEM entdeckten, beschrieb Dinic

[1970] unabhängig einen noch besseren Algorithmus. Er basiert auf folgender Definition:

Definition 8.16. Gegeben sei ein Netzwerk (G, u, s, t) und ein s - t -Fluss f . Der Level-Graph G_f^L von G_f ist der Digraph

$$(V(G), \{e = (x, y) \in E(G_f) : \text{dist}_{G_f}(s, x) + 1 = \text{dist}_{G_f}(s, y)\}).$$

Beachte, dass der Level-Graph azyklisch ist. Er kann leicht mit BFS in $O(m)$ -Zeit erstellt werden. Die s - t -Wege in G_f^L sind genau die kürzesten s - t -Wege in G_f .

Lemma 8.13(a) besagt, dass die Länge der kürzesten augmentierenden Wege während des EDMONDS-KARP-ALGORITHMUS nicht abnimmt. Wir nennen eine Folge augmentierender Wege gleicher Länge eine **Phase** des Algorithmus. Sei f der Fluss am Anfang einer Phase. Der Beweis von Lemma 8.13(b) ergibt, dass alle augmentierenden Wege dieser Phase bereits augmentierende Wege in G_f sind. Damit sind all diese Wege s - t -Wege in dem Level-Graphen von G_f . Die Gesamtheit aller Augmentierungen in einer Phase kann als ein blockierender Fluss in G_f^L betrachtet werden:

Definition 8.17. Ein s - t -Fluss f in einem gegebenen Netzwerk (G, u, s, t) heißt **blockierend**, wenn der Graph $(V(G), \{e \in E(G) : f(e) < u(e)\})$ keine s - t -Wege enthält.

Beachte, dass ein blockierender Fluss nicht maximal zu sein braucht. Diese Betrachtungen legen das folgende algorithmische Schema nahe:

DINIC' ALGORITHMUS

Input: Ein Netzwerk (G, u, s, t) .

Output: Ein s - t -Fluss f mit maximalem Wert.

- ① Setze $f(e) := 0$ für alle $e \in E(G)$.
- ② Erstelle den Level-Graph G_f^L von G_f .
- ③ Bestimme einen blockierenden s - t -Fluss f' in (G_f^L, u_f) . **If** $f' = 0$ **then stop**.
- ④ Augmentiere f um f' und **go to** ②.

Augmentieren wir f um f' , so bedeutet dies natürlich, dass wir $f(e)$ um $f'(e)$ für jedes $e \in E(G_f^L) \cap E(G)$ erhöhen, und $f(e)$ um $f'(\overleftarrow{e})$ für jedes $e \in E(G)$ mit $\overleftarrow{e} \in E(G_f^L)$ verringern.

Satz 8.18. (Dinic [1970]) DINIC' ALGORITHMUS arbeitet korrekt und terminiert nach höchstens n Iterationen.

Beweis: Terminiert der Algorithmus, so gibt es keinen s - t -Weg in G_f^L , also auch nicht in G_f . Es bleibt zu zeigen, dass die Länge eines kürzesten augmentierenden Weges (die aus $\{1, 2, \dots, n-1, \infty\}$ sein muss) mit jeder Iteration ansteigt. Wir betrachten nun den Fluss f am Anfang einer beliebigen Iteration. Es sei \bar{f} der augmentierte Fluss f nach ④. Beachte, dass $\text{dist}_{G_f}(s, y) \leq \text{dist}_{G_f}(s, x)$ für alle $(x, y) \in E(G_f) \setminus E(G_{\bar{f}}^L)$ und auch für gegenläufige Kanten (x, y) von Kanten in G_f^L , während $\text{dist}_{G_f}(s, y) = \text{dist}_{G_f}(s, x) + 1$ für alle $e = (x, y) \in E(G_f^L)$. Da jeder augmentierende Weg bezüglich \bar{f} nur Kanten aus $E(G_f) \cup \{\overleftarrow{e} : e \in E(G_f^L)\}$ enthält und auch mindestens eine Kante die nicht in G_f^L ist, enthält er mehr als $\text{dist}_{G_f}(s, t)$ Kanten. \square

Beachte, dass ② (unter Benutzung von BFS) und ④ in linearer Laufzeit implementiert werden können. Also bleibt nur noch zu zeigen, wie man einen blockierenden Fluss in einem azyklischen Digraphen auf effiziente Weise findet. Dinic hat eine $O(nm)$ -Schranke für jede Phase gefunden; dies ist nicht sehr schwer zu zeigen (Aufgabe 20). Wir werden nun Karzanovs schnelleren Algorithmus beschreiben. Er basiert auf der folgenden wichtigen Definition:

Definition 8.19. (Karzanov [1974]) Gegeben sei ein Netzwerk (G, u, s, t) . Ein s - t -Präfluss ist eine Funktion $f : E(G) \rightarrow \mathbb{R}_+$ mit $f(e) \leq u(e)$ für alle $e \in E(G)$ und $\text{ex}_f(v) \geq 0$ für alle $v \in V(G) \setminus \{s\}$. Ein Knoten $v \in V(G) \setminus \{s, t\}$ heißt **aktiv**, falls $\text{ex}_f(v) > 0$.

Offensichtlich ist ein s - t -Präfluss genau dann ein s - t -Fluss, wenn es keine aktiven Knoten gibt. Diesen Begriff werden wir im nächsten Abschnitt wieder antreffen.

Satz 8.20. (Karzanov [1974]) Ein blockierender Fluss in einem Netzwerk (G, u, s, t) mit einem azyklischen Digraphen G kann in $O(n^2)$ -Laufzeit bestimmt werden, wobei $n = |V(G)|$.

Beweis: Zunächst berechne man eine topologische Reihenfolge $V(G) = \{v_1, v_2, \dots, v_n\}$ (siehe Satz 2.20). Für jeden Knoten v außer s und t speichert der Algorithmus $\text{ex}_f(v)$, die aktuelle Liste der dort beginnenden Kanten und einen zunächst leeren Stapel (auf Englisch 'stack') von Paaren aus $\delta^-(v) \times \mathbb{R}_+$. Alle Knoten sind am Anfang als nicht eingefroren markiert.

Der Algorithmus speichert durchweg einen s - t -Präfluss f . Am Anfang setzen wir $f(e) := u(e)$ für jede Kante $e = (s, v) \in \delta^+(s)$ und platzieren $(e, u(e))$ auf dem Stapel von v . Für alle anderen Kanten e setzen wir am Anfang $f(e) := 0$. Dann laufen die folgenden zwei Schritte alternierend ab, bis keine aktiven Knoten mehr vorhanden sind:

Im Push-Schritt werden die Knoten in ihrer topologischen Reihenfolge gescannt. Solange ein Knoten v aktiv ist und es eine Kante $e = (v, w)$ gibt mit $f(e) < u(e)$ und w nicht eingefroren, erhöhen wir $f(e)$ um $\delta := \min\{u(e) - f(e), \text{ex}_f(v)\}$ und platzieren (e, δ) auf dem Stapel von w .

Im Ausgleichs-Schritt bearbeiten wir den aktiven Knoten v_i für den i maximal ist. Wir entfernen das oberste Paar (e, δ) des Stapels von v_i (d. h. das letzte dort platzierte Paar) und verringern $f(e)$ um $\delta' := \min\{\delta, \text{ex}_f(v_i)\}$. Ist v_i nach wie vor aktiv, so wiederholen wir diesen Schritt mit dem nächsten Paar des Stapels. Am Ende ist v_i nicht mehr aktiv und wird als eingefroren markiert.

Wird ein Knoten v_i einem Ausgleichs-Schritt unterzogen, so sind zu dem Zeitpunkt alle Knoten v_j mit $j > i$ nicht aktiv, also werden ihre ankommenden Kanten niemals weniger Fluss als zum jetzigen Zeitpunkt enthalten. Dies folgt aus der Tatsache, dass der Fluss während eines Ausgleichs-Schrittes gemäß dem Stapel reduziert wird: zuerst in Kanten für die der Fluss zuletzt erhöht worden war. Demnach wird v_i nie mehr aktiv werden. Somit wird jeder Knoten höchstens einmal einem Ausgleichs-Schritt unterzogen, also ist die Anzahl der Iterationen kleiner als n . Jeder Push-Schritt hat $O(n + p)$ -Laufzeit, wobei p die Anzahl der Kanten ist, die einen saturierenden Push erhalten. Beachte, dass eine Kante die saturiert worden ist, niemals mehr in einem Push-Schritt betrachtet werden wird. Somit ist die Gesamtaufzeit für alle Push-Schritte $O(n^2 + m)$, und dies gilt auch für die Gesamtaufzeit aller Ausgleichs-Schritte.

Zu jedem Zeitpunkt des Algorithmus ist t in $(V(G), \{e \in E(G) : f(e) < u(e)\})$ weder von s noch von irgendeinem eingefrorenen Knoten aus erreichbar. Somit ist f zum Schluss ein blockierender Fluss. \square

Es folgt nun ein zweiter, von Malhotra, Kumar und Maheshwari [1978] stammender Beweis:

Zweiter Beweis des Satzes 8.20: Zunächst berechne man eine topologische Reihenfolge von G (siehe Satz 2.20). Für ein gegebenes $v \in V(G)$ sei $G_{\leq v}$ bzw. und $G_{\geq v}$ der von allen Knoten bis v bzw. von v an induzierte Teilgraph.

Wir beginnen mit $f(e) = 0$ für alle $e \in E(G)$. Sei

$$\alpha := \min \left\{ \min\{u_f(\delta_G^-(v)) : v \in V(G) \setminus \{s\}\}, \min\{u_f(\delta_G^+(v)) : v \in V(G) \setminus \{t\}\} \right\},$$

und v ein Knoten in dem das Minimum angenommen wird.

Zunächst bestimmen wir einen v - t -Fluss g mit dem Wert α in $G_{\geq v}$ (außer wenn $v = t$). Dies bewerkstelligen wir indem die Knoten in topologischer Reihenfolge gescannt werden. Für jeden Knoten w mit der Eigenschaft: $w = v$ oder es sind bereits Vorfahren bearbeitet worden, setzen wir $\beta := \alpha$ falls $w = v$, und $\beta := g(\delta_G^-(w))$ sonst. Sei ferner $\delta_G^+(w) = \{e_1, \dots, e_k\}$ und setze $g(e_j) := \min\{u_f(e_j), \beta - \sum_{i=1}^{j-1} g(e_i)\}$ für $j = 1, \dots, k$. Analog bestimmen wir einen s - v -Fluss g' mit dem Wert α in $G_{\leq v}$ (außer wenn $v = s$), indem die Knoten in umgekehrter topologischer Reihenfolge gescannt werden. Wir setzen nun $f(e) := f(e) + g(e) + g'(e)$ für alle $e \in E(G)$. Ist $v \in \{s, t\}$, so hören wir auf: f ist ein blockierender Fluss. Ist $v \notin \{s, t\}$, so entfernen wir v und die Kanten mit denen v inzident ist, und iterieren.

Dieser Algorithmus terminiert nach höchstens $n - 1$ Iterationen und arbeitet offensichtlich korrekt. Er kann so implementiert werden, dass jede Kante höchstens einmal gescannt wird nachdem sie saturiert worden ist. Bei jeder Iteration genügt es, wenn höchstens $n - 2$ Kanten gescannt werden (eine an jedem Knoten) zusätzlich

zu denen, die in dieser Iteration saturiert werden. Indem wir immer die Werte von $u_f(\delta_G^-(v))$ und $u_f(\delta_G^+(v))$ aktualisieren, können wir α und v bei jeder Iteration in $O(n)$ -Laufzeit bestimmen. Somit folgt die $O(m + n^2)$ -Laufzeit des Algorithmus. \square

Korollar 8.21. *Es gibt einen $O(n^3)$ -Algorithmus für das MAXIMUM-FLOW-PROBLEM.*

Beweis: Man benutze Satz 8.20 um Schritt ③ des DINIC' ALGORITHMUS zu implementieren. \square

Weitere Verbesserungen stammen von Cherkassky [1977], Galil [1980], Galil und Namaad [1980], Shiloach [1978], Sleator [1980] und Sleator und Tarjan [1983]. Die zwei zuletzt genannten Arbeiten beschreiben einen $O(m \log n)$ -Algorithmus zur Bestimmung blockierender Flüsse in einem azyklischen Netzwerk mittels einer „dynamic trees“ genannten Datenstruktur. Benutzt man diese als Subroutine in DINIC' ALGORITHMUS, so erhält man einen $O(mn \log n)$ -Algorithmus für das MAXIMUM-FLOW-PROBLEM. Wir werden jedoch keine dieser Algorithmen näher beschreiben (siehe Tarjan [1983]), da wir im nächsten Abschnitt einen noch schnelleren Netzwerkflussalgorithmus behandeln werden.

Wir schließen diesen Abschnitt mit einer Beschreibung des schwach polynomischen Algorithmus von Fujishige [2003], hauptsächlich wegen seiner Einfachheit:

FUJISHIGES ALGORITHMUS

Input: Ein Netzwerk (G, u, s, t) mit $u : E(G) \rightarrow \mathbb{Z}_+$.

Output: Ein $s-t$ -Fluss f mit maximalem Wert.

- ① Setze $f(e) := 0$ für alle $e \in E(G)$. Setze $\alpha := \max\{u(e) : e \in E(G)\}$.
- ② Setze $i := 1$, $v_1 := s$, $X := \emptyset$ und $b(v) := 0$ für alle $v \in V(G)$.
- ③ **For** $e = (v_i, w) \in \delta_{G_f}^+(v_i)$ mit $w \notin \{v_1, \dots, v_i\}$ **do**:
Setze $b(w) := b(w) + u_f(e)$. **If** $b(w) \geq \alpha$ **then** setze $X := X \cup \{w\}$.
- ④ **If** $X = \emptyset$ **then**:
Setze $\alpha := \lfloor \frac{\alpha}{2} \rfloor$. **If** $\alpha = 0$ **then stop** **else go to** ②.
- ⑤ Setze $i := i + 1$. Wähle $v_i \in X$ und setze $X := X \setminus \{v_i\}$.
If $v_i \neq t$ **then go to** ③.
- ⑥ Setze $\beta(t) := \alpha$ und $\beta(v) := 0$ für alle $v \in V(G) \setminus \{t\}$.
While $i > 1$ **do**:
For $e = (p, v_i) \in \delta_{G_f}^-(v_i)$ mit $p \in \{v_1, \dots, v_{i-1}\}$ **do**:
Setze $\beta' := \min\{\beta(v_i), u_f(e)\}$.
Augmentiere f entlang e um β' .
Setze $\beta(v_i) := \beta(v_i) - \beta'$ und $\beta(p) := \beta(p) + \beta'$.
Setze $i := i - 1$.
- ⑦ **Go to** ②.

Satz 8.22. FUJISHIGES ALGORITHMUS löst das MAXIMUM-FLOW-PROBLEM für einfache Digraphen G und ganzzahlige Kapazitäten $u : E(G) \rightarrow \mathbb{Z}_+$ korrekt in $O(mn \log u_{\max})$ -Zeit, wobei $n := |V(G)|$, $m := |E(G)|$ und $u_{\max} := \max\{u(e) : e \in E(G)\}$.

Beweis: Wir nennen eine mit ④ oder ⑦ endende Folge von Schritten eine Iteration. In ②–⑤ ist v_1, \dots, v_i immer eine in gewisser Weise angeordnete Knotenmenge mit $b(v_j) = u_f(E^+(\{v_1, \dots, v_{j-1}\}, \{v_j\})) \geq \alpha$ für $j = 2, \dots, i$. In ⑥ wird der Fluss f um die invariante Größe $\sum_{v \in V(G)} \beta(v) = \alpha$ augmentiert, und nach obigem erhalten wir somit einen s - t -Fluss mit einem um α Einheiten größeren Wert.

Also wird α nach höchstens $n - 1$ Iterationen zum ersten Mal verringert. Wenn wir α in ④ auf $\alpha' = \lfloor \frac{\alpha}{2} \rfloor \geq \frac{\alpha}{3}$ verringern, haben wir einen s - t -Schnit $\delta_{G_f}^+(\{v_1, \dots, v_i\})$ in G_f mit Kapazität kleiner als $\alpha(|V(G)| - i)$, da $b(v) = u_f(E^+(\{v_1, \dots, v_i\}, \{v\})) < \alpha$ für alle $v \in V(G) \setminus \{v_1, \dots, v_i\}$. Nach Lemma 8.3(b) ist der Wert eines maximalen s - t -Flusses in G_f kleiner als $\alpha(n - i) < 3\alpha'n$. Damit wird α nach weniger als $3n$ Iterationen nochmals verringert. Wird α von 1 auf 0 verringert, so haben wir einen s - t -Schnit mit Kapazität 0 in G_f , also ist f maximal.

Da α höchstens $1 + \log u_{\max}$ mal verringert wird, bis es 0 erreicht, und da jede Iteration $O(m)$ -Zeit benötigt, haben wir eine $O(mn \log u_{\max})$ Gesamtlaufzeit. \square

Eine solche Skalierungstechnik ist oft sehr nützlich und wird wieder in Kapitel 9 erscheinen. Fujishige [2003] hat auch eine Variante seines Algorithmus ohne Skalierung beschrieben, wo v_i in ⑤ als ein $\max\{b(v) : v \in V(G) \setminus \{v_1, \dots, v_{i-1}\}\}$ erreichender Knoten gewählt wird. Die resultierende Reihenfolge heißt MA-Reihenfolge und wird wieder in Abschnitt 8.7 erscheinen. Die Laufzeit dieser Variante ist etwas höher als die obige und ist auch nicht streng polynomiell (Shioura [2004]). Siehe Aufgabe 25.

8.5 Der Goldberg-Tarjan-Algorithmus

In diesem Abschnitt werden wir den von Goldberg und Tarjan [1988] stammenden PUSH-RELABEL-ALGORITHMUS beschreiben und zeigen, dass er eine $O(n^2\sqrt{m})$ -Laufzeit hat.

Ausgeklügelte Implementierungen mit „dynamic trees“ (siehe Sleator und Tarjan [1983]) ergeben Netzwerkflussalgorithmen mit $O\left(nm \log \frac{n^2}{m}\right)$ -Laufzeit (Goldberg und Tarjan [1988]) und $O\left(nm \log \left(\frac{n}{m} \sqrt{\log u_{\max}} + 2\right)\right)$ -Laufzeit, wobei u_{\max} die maximale (ganzzahlige) Kantenkapazität ist (Ahuja, Orlin und Tarjan [1989]). Die bis jetzt besten bekannten Schranken sind $O(nm)$ (Orlin [2013]),

$$O\left(\min\{m^{1/2}, n^{2/3}\}m \log\left(\frac{n^2}{m}\right) \log u_{\max}\right)$$

(Goldberg und Rao [1998]) und $O(m\sqrt{n} \log^{O(1)}(n) \log^2 u_{\max})$ (Lee und Sidford [2014]).

Nach Definition und Satz 8.5 hat ein s - t -Fluss f genau dann maximalen Wert, wenn die folgenden beiden Bedingungen erfüllt sind:

- $\text{ex}_f(v) = 0$ für alle $v \in V(G) \setminus \{s, t\}$;
- Es gibt keinen f -augmentierenden Weg.

Für die bisher besprochenen Algorithmen ist die erste Bedingung stets erfüllt, und die Algorithmen terminieren wenn die zweite Bedingung erfüllt ist. Der PUSH-RELABEL-ALGORITHMUS beginnt mit einem die zweite Bedingung erfüllenden Fluss f und hält laufend ein solches f bereit. Natürlich terminiert er, wenn die erste Bedingung auch erfüllt ist. Somit wird f während des Algorithmus kein s - t -Fluss sein (außer bei Terminierung), sondern ein s - t -Präfluss (siehe Definition 8.19).

Definition 8.23. Sei (G, u, s, t) ein Netzwerk und f ein s - t -Präfluss. Eine **Distanzmarkierung** ist eine Funktion $\psi : V(G) \rightarrow \mathbb{Z}_+$ mit $\psi(t) = 0$, $\psi(s) = n := |V(G)|$ und $\psi(v) \leq \psi(w) + 1$ für alle $(v, w) \in E(G_f)$. Eine Kante $e = (v, w) \in E(G)$ heißt **erlaubte Kante**, falls $e \in E(G_f)$ und $\psi(v) = \psi(w) + 1$.

Ist ψ eine Distanzmarkierung, so ist $\psi(v)$ (mit $v \neq s$) eine untere Schranke für die Distanz zu t (die Kantenanzahl eines kürzesten v - t -Weges) in G_f .

Der weiter unten beschriebene PUSH-RELABEL-ALGORITHMUS arbeitet immer mit einem s - t -Präfluss f und einer Distanzmarkierung ψ . Er beginnt mit dem Präfluss, der auf jeder in s beginnenden Kante gleich der Kapazität dieser Kante ist und 0 auf allen anderen Kanten. Die erste Distanzmarkierung ist $\psi(s) = n$ und $\psi(v) = 0$ für alle $v \in V(G) \setminus \{s\}$.

Danach führt der Algorithmus die Update-Operationen PUSH (Update von f) und RELABEL (Update von ψ) in beliebiger Reihenfolge aus.

PUSH-RELABEL-ALGORITHMUS

Input: Ein Netzwerk (G, u, s, t) .

Output: Ein maximaler s - t -Fluss f .

- ① Setze $f(e) := u(e)$ für jedes $e \in \delta^+(s)$.
Setze $f(e) := 0$ für jedes $e \in E(G) \setminus \delta^+(s)$.
- ② Setze $\psi(s) := n := |V(G)|$ und $\psi(v) := 0$ für alle $v \in V(G) \setminus \{s\}$.
- ③ **While** es gibt einen aktiven Knoten **do**:
 - Sei v ein aktiver Knoten.
 - If** kein $e \in \delta_{G_f}^+(v)$ ist erlaubte Kante
then RELABEL(v),
else sei $e \in \delta_{G_f}^+(v)$ eine erlaubte Kante und PUSH(e).

PUSH(e)

- ① Setze $\gamma := \min\{\text{ex}_f(v), u_f(e)\}$, wobei e in v beginnt.
- ② Augmentiere f entlang e um γ .

RELABEL(v)

- ① Setze $\psi(v) := \min\{\psi(w) + 1 : (v, w) \in \delta_{G_f}^+(v)\}$.

Proposition 8.24. Während des gesamten Ablaufs des PUSH-RELABEL-ALGORITHMUS ist f stets ein s - t -Präfluss und ψ stets eine Distanzmarkierung bezüglich f . Für jedes $v \in V(G)$ gilt: $\psi(v)$ wird durch jedes RELABEL(v) streng erhöht.

Beweis: Wir müssen zeigen, dass die Operationen PUSH und RELABEL diese beiden Eigenschaften nicht zerstören. Klar ist, dass f nach einer PUSH-Operation wieder ein s - t -Präfluss ist. Eine RELABEL-Operation ändert f nicht einmal.

Wird RELABEL(v) aufgerufen und war ψ vorher eine Distanzmarkierung, so wird $\psi(v)$ streng erhöht (da kein $e \in \delta_{G_f}^+(v)$ zulässig war), und ψ bleibt eine Distanzmarkierung.

Schließlich zeigen wir, dass ψ nach einer PUSH-Operation wieder eine Distanzmarkierung bezüglich des neuen Präflusses ist. Wir müssen zeigen, dass $\psi(a) \leq \psi(b) + 1$ für alle neuen Kanten (a, b) in G_f . Wenden wir aber PUSH(e) auf eine Kante $e = (v, w)$ an, so ist die einzige mögliche neue Kante in G_f die gegenläufige Kante von e und hier gilt $\psi(w) = \psi(v) - 1$, da e eine erlaubte Kante ist. \square

Lemma 8.25. Sei f ein s - t -Präfluss und ψ eine Distanzmarkierung bezüglich f . Dann gelten die drei folgenden Aussagen:

- (a) Es ist s von jedem aktiven Knoten v aus in G_f erreichbar.
- (b) Gibt es zwei Knoten $v, w \in V(G)$ mit der Eigenschaft, dass w von v aus in G_f erreichbar ist, so gilt $\psi(v) \leq \psi(w) + n - 1$.
- (c) Es ist t nicht von s aus in G_f erreichbar.

Beweis: (a): Sei v ein aktiver Knoten und R die Menge der von v aus in G_f erreichbaren Knoten. Dann ist $f(e) = 0$ für alle $e \in \delta_G^-(R)$ und es gilt

$$\sum_{w \in R} \text{ex}_f(w) = \sum_{e \in \delta_G^-(R)} f(e) - \sum_{e \in \delta_G^+(R)} f(e) \leq 0.$$

Aber v ist aktiv, also gilt $\text{ex}_f(v) > 0$ und somit gibt es einen Knoten $w \in R$ mit $\text{ex}_f(w) < 0$. Da f ein s - t -Präfluss ist, muss dieser Knoten s sein.

(b): Angenommen, es liegt ein v - w -Weg in G_f vor, etwa mit den Knoten $v = v_0, v_1, \dots, v_k = w$. Da ψ eine Distanzmarkierung bezüglich f ist, gilt $\psi(v_i) \leq \psi(v_{i+1}) + 1$ für $i = 0, \dots, k - 1$. Somit gilt $\psi(v) \leq \psi(w) + k$. Beachte, dass $k \leq n - 1$.

(c): Diese Aussage folgt aus (b), da $\psi(s) = n$ und $\psi(t) = 0$. \square

Mit Aussage (c) können wir den folgenden Satz beweisen:

Satz 8.26. Bei Terminierung des Algorithmus ist f ein s - t -Fluss mit maximalem Wert.

Beweis: Es ist f ein s - t -Fluss, da es keine aktiven Knoten gibt. Nach Lemma 8.25(c) gibt es keinen augmentierenden Weg. Mit Satz 8.5 folgt dann, dass f maximal ist. \square

Die Frage ist nun, wie viele PUSH- und RELABEL-Operationen stattfinden.

Lemma 8.27.

- (a) Für jedes $v \in V(G)$ gilt: $\psi(v)$ wird niemals verringert und $\psi(v) \leq 2n - 1$ zu jedem Zeitpunkt des Algorithmus.
- (b) Kein Knoten wird mehr als $(2n - 1)$ mal geRELABELT. Die Gesamterhöhung von $\sum_{v \in V(G)} \psi(v)$ während des Ablaufs des Algorithmus ist höchstens gleich $2n^2 - n$.

Beweis: Nach Proposition 8.24 folgt, dass $\psi(v)$ durch jedes RELABEL(v) streng erhöht wird. Ferner ändern wir $\psi(v)$ durch RELABEL(v) nur dann, wenn v ein aktiver Knoten ist. Nach Lemma 8.25(a) und (b) haben wir $\psi(v) \leq \psi(s) + n - 1 = 2n - 1$. Hieraus folgen (a) und (b). \square

Nun werden wir die Anzahl der PUSH-Operationen untersuchen. Dabei unterscheiden wir zwischen einem **saturierenden Push** (wo $u_f(e) = 0$ nach dem Push) und einem **nichtsaturierenden Push**. Wie üblich, setzen wir $m := |E(G)|$ (und $n := |V(G)|$).

Lemma 8.28. Die Anzahl der saturierenden Pushes ist höchstens $2mn$.

Beweis: Nach jedem saturierenden Push von v nach w kann kein weiterer solcher Push stattfinden, bis $\psi(w)$ um mindestens 2 gewachsen ist, ein Push von w nach v stattgefunden hat und $\psi(v)$ um mindestens 2 gewachsen ist. Zusammen mit Lemma 8.27(a) folgt daraus, dass es höchstens n saturierende Pushes auf jeder Kante $\overset{\leftrightarrow}{(v, w)} \in E(\overleftrightarrow{G})$ gibt. \square

Die Anzahl der nichtsaturierenden Pushes kann im Allgemeinen die Ordnung n^2m haben (Aufgabe 26). Wählen wir in ③ einen aktiven Knoten v mit $\psi(v)$ maximal, so können wir die Schranke verbessern. Wir können annehmen, dass $n \leq m \leq n^2$.

Lemma 8.29. Wählen wir als v in ③ des PUSH-RELABEL-ALGORITHMUS immer einen aktiven Knoten mit $\psi(v)$ maximal, so ist die Anzahl der nichtsaturierenden Pushes höchstens $8n^2\sqrt{m}$.

Beweis: Wir nennen die Laufzeit zwischen zwei aufeinander folgenden Änderungen von $\psi^* := \max\{\psi(v) : v \text{ aktiv}\}$ eine Phase. Da ψ^* nur durch eine RELABEL-Operation zunehmen kann, ist die Gesamtzunahme von ψ^* geringer als $2n^2$. Da zu Beginn $\psi^* = 0$ war, wird es weniger als $2n^2$ mal verringert, also ist die Anzahl der Phasen kleiner als $4n^2$.

Es sei eine Phase billig, falls sie höchstens \sqrt{m} nichtsaturierende Pushes enthält, sonst teuer. Offensichtlich gibt es höchstens $4n^2\sqrt{m}$ nichtsaturierende Pushes in billigen Phasen. Sei

$$\Phi := \sum_{v \in V(G): v \text{ aktiv}} |\{w \in V(G) : \psi(w) \leq \psi(v)\}|.$$

Zu Beginn gilt $\Phi \leq n^2$. Eine RELABEL-Operation kann Φ um höchstens n erhöhen. Ein saturierender Push kann Φ um höchstens n erhöhen. Ein nichtsaturierender Push kann Φ nicht erhöhen. Da wir am Ende $\Phi = 0$ haben, ist die Gesamtannahme von Φ höchstens $n^2 + n(2n^2 - n) + n(2mn) \leq 4mn^2$.

Nun betrachten wir die nichtsaturierenden Pushes in einer teuren Phase. Jeder dieser Pushes schiebt einen Fluss entlang einer Kante (v, w) mit $\psi(v) = \psi^* = \psi(w) + 1$. Damit wird v deaktiviert und w möglicherweise aktiv.

Da die Phase dadurch endet, dass der letzte aktive Knoten v mit $\psi(v) = \psi^*$ deaktiviert wird oder dass geRELABELT wird, bleibt die Menge der Knoten w mit $\psi(w) = \psi^*$ während der Phase konstant, und sie enthält mehr als \sqrt{m} Knoten, da die Phase teuer ist. Also wird Φ durch jeden nichtsaturierenden Push in einer teuren Phase um mindestens \sqrt{m} verringert. Somit ist die Gesamtanzahl der nichtsaturierenden Pushes in teuren Phasen höchstens $\frac{4mn^2}{\sqrt{m}} = 4n^2\sqrt{m}$. \square

Dieser Beweis stammt von Cheriyan und Mehlhorn [1999]. Nun bekommen wir endlich den

Satz 8.30. (Goldberg und Tarjan [1988], Cheriyan und Maheshwari [1989], Tunçel [1994]) *Der PUSH-RELABEL-ALGORITHMUS löst das MAXIMUM-FLOW-PROBLEM korrekt und kann mit $O(n^2\sqrt{m})$ -Laufzeit implementiert werden.*

Beweis: Die Korrektheit folgt aus Satz 8.26.

Wie in Lemma 8.29 wählen wir für v in ③ immer einen aktiven Knoten mit maximalem $\psi(v)$. Um dies zu erleichtern, legen wir doppelt verbundene Listen L_0, \dots, L_{2n-1} an, wobei L_i die aktiven Knoten v mit $\psi(v) = i$ enthält. Diese Listen können bei jeder PUSH- und RELABEL-Operation in konstanter Zeit aktualisiert werden.

Wir beginnen, indem wir L_i für $i = 0$ scannen. Wird ein Knoten geRELABELT, so erhöhen wir i entsprechend. Finden wir eine leere Liste L_i für das laufende i (nach Deaktivierung des letzten aktiven Knotens zu dem Zeitpunkt), so verringern wir i bis L_i nicht mehr leer ist. Da wir i nach Lemma 8.27(b) höchstens $2n^2$ mal erhöhen, verringern wir i auch höchstens $2n^2$ mal.

Als zweite Datenstruktur legen wir für jeden Knoten v eine doppelt verbundene Liste A_v der in v beginnenden erlaubten Kanten an. Diese können bei jeder PUSH-Operation in konstanter Zeit aktualisiert werden und bei jeder RELABEL-Operation in Zeit proportional zur Gesamtanzahl der mit dem geRELABELten Knoten inzidenten Kanten.

Somit benötigt $\text{RELABEL}(v)$ insgesamt $O(|\delta_G(v)|)$ -Zeit und nach Lemma 8.27(b) benötigt RELABEL insgesamt $O(mn)$ -Zeit. Jedes PUSH benötigt konstante Zeit und nach Lemma 8.28 und Lemma 8.29 ist die Gesamtanzahl der Pushes $O(n^2\sqrt{m})$. \square

8.6 Gomory-Hu-Bäume

Jeder Algorithmus für das MAXIMUM-FLOW-PROBLEM liefert auch eine Lösung des folgenden Problems:

MINIMUM-CAPACITY-CUT-PROBLEM

Instanz: Ein Netzwerk (G, u, s, t) .

Aufgabe: Ein $s-t$ -Schnitt in G minimaler Kapazität.

Proposition 8.31. Das MINIMUM-CAPACITY-CUT-PROBLEM kann mit derselben Laufzeit wie das MAXIMUM-FLOW-PROBLEM gelöst werden, insbesondere mit $O(n^2\sqrt{m})$ Laufzeit.

Beweis: Wir berechnen einen $s-t$ -Fluss f mit maximalem Wert für ein Netzwerk (G, u, s, t) und setzen X gleich der Menge aller von s aus in G_f erreichbaren Knoten. Die Menge X kann mit dem GRAPH-SCANNING-ALGORITHMUS in linearer Zeit berechnet werden (Proposition 2.17). Nach Lemma 8.3 und Satz 8.5 ist $\delta_G^+(X)$ ein $s-t$ -Schnitt minimaler Kapazität. Die $O(n^2\sqrt{m})$ -Laufzeit folgt mit Satz 8.30 (und ist nicht die bestmögliche). \square

In diesem Abschnitt betrachten wir das Problem, für jedes Knotenpaar s, t in einem ungerichteten Graphen G mit den Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ einen $s-t$ -Schnitt minimaler Kapazität zu finden.

Dieses Problem kann auf das obige zurückgeführt werden: Für alle Knotenpaare $s, t \in V(G)$ löse man das MINIMUM-CAPACITY-CUT-PROBLEM für (G', u', s, t) , wobei (G', u') aus (G, u) dadurch hervorgeht, dass man jede ungerichtete Kante $\{v, w\}$ durch zwei gegenläufig gerichtete Kanten (v, w) und (w, v) mit $u'((v, w)) = u'((w, v)) = u(\{v, w\})$ ersetzt. Auf diese Weise erhalten wir $s-t$ -Schnitte minimaler Kapazität für alle s, t nach $\binom{n}{2}$ Flussberechnungen.

Diesen Abschnitt widmen wir dem eleganten Verfahren von Gomory und Hu [1961], welches nur $n - 1$ Flussberechnungen benötigt. Die Abschnitte 12.3 und 20.4 enthalten Anwendungen dieses Verfahrens.

Definition 8.32. Sei G ein ungerichteter Graph und $u : E(G) \rightarrow \mathbb{R}_+$ eine Kapazitätsfunktion. Für je zwei Knoten $s, t \in V(G)$ sei λ_{st} deren **lokaler Kantenzusammenhang**, d. h. die minimale Kapazität eines s und t trennenden Schnittes.

Der Kantenzusammenhang eines Graphen ist offensichtlich der minimale lokale Kantenzusammenhang bezüglich Einheitskapazitäten.

Lemma 8.33. Für je drei Knoten $i, j, k \in V(G)$ gilt $\lambda_{ik} \geq \min\{\lambda_{ij}, \lambda_{jk}\}$.

Beweis: Sei $\delta(A)$ ein Schnitt mit $i \in A, k \in V(G) \setminus A$ und $u(\delta(A)) = \lambda_{ik}$. Ist $j \in A$, dann trennt $\delta(A)$ die Knoten j und k , somit gilt $u(\delta(A)) \geq \lambda_{jk}$. Ist $j \in V(G) \setminus A$, dann trennt $\delta(A)$ die Knoten i und j , somit gilt $u(\delta(A)) \geq \lambda_{ij}$. Damit folgt $\lambda_{ik} = u(\delta(A)) \geq \min\{\lambda_{ij}, \lambda_{jk}\}$. \square

Diese Bedingung ist nicht nur notwendig: Für den Fall, dass die Zahlen $(\lambda_{ij})_{1 \leq i, j \leq n}$ zusätzlich $\lambda_{ij} = \lambda_{ji}$ erfüllen, ist sie dafür hinreichend, dass es einen Graphen gibt, dessen lokale Kantenzusammenhänge genau diese Zahlen sind (Aufgabe 32).

Definition 8.34. Sei G ein ungerichteter Graph und $u : E(G) \rightarrow \mathbb{R}_+$ eine Kapazitätsfunktion. Ein Baum T heißt **Gomory-Hu-Baum** für (G, u) , falls $V(T) = V(G)$ und

$$\lambda_{st} = \min_{e \in E(P_{st})} u(\delta_G(C_e)) \quad \text{für alle } s, t \in V(G),$$

wobei P_{st} der (einzig bestimme) s - t -Weg in T ist und C_e und $V(G) \setminus C_e$ für $e \in E(T)$ die Zusammenhangskomponenten von $T - e$ sind (d. h. $\delta_G(C_e)$ ist der Fundamentalschnitt von e bezüglich T).

Wie wir noch sehen werden, besitzt jeder ungerichtete Graph einen Gomory-Hu-Baum. Daraus folgt, dass es für jeden ungerichteten Graphen G eine Liste von $n - 1$ Schnitten gibt, mit der Eigenschaft, dass die Liste für jedes Knotenpaar $s, t \in V(G)$ einen s - t -Schnitt minimaler Kapazität enthält.

Dies gilt nicht für Digraphen: Für jedes $n \in \mathbb{N}$ haben Jelinek und Mayeda [1963] einen Digraphen G mit n Knoten und Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ konstruiert, mit der Eigenschaft, dass die Menge $\{\min\{u(\delta^+(X)) : s \in X \subseteq V(G) \setminus \{t\}\} : s, t \in V(G), s \neq t\}$ $(n + 2)(n - 1)/2$ verschiedene Zahlen enthält.

Im Allgemeinen kann ein Gomory-Hu-Baum nicht als Teilgraph von G gewählt werden. Betrachten wir das Beispiel $G = K_{3,3}$ und $u \equiv 1$, so ist $\lambda_{st} = 3$ für alle $s, t \in V(G)$ und es ist leicht zu sehen, dass die Gomory-Hu-Bäume für (G, u) genau die Sterne mit fünf Kanten sind.

Die grundlegende Idee des Algorithmus zur Bestimmung eines Gomory-Hu-Baumes ist die folgende. Zunächst wählen wir ein Knotenpaar $s, t \in V(G)$ und bestimmen einen s - t -Schnitt minimaler Kapazität, etwa $\delta(A)$. Sei $B := V(G) \setminus A$. Dann kontrahieren wir A (oder B) zu einem einzigen Knoten, wählen ein weiteres Knotenpaar $s', t' \in B$ (bzw. $s', t' \in A$) und bestimmen einen s' - t' -Schnitt minimaler Kapazität in dem kontrahierten Graphen G' . So fahren wir fort, indem wir stets ein nicht von einem der bereits bestimmten Schnitte getrenntes Knotenpaar s', t' wählen. Bei jedem Schritt kontrahieren wir – für jeden bereits bestimmten Schnitt $E(A', B') - A'$ oder B' , je nachdem welcher Teil s' und t' nicht enthält.

Irgendwann sind dann alle Knotenpaare getrennt. Wir haben insgesamt $n - 1$ Schnitte bestimmt. Der entscheidende Punkt ist nun, dass ein s - t -Schnitt minimaler Kapazität in dem kontrahierten Graphen G' auch ein s - t -Schnitt minimaler Kapazität in G ist. Dies wird in dem folgenden Lemma präzisiert. Beachte, dass bei der Kontraktion der Knotenmenge A in (G, u) die Kapazität jeder Kante in G' gleich der Kapazität der entsprechenden Kante in G ist.

Lemma 8.35. Sei G ein ungerichteter Graph und $u : E(G) \rightarrow \mathbb{R}_+$ eine Kapazitätsfunktion. Seien $s, t \in V(G)$ und $\delta(A)$ ein s - t -Schnitt minimaler Kapazität in (G, u) . Angenommen, es gehe (G', u') aus (G, u) durch die Kontraktion von A zu

einem einzigen Knoten hervor. Seien $s', t' \in V(G) \setminus A$. Dann gilt: Für jeden $s'-t'$ -Schnitt minimaler Kapazität $\delta(K \cup \{A\})$ in (G', u') ist $\delta(K \cup A)$ ein $s'-t'$ -Schnitt minimaler Kapazität in (G, u) .

Beweis: Seien s, t, A, s', t', G', u' wie oben. O. B. d. A. können wir annehmen, dass $s \in A$. Es genügt zu beweisen, dass es einen $s'-t'$ -Schnitt minimaler Kapazität $\delta(A')$ in (G, u) mit $A \subset A'$ gibt. Also sei $\delta(C)$ ein $s'-t'$ -Schnitt minimaler Kapazität in (G, u) . O. B. d. A. können wir annehmen, dass $s \in C$.

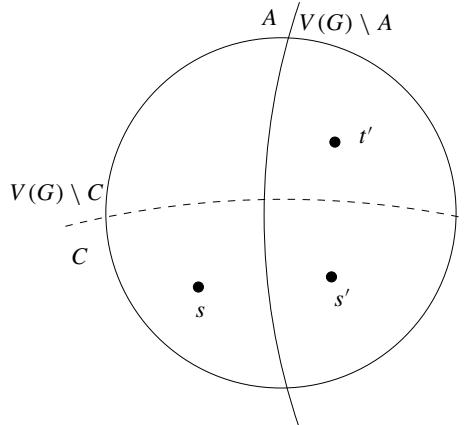


Abbildung 8.3.

Da $u(\delta(\cdot))$ submodular ist (siehe Lemma 2.1(c)), folgt $u(\delta(A)) + u(\delta(C)) \geq u(\delta(A \cap C)) + u(\delta(A \cup C))$. Aber $\delta(A \cap C)$ ist ein $s-t$ -Schnitt, somit gilt $u(\delta(A \cap C)) \geq \lambda_{st} = u(\delta(A))$. Daraus folgt $u(\delta(A \cup C)) \leq u(\delta(C)) = \lambda_{s't'}$, womit wir bewiesen haben, dass $\delta(A \cup C)$ ein $s'-t'$ -Schnitt minimaler Kapazität ist. (Siehe Abb. 8.3.) \square

Nun beschreiben wir den Algorithmus zur Konstruktion eines Gomory-Hu-Baumes. Beachte, dass die Knoten der zwischenzeitlichen Bäume T Knotenmengen des ursprünglichen Graphen sind; sie bilden in der Tat eine Partition von $V(G)$. Zu Beginn bildet $V(G)$ den einzigen Knoten von T . In jeder Iteration wird ein mindestens zwei Knoten von G enthaltender Knoten von T gewählt und zweigespalten.

GOMORY-HU-ALGORITHMUS

Input: Ein ungerichteter Graph G und eine Kapazitätsfunktion $u : E(G) \rightarrow \mathbb{R}_+$.
Output: Ein Gomory-Hu-Baum T für (G, u) .

- ① Setze $V(T) := \{V(G)\}$ und $E(T) := \emptyset$.
 - ② Wähle ein $X \in V(T)$ mit $|X| \geq 2$. If es gibt kein solches X then go to ⑥.
 - ③ Wähle $s, t \in X$ mit $s \neq t$.
For jede Zusammenhangskomponente C von $T - X$ **do**: Sei
 $S_C := \bigcup_{Y \in V(C)} Y$.
Es gehe (G', u') aus (G, u) hervor durch Kontraktion von S_C zu einem einzigen Knoten v_C für jede Zusammenhangskomponente C von $T - X$.
(Also ist $V(G') = X \cup \{v_C : C \text{ Zusammenhangskomponente von } T - X\}$.)
 - ④ Bestimme einen $s-t$ -Schnitt minimaler Kapazität $\delta(A')$ in (G', u') . Sei
 $B' := V(G') \setminus A'$.
Setze $A := \left(\bigcup_{v_C \in A' \setminus X} S_C \right) \cup (A' \cap X)$ und $B := \left(\bigcup_{v_C \in B' \setminus X} S_C \right) \cup (B' \cap X)$.
 - ⑤ Setze $V(T) := (V(T) \setminus \{X\}) \cup \{A \cap X, B \cap X\}$.
For jede mit dem Knoten X inzidente Kante $e = \{X, Y\} \in E(T)$ **do**:
If $Y \subseteq A$ **then** setze $e' := \{A \cap X, Y\}$ **else** setze $e' := \{B \cap X, Y\}$.
Setze $E(T) := (E(T) \setminus \{e\}) \cup \{e'\}$ und $w(e') := w(e)$.
Setze $E(T) := E(T) \cup \{A \cap X, B \cap X\}$.
Setze $w(\{A \cap X, B \cap X\}) := u'(\delta_{G'}(A'))$.
Go to ②.
 - ⑥ Ersetze alle $\{x\} \in V(T)$ durch x und alle $\{\{x\}, \{y\}\} \in E(T)$ durch $\{x, y\}$.
Stop.
-

Abbildung 8.4 zeigt die Modifizierung von T in ⑤. Um die Korrektheit dieses Algorithmus zu beweisen, brauchen wir das folgende Lemma:

Lemma 8.36. *Jedes Mal am Ende von ④ haben wir*

- (a) $A \dot{\cup} B = V(G)$
- (b) $E(A, B)$ ist ein $s-t$ -Schnitt minimaler Kapazität in (G, u) .

Beweis: Die Elemente von $V(T)$ sind immer nichtleere Teilmengen von $V(G)$, in der Tat liefert $V(T)$ eine Partition von $V(G)$. Daraus folgt (a) leicht.

Zum Beweis von (b) bemerken wir zunächst, dass die Aussage für die erste Iteration trivial ist (da hier $G' = G$ ist). Wir zeigen nun, dass die Aussage bei jeder Iteration erhalten bleibt.

Seien C_1, \dots, C_k die Zusammenhangskomponenten von $T - X$. Wir werden diese nun eine nach der anderen kontrahieren; es gehe (G_i, u_i) für $i = 0, \dots, k$ aus (G, u) hervor durch Kontraktion jedes der S_{C_1}, \dots, S_{C_i} zu einem einzigen Knoten. Somit ist (G_k, u_k) der mit (G', u') in ③ des Algorithmus bezeichnete Graph.

Behauptung: Für jeden $s-t$ -Schnitt minimaler Kapazität $\delta(A_i)$ in (G_i, u_i) ist $\delta(A_{i-1})$ ein $s-t$ -Schnitt minimaler Kapazität in (G_{i-1}, u_{i-1}) , wobei

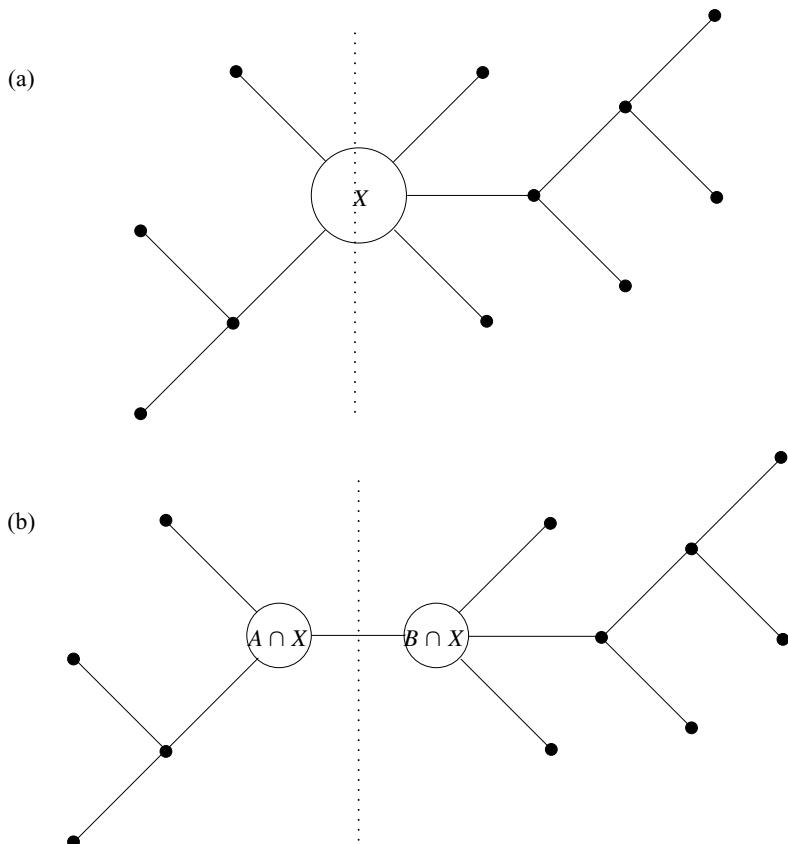


Abbildung 8.4.

$$A_{i-1} := \begin{cases} (A_i \setminus \{v_{C_i}\}) \cup S_{C_i} & \text{für } v_{C_i} \in A_i \\ A_i & \text{für } v_{C_i} \notin A_i \end{cases}.$$

Die schrittweise Anwendung dieser Behauptung für $k, k-1, \dots, 1$ ergibt (b).

Zum Beweis der Behauptung sei $\delta(A_i)$ ein s - t -Schnitt minimaler Kapazität in (G_i, u_i) . Wegen der Annahme, dass (b) für die früheren Iterationen gilt, ist $\delta(S_{C_i})$ ein s_i - t_i -Schnitt minimaler Kapazität in (G, u) für geeignete $s_i, t_i \in V(G)$. Auch sind $s, t \in V(G) \setminus S_{C_i}$. Mit Lemma 8.35 folgt der Beweis. \square

Lemma 8.37. Zu jedem Zeitpunkt des Algorithmus (bis ⑥ erreicht wird) gilt für alle $e \in E(T)$

$$w(e) = u\left(\delta_G\left(\bigcup_{Z \in C_e} Z\right)\right),$$

wobei C_e und $V(T) \setminus C_e$ die Zusammenhangskomponenten von $T - e$ sind. Ferner gibt es für alle $e = \{P, Q\} \in E(T)$ Knoten $p \in P$ und $q \in Q$ mit $\lambda_{pq} = w(e)$.

Beweis: Beide Aussagen sind trivial am Anfang des Algorithmus, wo T keine Kanten enthält; wir zeigen nun, dass sie immer gelten. Sei also X der in ② von irgendeiner Iteration gewählte Knoten von T . Seien ferner s, t, A', B', A, B die in den anschließenden Schritten ③ und ④ bestimmten Größen. O. B. d. A. können wir annehmen, dass $s \in A'$.

Kanten, die nicht mit X incident sind, bleiben in ⑤ unberührt. Für die neue Kante $\{A \cap X, B \cap X\}$ ist $w(e)$ offensichtlich korrekt gesetzt, und es folgt $\lambda_{st} = w(e)$, $s \in A \cap X$ und $t \in B \cap X$.

Also betrachten wir nun eine durch e' in ⑤ ersetzte Kante $e = \{X, Y\}$. O. B. d. A. können wir annehmen, dass $Y \subseteq A$, somit ist $e' = \{A \cap X, Y\}$. Unter der Annahme, dass die Aussagen für e gelten, behaupten wir, dass sie auch für e' gelten. Für die erste Aussage ist dies trivial, da $w(e) = w(e')$ ist und $u(\delta_G(\bigcup_{Z \in C_e} Z))$ unverändert bleibt.

Zum Beweis der zweiten Aussage nehmen wir an, dass es $p \in X$ und $q \in Y$ mit $\lambda_{pq} = w(e)$ gibt. Ist $p \in A \cap X$, so sind wir fertig. Also sei von nun an $p \in B \cap X$ (siehe Abb. 8.5).

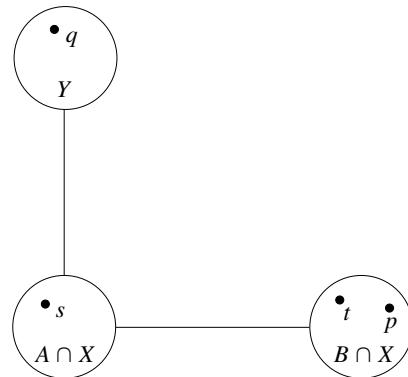


Abbildung 8.5.

Wir behaupten, dass $\lambda_{sq} = \lambda_{pq}$. Da $\lambda_{pq} = w(e) = w(e')$ und $s \in A \cap X$, folgt damit der Beweis.

Nach Lemma 8.33 ist

$$\lambda_{sq} \geq \min\{\lambda_{st}, \lambda_{tp}, \lambda_{pq}\}.$$

Da $E(A, B)$ nach Lemma 8.36(b) ein $s-t$ -Schnitt minimaler Kapazität ist und da $s, q \in A$, folgt mit Lemma 8.35, dass λ_{sq} bei der Kontraktion von B unverändert bleibt. Da $t, p \in B$, folgt hieraus, dass λ_{sq} bei dem Hinzufügen einer Kante $\{t, p\}$ mit beliebig hoher Kapazität unverändert bleibt. Somit gilt

$$\lambda_{sq} \geq \min\{\lambda_{st}, \lambda_{pq}\}.$$

Beachte nun, dass $\lambda_{st} \geq \lambda_{pq}$, da der $s-t$ -Schnitt minimaler Kapazität $E(A, B)$ auch p und q trennt. Damit haben wir

$$\lambda_{sq} \geq \lambda_{pq}.$$

Zum Beweis der Gleichheit bemerken wir, dass $w(e)$ gleich der Kapazität eines X und Y trennenden Schnittes ist, der somit auch s und q trennt. Damit haben wir

$$\lambda_{sq} \leq w(e) = \lambda_{pq}$$

und das Ende des Beweises. \square

Satz 8.38. (Gomory und Hu [1961]) *Der GOMORY-HU-ALGORITHMUS arbeitet korrekt. Jeder ungerichtete Graph besitzt einen Gomory-Hu-Baum und ein solcher Baum kann in $O(n^3\sqrt{m})$ -Zeit bestimmt werden.*

Beweis: Die Komplexität des Algorithmus wird offensichtlich durch $n - 1$ mal die Komplexität der Bestimmung eines $s-t$ -Schnittes minimaler Kapazität bestimmt, da alles andere in $O(n^3)$ -Zeit implementiert werden kann. Nach Proposition 8.31 erhalten wir somit die $O(n^3\sqrt{m})$ Schranke.

Wir beweisen nun, dass der Output T des Algorithmus ein Gomory-Hu-Baum für (G, u) ist. Es sollte klar sein, dass T ein Baum mit $V(T) = V(G)$ ist. Seien $s, t \in V(G)$ und P_{st} der (eindeutig bestimmte) $s-t$ -Weg in T . Für jedes $e \in E(T)$ seien C_e und $V(G) \setminus C_e$ die Zusammenhangskomponenten von $T - e$.

Da $\delta(C_e)$ ein $s-t$ -Schnitt für jedes $e \in E(P_{st})$ ist, folgt

$$\lambda_{st} \leq \min_{e \in E(P_{st})} u(\delta(C_e)).$$

Andererseits folgt durch die wiederholte Anwendung von Lemma 8.33, dass

$$\lambda_{st} \geq \min_{\{v, w\} \in E(P_{st})} \lambda_{vw}.$$

Wenden wir nun Lemma 8.37 auf die Situation vor Ausführung von ⑥ an (wo jeder Knoten X von T eine einelementige Knotenmenge ist), so bekommen wir

$$\lambda_{st} \geq \min_{e \in E(P_{st})} u(\delta(C_e)),$$

womit wir Gleichheit haben. \square

Ein ähnlicher Algorithmus mit demselben Ziel (der eventuell leichter zu implementieren ist) wurde von Gusfield [1990] vorgeschlagen. Für Digraphen haben Cheung, Lau und Leung [2011] gezeigt, wie man die minimale Kardinalität eines $s-t$ -Schnittes für alle Paare $s, t \in V(G)$ in $O(m^{2.38})$ -Zeit berechnen kann.

8.7 Die minimale Kapazität eines Schnittes in einem ungerichteten Graphen

Sind wir nur interessiert an einem Schnitt minimaler Kapazität in einem ungerichteten Graphen G mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$, so gibt es ein einfacheres

Verfahren mit $n - 1$ Flussberechnungen: Man braucht nur einen s - t -Schnitt minimaler Kapazität für einen bestimmten Knoten s und jedes $t \in V(G) \setminus \{s\}$ zu berechnen. Es gibt jedoch effizientere Algorithmen.

Hao und Orlin [1994] haben einen $O(nm \log \frac{n^2}{m})$ -Algorithmus zur Bestimmung eines Schnittes minimaler Kapazität gefunden. Sie benutzen eine modifizierte Version des PUSH-RELABEL-ALGORITHMUS.

Möchten wir bloß den Kantenzusammenhang eines gegebenen Graphen (d. h. mit Einheitskapazitäten) berechnen, so stammen die schnellsten bekannten Algorithmen von Henzinger, Rao und Wang [2017] und Gabow [1995] mit fast-linearen Laufzeiten. Wir bemerken noch, dass das MAXIMUM-FLOW-PROBLEM in einem ungerichteten Graphen mit Einheitskapazitäten auch schneller als der allgemeine Fall gelöst werden kann (Karger und Levine [1998]).

Nagamochi und Ibaraki [1992] haben einen vollkommen andersartigen Algorithmus zur Bestimmung eines Schnittes minimaler Kapazität in einem ungerichteten Graphen beschrieben. Deren Algorithmus berechnet überhaupt keine maximalen Flüsse. In diesem Abschnitt werden wir eine von Stoer und Wagner [1997] und unabhängig von Frank [1994] stammende vereinfachte Version dieses Algorithmus vorstellen. Wir beginnen mit einer einfachen Definition.

Definition 8.39. Gegeben sei ein ungerichteter Graph G mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$. Eine Reihenfolge v_1, \dots, v_n der Knoten heißt eine **MA-Reihenfolge (maximum adjacency order)**, falls für alle $i \in \{2, \dots, n\}$ gilt:

$$\sum_{e \in E(\{v_1, \dots, v_{i-1}\}, \{v_i\})} u(e) = \max_{j \in \{i, \dots, n\}} \sum_{e \in E(\{v_1, \dots, v_{i-1}\}, \{v_j\})} u(e).$$

Proposition 8.40. Gegeben sei ein ungerichteter Graph G mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$. Eine MA-Reihenfolge kann in $O(m + n \log n)$ -Zeit gefunden werden.

Beweis: Betrachte den folgenden Algorithmus. Setze zuerst $\alpha(v) := 0$ für alle $v \in V(G)$. Führe dann für $i := 1$ bis n Folgendes aus: Wähle v_i mit maximalem α -Wert aus $V(G) \setminus \{v_1, \dots, v_{i-1}\}$ (Tie-Breaks werden beliebig gelöst), und setze $\alpha(v) := \alpha(v) + \sum_{e \in E(\{v_i\}, \{v\})} u(e)$ für alle $v \in V(G) \setminus \{v_1, \dots, v_i\}$.

Die Korrektheit dieses Algorithmus ist offensichtlich. Wird er mittels eines Fibonacci-Heaps implementiert und wird jeder Knoten v mit Schlüssel $-\alpha(v)$ gespeichert bis er gewählt wird, so erreichen wir nach Satz 6.7 eine $O(m + n \log n)$ -Laufzeit, da es n INSERT-, n DELETEMIN- und (höchstens) m DECREASEKEY-Operationen gibt. \square

Lemma 8.41. (Stoer und Wagner [1997], Frank [1994]) Sei G ein ungerichteter Graph mit $n := |V(G)| \geq 2$, Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und einer MA-Reihenfolge v_1, \dots, v_n . Dann ist

$$\lambda_{v_{n-1}v_n} = \sum_{e \in \delta(v_n)} u(e).$$

Beweis: Es ist klar, dass wir nur „ \geq “ zu zeigen brauchen. Dies werden wir mittels Induktion über $|V(G)| + |E(G)|$ tun. Für $|V(G)| < 3$ ist die Aussage trivial. Wir können annehmen, dass es keine Kante $e = \{v_{n-1}, v_n\} \in E(G)$ gibt, denn sonst würden wir sie entfernen (wobei beide Seiten der zu beweisenden Gleichung um $u(e)$ verringert werden) und die Induktionsvoraussetzung anwenden.

Die rechte Seite bezeichnen wir mit R . Natürlich ist v_1, \dots, v_{n-1} eine MA-Reihenfolge in $G - v_n$. Mittels Induktion folgt sodann, dass

$$\lambda_{v_{n-2}v_{n-1}}^{G-v_n} = \sum_{e \in E(\{v_{n-1}\}, \{v_1, \dots, v_{n-2}\})} u(e) \geq \sum_{e \in E(\{v_n\}, \{v_1, \dots, v_{n-2}\})} u(e) = R.$$

Die Ungleichung folgt, da v_1, \dots, v_n eine MA-Reihenfolge für G war. Die anschließende Gleichung folgt, da $\{v_{n-1}, v_n\} \notin E(G)$. Somit haben wir $\lambda_{v_{n-2}v_{n-1}}^G \geq \lambda_{v_{n-2}v_{n-1}}^{G-v_n} \geq R$.

Andererseits ist v_1, \dots, v_{n-2}, v_n eine MA-Reihenfolge in $G - v_{n-1}$. Also folgt mittels Induktion, dass

$$\lambda_{v_{n-2}v_n}^{G-v_{n-1}} = \sum_{e \in E(\{v_n\}, \{v_1, \dots, v_{n-2}\})} u(e) = R,$$

wiederum da $\{v_{n-1}, v_n\} \notin E(G)$. Somit haben wir $\lambda_{v_{n-2}v_n}^G \geq \lambda_{v_{n-2}v_n}^{G-v_{n-1}} = R$.

Schließlich folgt mit Lemma 8.33, dass $\lambda_{v_{n-1}v_n} \geq \min\{\lambda_{v_{n-1}v_{n-2}}, \lambda_{v_{n-2}v_n}\} \geq R$. \square

Beachte, dass die Existenz der beiden Knoten x, y mit $\lambda_{xy} = \sum_{e \in \delta(x)} u(e)$ bereits von Mader [1972] bewiesen wurde. Sie folgt auch leicht aus der Existenz eines Gomory-Hu-Baumes (Aufgabe 34).

Satz 8.42. (Nagamochi und Ibaraki [1992], Stoer und Wagner [1997]) Ein Schnitt minimaler Kapazität in einem ungerichteten Graphen mit nichtnegativen Kapazitäten kann in $O(mn + n^2 \log n)$ -Zeit bestimmt werden.

Beweis: Wir können annehmen, dass der gegebene Graph G einfach ist, da wir parallele Kanten vereinen können. Sei $\lambda(G)$ die minimale Kapazität eines Schnittes in G . Der Algorithmus läuft wie folgt:

Sei $G_0 := G$. Im i -ten Schritt ($i = 1, \dots, n-1$) wähle man Knoten $x, y \in V(G_{i-1})$ mit

$$\lambda_{xy}^{G_{i-1}} = \sum_{e \in \delta_{G_{i-1}}(x)} u(e).$$

Nach Proposition 8.40 und Lemma 8.41 kann dies in $O(m + n \log n)$ -Zeit erreicht werden. Setze $\gamma_i := \lambda_{xy}^{G_{i-1}}$ und $z_i := x$. Es gehe G_i aus G_{i-1} durch Kontraktion von $\{x, y\}$ hervor. Beachte, dass

$$\lambda(G_{i-1}) = \min\{\lambda(G_i), \gamma_i\}, \quad (8.1)$$

weil ein Schnitt minimaler Kapazität in G_{i-1} entweder x und y trennt (in diesem Fall ist seine Kapazität gleich γ_i) oder nicht trennt (in diesem Fall ändert die Kontraktion von $\{x, y\}$ gar nichts).

Hat man G_{n-1} mit nur einem Knoten erreicht, so wähle man ein $k \in \{1, \dots, n-1\}$ mit γ_k minimal. Wir behaupten, dass $\delta(X)$ ein Schnitt minimaler Kapazität in G ist, wobei X diejenige Knotenmenge in G ist, deren Kontraktion zu dem Knoten z_k von G_{k-1} führte. Dies sieht man aber leicht: Nach (8.1) ist $\lambda(G) = \min\{\gamma_1, \dots, \gamma_{n-1}\} = \gamma_k$ und γ_k ist die Kapazität des Schnittes $\delta(X)$. \square

Ein randomisierter Kontraktionsalgorithmus zur Bestimmung eines Schnittes minimaler Kapazität (mit hoher Wahrscheinlichkeit) wird in Aufgabe 39 besprochen. Wir erwähnen noch, dass der Knotenzusammenhang eines Graphen mit $O(n^2)$ Flussberechnungen in einem Graphen mit Einheitskapazitäten bestimmt werden kann (Aufgabe 41).

In diesem Abschnitt haben wir gezeigt, wie man $f(X) := u(\delta(X))$ über der Menge $\emptyset \neq X \subset V(G)$ minimiert. Beachte, dass die Funktion $f : 2^{V(G)} \rightarrow \mathbb{R}_+$ submodular und symmetrisch ist (d. h. $f(A) = f(V(G) \setminus A)$ für alle A). Der oben präsentierte Algorithmus wurde von Queyranne [1998] für die Minimierung allgemeiner symmetrischer submodularer Funktionen verallgemeinert; siehe Abschnitt 14.5.

Das Problem der Bestimmung eines Schnittes maximalen Gewichtes ist viel schwieriger und wird in Abschnitt 16.2 besprochen.

Aufgaben

- Sei (G, u, s, t) ein Netzwerk und seien $\delta^+(X)$ und $\delta^+(Y)$ s - t -Schnitte minimaler Kapazität in (G, u) . Man zeige, dass $\delta^+(X \cap Y)$ und $\delta^+(X \cup Y)$ auch s - t -Schnitte minimaler Kapazität in (G, u) sind.
- Man zeige, dass der FORD-FULKERSON-ALGORITHMUS im Falle von irrationalen Kapazitäten nicht zu terminieren braucht.

Hinweis: Man betrachte das folgende Netzwerk (Abb. 8.6):

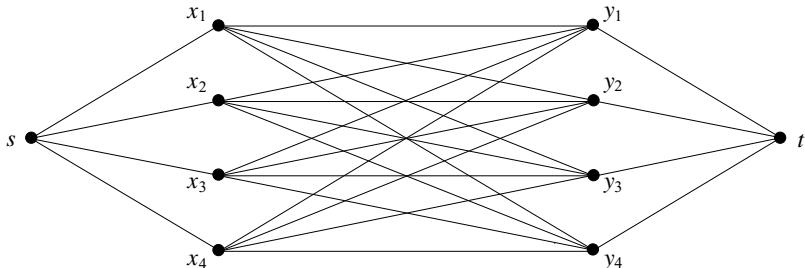


Abbildung 8.6.

Alle Liniensegmente sind Kanten in beiden Richtungen. Jede Kante hat Kapazität $S = \frac{1}{1-\sigma}$ bis auf die folgenden vier Kanten mit den Kapazitäten

$$u((x_1, y_1)) = 1, \quad u((x_2, y_2)) = \sigma, \quad u((x_3, y_3)) = u((x_4, y_4)) = \sigma^2,$$

wobei $\sigma = \frac{\sqrt{5}-1}{2}$. Man beachte, dass $\sigma^n = \sigma^{n+1} + \sigma^{n+2}$.
(Ford und Fulkerson [1962])

- * 3. Sei G ein Digraph und M die Inzidenzmatrix von G . Man beweise, dass für alle $c, l, u \in \mathbb{Z}^{E(G)}$ mit $l \leq u$:

$$\max \left\{ cx : x \in \mathbb{Z}^{E(G)}, l \leq x \leq u, Mx = 0 \right\} = \min \left\{ y'u - y''l : y', y'' \in \mathbb{Z}_+^{E(G)}, zM + y' - y'' = c \text{ für ein } z \in \mathbb{Z}^{V(G)} \right\}.$$

Man zeige, wie man mit diesem Resultat Satz 8.6 und Korollar 8.7 beweisen kann.

- 4. Man beweise den Zirkulationssatz von Hoffman: Gegeben sei ein Digraph G und untere bzw. obere Kapazitäten $l, u : E(G) \rightarrow \mathbb{R}_+$ mit $l(e) \leq u(e)$ für alle $e \in E(G)$. Es gibt eine Zirkulation f mit $l(e) \leq f(e) \leq u(e)$ für alle $e \in E(G)$ genau dann, wenn

$$\sum_{e \in \delta^-(X)} l(e) \leq \sum_{e \in \delta^+(X)} u(e) \quad \text{für alle } X \subseteq V(G).$$

Bemerkung: Aus dem Zirkulationssatz von Hoffman folgt relativ leicht das Max-Flow-Min-Cut-Theorem.

(Hoffman [1960])

- 5. Man betrachte ein Netzwerk (G, u, s, t) , einen s - t -Fluss f mit maximalem Wert und den Residualgraph G_f . Es gehe der Digraph H aus G_f hervor durch Kontraktion der Menge S der von s aus erreichbaren Knoten zu einem einzigen Knoten v_S , Kontraktion der Menge T derjenigen Knoten, von denen aus t erreichbar ist, zu einem einzigen Knoten v_T , und Kontraktion jeder starken Zusammenhangskomponente X von $G_f - (S \cup T)$ zu einem einzigen Knoten v_X . Man beachte, dass H azyklisch ist. Man beweise, dass es eine Bijektion gibt zwischen den Mengen $X \subseteq V(G)$, für die $\delta_G^+(X)$ ein s - t -Schnitt minimaler Kapazität in (G, u) ist, und den Mengen $Y \subseteq V(H)$, für die $\delta_H^+(Y)$ ein gerichteter v_T - v_S -Schnitt in H ist (d. h. ein gerichteter v_T und v_S trennender Schnitt in H).

Bemerkung: Diese Aussage gilt auch für G_f ohne jegliche Kontraktion anstelle von H . Wir werden jedoch die Aussage in der oben gegebenen Form in Abschnitt 20.5 anwenden.

(Picard und Queyranne [1980])

- 6. Sei G ein Digraph, $s, t \in V(G)$ und $c, c' : E(G) \rightarrow \mathbb{R}$ mit $c(e) \geq c'(e)$ für alle $e \in E(G)$. Gesucht wird eine Menge $X \subset V(G)$ mit $s \in X$ und $t \notin X$, für die $\sum_{e \in \delta^+(X)} c(e) - \sum_{e \in \delta^-(X)} c'(e)$ minimal ist.
 - (a) Man zeige, wie man dieses Problem auf das MINIMUM-CAPACITY-CUT-PROBLEM zurückführen kann.
 - (b) Man betrachte den Spezialfall $c = c'$. Ist es möglich, diesen in linearer Zeit zu lösen?

- * 7. Sei G ein azyklischer Digraph mit den Abbildungen $\sigma, \tau, c : E(G) \rightarrow \mathbb{R}_+$ und einer Zahl $C \in \mathbb{R}_+$. Gesucht wird eine Abbildung $x : E(G) \rightarrow \mathbb{R}_+$ mit $\sigma(e) \leq x(e) \leq \tau(e)$ für alle $e \in E(G)$ und $\sum_{e \in E(G)} (\tau(e) - x(e))c(e) \leq C$. Unter den zulässigen Lösungen möchte man die Länge (bezüglich x) des längsten Weges in G minimieren.

Dieses Problem kann folgendermaßen interpretiert werden. Den Kanten entsprechen Jobs, die Größe $\sigma(e)$ bzw. $\tau(e)$ ist die minimale bzw. maximale für Job e benötigte Zeit, und $c(e)$ sind die durch die Reduzierung der für Job e benötigten Zeit um eine Zeiteinheit verursachten Kosten. Sind $e = (i, j)$ und $e' = (j, k)$ zwei Jobs, so muss Job e beendet werden, bevor Job e' begonnen werden kann. Es steht eine feste Geldsumme C zur Verfügung und man möchte die insgesamt benötigte Zeit minimieren.

Man zeige, wie man dieses Problem mittels Netzwerkflussmethoden lösen kann. Die oben beschriebene Anwendung ist als PERT bekannt (program evaluation and review technique), oder auch als CPM (critical path method). Dieses Problem ist auch als die Budget-Version des Zeit-Kosten-Tradeoff-Problems bekannt.

Hinweis: Man füge eine Quelle s und eine Senke t hinzu. Man beginne mit $x = \tau$ und reduziere die Länge des längsten $s-t$ -Weges (bezüglich x) schrittweise mit minimalen Kosten. Man benutze Aufgabe 8 in Kapitel 7, Aufgabe 9 in Kapitel 3 und Aufgabe 6.

(Phillips und Dessouky [1977])

- * 8. Sei (G, c, s, t) ein Netzwerk mit der Eigenschaft, dass G planar ist und es bei dem Hinzufügen einer weiteren Kante $e = (s, t)$ auch bleibt. Man betrachte den folgenden Algorithmus. Man beginne mit dem Fluss $f \equiv 0$ und setze $G' := G_f$. Bei jeder Iteration betrachte man den Rand B eines e enthaltenden Gebietes von $G' + e$ (bezüglich einer festen planaren Einbettung). Man augmentiere f entlang $B - e$. Es bestehe G' aus den vorwärts gerichteten Kanten von G_f . Man iteriere weiter, so lange t von s aus in G' erreichbar ist.

Man beweise, dass dieser Algorithmus einen $s-t$ -Fluss mit maximalem Wert berechnet. Man verweise Satz 2.40 um zu zeigen, dass er mit $O(n^2)$ -Laufzeit implementiert werden kann.

(Ford und Fulkerson [1956], Hu [1969])

Bemerkung: Dieses Problem kann in $O(n)$ -Zeit gelöst werden. Für allgemeine planare Netzwerke gibt es einen $O(n \log n)$ -Algorithmus; siehe Weihe [1997] und Borradaile und Klein [2009].

9. Man zeige, dass die gerichtete kantendisjunkte Version des Satzes von Menger (Satz 8.9) auch direkt aus Satz 6.18 folgt.
10. Sei G ein ungerichteter Graph. Man beweise, dass man in linearer Zeit eine Orientierung G' von G mit der folgenden Eigenschaft berechnen kann. Für jedes Paar $v, w \in V(G)$ gilt: Gibt es zwei kantendisjunkte $v-w$ -Wege in G , so hat G' einen (gerichteten) $v-w$ -Weg.

Hinweis: Man benutze DFS.

(Tarjan [1972])

11. Sei G ein Digraph mit konservativen Gewichten $c : E(G) \rightarrow \mathbb{R}$ und zwei Knoten $s, t \in V(G)$, so dass t von s aus erreichbar ist. Angenommen, für jede Kante $e \in E(G)$ gilt $\text{dist}_{(G-e,c)}(s, t) = \text{dist}_{(G,c)}(s, t)$. Man beweise, dass es dann zwei kantendisjunkte kürzeste $s-t$ -Wege in (G, c) gibt.
12. Man betrachte einen ungerichteten Graphen G mit Kantenzusammenhang $k \in \mathbb{N}$ und (nicht notwendigerweise verschiedenen) Knoten $v_0, v_1, \dots, v_k \in V(G)$. Man beweise, dass es in G kantendisjunkte Wege P_1, \dots, P_k gibt, so dass P_i ein v_0-v_i -Weg ist ($i = 1, \dots, k$).
13. Sei G ein Graph (gerichtet oder ungerichtet), x, y, z drei Knoten und $\alpha, \beta \in \mathbb{N}$ mit $\alpha \leq \lambda_{xy}$, $\beta \leq \lambda_{xz}$ und $\alpha + \beta \leq \max\{\lambda_{xy}, \lambda_{xz}\}$. Man beweise, dass es α $x-y$ -Wege und β $x-z$ -Wege gibt, so dass diese $\alpha + \beta$ Wege kantendisjunkt sind.
14. Sei G ein Digraph mit der Eigenschaft: Für je zwei Knoten s und t enthält er k paarweise kantendisjunkte $s-t$ -Wege (ein solcher Graph heißt stark k -fach kantenzusammenhängend).
Sei H ein Digraph mit $V(H) = V(G)$ und $|E(H)| = k$. Man beweise, dass die Instanz (G, H) des GERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS eine Lösung besitzt.
(Mader [1981] und Shiloach [1979])
15. Sei G ein Digraph mit mindestens k Kanten. Man beweise: G enthält k paarweise kantendisjunkte $s-t$ -Wege für je zwei Knoten s und t genau dann, wenn $G - \{e_1, \dots, e_k\}$ für je k paarweise verschiedene Kanten $e_1 = (x_1, y_1), \dots, e_k = (x_k, y_k)$, k paarweise kantendisjunkte aufspannende Arboreszenzen T_1, \dots, T_k enthält, wobei T_i die Wurzel y_i hat ($i = 1, \dots, k$).
Bemerkung: Dieses Resultat verallgemeinert Aufgabe 14. *Hinweis:* Man benutze Satz 6.18.
(Su [1997])
16. Sei G ein Digraph mit Kapazitäten $c : E(G) \rightarrow \mathbb{R}_+$ und $r \in V(G)$. Kann man einen r -Schnitt minimaler Kapazität in polynomieller Zeit bestimmen? Kann man ferner einen gerichteten Schnitt minimaler Kapazität in polynomieller Zeit bestimmen (oder entscheiden, dass G stark zusammenhängend ist)?
Bemerkung: Die Antwort zu der ersten Frage löst das SEPARATIONS-PROBLEM für das MINIMUM-WEIGHT-ROOTED-ARBORESCENCE-PROBLEM; siehe Kollar 6.15.
17. Eine Fluggesellschaft möchte für eine gegebene Menge von Linienflügen so wenige Flugzeuge wie möglich einsetzen. Alle zur Verfügung stehenden Flugzeuge sind vom selben Typ. Für jeden der Linienflüge ist die Abflugzeit und Flugdauer bekannt. Ferner ist für je zwei Linienflüge i und j bekannt, wieviel Zeit für ein Flugzeug nach Beendigung von Flug i benötigt wird, bis es Flug j antreten kann (diese Zeit hängt insbesondere davon ab, wo Flug i endet und Flug j beginnt). Man zeige, wie man einen solchen Flugplan mit möglichst wenigen Flugzeugen effizient berechnen kann.

18. Sei G ein ungerichteter Graph, $s, t \in V(G)$ und $k \in \mathbb{N}$. Wie kann man in polynomieller Zeit k kantendisjunkte kürzeste $s-t$ -Wege bestimmen oder entscheiden, dass keine solchen Wege existieren?
19. Man beweise, dass der Wert eines blockierenden $s-t$ -Flusses in einem Netzwerk (G, u, s, t) , wobei G ein azyklischer Digraph ist, mindestens $\frac{1}{|V(G)|}$ mal der Wert eines maximalen $s-t$ -Flusses ist. Man zeige, dass diese Schranke bis auf einen konstanten Faktor die bestmögliche ist.
20. Man zeige, wie man einen blockierenden Fluss in einem azyklischen Netzwerk in $O(nm)$ -Zeit findet, indem man schrittweise entlang eines aus nichtsaturierten Kanten bestehenden Weges augmentiert und DEPTH-FIRST-SEARCH benutzt, um einen solchen Weg zu bestimmen. Man zeige, wie man eine $O(m)$ -Laufzeit erreichen kann, wenn alle Kanten Einheitskapazität haben.
- * 21. Sei (G, u, s, t) ein Netzwerk mit $u(e) = 1$ für alle Kanten $e \in E(G)$.
- Man zeige, dass ein $s-t$ -Fluss mit maximalem Wert in $O(mn^{2/3})$ -Zeit berechnet werden kann.
 - Es habe G zusätzlich die Eigenschaft, dass für jedes $v \in V(G) \setminus \{s, t\}$ $|\delta^-(v)| = 1$ oder $|\delta^+(v)| = 1$ gilt. Man zeige, dass ein $s-t$ -Fluss mit maximalem Wert in $O(m\sqrt{n})$ -Zeit berechnet werden kann.
- Hinweis:* Man betrachte DINIC' ALGORITHMUS und den Fall, dass kein augmentierender Weg kürzere Länge als $\lceil n^{2/3} \rceil$ bzw. $\lceil \sqrt{n} \rceil$ in (a) bzw. (b) hat. Man bestimme eine Schranke für die Anzahl der restlichen Iterationen und benutze den zweiten Teil von Aufgabe 20.
(Karzanov [1973], Even und Tarjan [1975])
22. Ein $s-t$ -Präfluss f heißt *maximal*, falls $\text{ex}_f(t)$ maximal ist.
- Man zeige, dass es für jeden maximalen Präfluss f einen maximalen Fluss f' mit $f'(e) \leq f(e)$ für alle $e \in E(G)$ gibt.
 - Man zeige, wie man einen maximalen Präfluss in $O(nm)$ -Zeit in einen maximalen Fluss umbauen kann.
23. Sei (G, u, s, t) ein Netzwerk, in welchem $G - t$ eine Arboreszenz ist. Man zeige, wie man einen $s-t$ -Fluss mit maximalem Wert in linearer Zeit findet.
- Hinweis:* Man benutze DFS.
- * 24. Sei (G, u, s, t) ein Netzwerk, in welchem der zugrunde liegende ungerichtete Graph von $G - \{s, t\}$ ein Wald ist. Man zeige, wie man einen $s-t$ -Fluss mit maximalem Wert in linearer Zeit findet.
(Vygen [2002])
25. Man betrachte eine modifizierte Version von FUJISHIGES ALGORITHMUS, in welcher man in ⑤ ein $v_i \in V(G) \setminus \{v_1, \dots, v_{i-1}\}$ so wählt, dass $b(v_i)$ maximal ist, ferner ④ durch den folgenden Schritt ersetzt: Man terminiere den Algorithmus, falls $b(v) = 0$ für alle $v \in V(G) \setminus \{v_1, \dots, v_i\}$, und schließlich am Anfang von ⑥ $\beta(t) := \min_{j=2}^i b(j)$ setzt. Dann werden X und α nicht mehr gebraucht.
- Man zeige, dass diese Version des Algorithmus korrekt arbeitet.

- (b) Sei α_k die Zahl $\min_{j=2}^i b(j)$ in Iteration k (oder Null, falls der Algorithmus vor Iteration k terminiert). Man zeige, dass $\min_{l=k+1}^{k+2n} \alpha_l \leq \frac{1}{2}\alpha_k$ für alle k . Man folgere hieraus, dass die Anzahl der Iterationen $O(n \log u_{\max})$ ist.
- (c) Man zeige, wie man eine Iteration in $O(m + n \log n)$ -Zeit implementieren kann.
26. Man beweise, dass der PUSH-RELABEL-ALGORITHMUS $O(n^2m)$ nichtsaturierende Pushes vollzieht, unabhängig von der Wahl von v in ③.
27. Sei (G, u, s, t) ein Netzwerk, f ein $s-t$ -Präfluss und ψ eine Distanzmarkierung bezüglich f mit $\psi(v) \leq 2n$ für $v \in V(G)$. Man definiere $\psi'(v) := \min\{\text{dist}_{G_f}(v, t), n + \text{dist}_{G_f}(v, s), 2n\}$ für $v \in V(G)$. Man zeige, dass ψ' eine Distanzmarkierung bezüglich f ist, und dass $\psi \leq \psi'$.
- Bemerkung:* Der PUSH-RELABEL-ALGORITHMUS läuft in der Praxis effizienter, wenn man von Zeit zu Zeit, z. B. nach je n RELABEL-Operationen, ψ durch ψ' ersetzt.
28. Gegeben sei ein azyklischer Digraph G mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$. Man finde einen gerichteten Schnitt maximalen Gewichtes in G . Man zeige, wie dieses Problem auf das MINIMUM-CAPACITY-CUT-PROBLEM zurückgeführt werden kann.
- Hinweis:* Man benutze Aufgabe 6.
29. Sei G ein azyklischer Digraph mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$. Gesucht wird die Kantenmenge $F \subseteq E(G)$ maximalen Gewichtes mit der Eigenschaft, dass kein Weg in G mehr als eine Kante aus F enthält. Man zeige, dass dieses Problem äquivalent mit folgendem ist: Gesucht wird der gerichtete Schnitt maximalen Gewichtes in G . (Folglich kann das erste Problem nach Aufgabe 28 in $O(n^3)$ -Zeit gelöst werden.)
30. Sei G ein Digraph und $p : V(G) \rightarrow \mathbb{R}$. Man zeige, wie man eine Menge $X \subseteq V(G)$ mit $\delta^+(X) = \emptyset$ bestimmt, für die $p(X)$ maximal ist.
- Bemerkung:* Dies wurde dazu benutzt, ein Modell für den Tagebau zu erstellen, wobei $p(v)$ der (möglicherweise auch negative) Erlös des Abbaus von v ist und eine Kante (v, w) die Nebenbedingung: Kein Abbau von v ohne Abbau von w bedeutet.
31. Gegeben sei ein ungerichteter Graph G mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und eine Menge $T \subseteq V(G)$ mit $|T| \geq 2$. Gesucht wird eine Menge $X \subset V(G)$ mit $T \cap X \neq \emptyset$ und $T \setminus X \neq \emptyset$, so dass $\sum_{e \in \delta(X)} u(e)$ minimal ist. Man zeige, wie man dieses Problem in $O(n^4)$ -Zeit löst, wobei $n = |V(G)|$.
32. Seien λ_{ij} , $1 \leq i, j \leq n$, nichtnegative Zahlen mit $\lambda_{ij} = \lambda_{ji}$ und $\lambda_{ik} \geq \min\{\lambda_{ij}, \lambda_{jk}\}$ für je drei paarweise verschiedene Indizes $i, j, k \in \{1, \dots, n\}$. Man zeige, dass es einen Graphen G mit $V(G) = \{1, \dots, n\}$ und Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ gibt, so dass die lokalen Kantenzusammenhänge genau die Zahlen λ_{ij} sind.
- Hinweis:* Man betrachte einen aufspannenden Baum maximalen Gewichtes in (K_n, c) , wobei $c(\{i, j\}) := \lambda_{ij}$.
(Gomory und Hu [1961])

33. Sei G ein ungerichteter Graph mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und $T \subseteq V(G)$ eine Knotenmenge mit $|T|$ gerade. Ein T -Schnitt in G ist ein Schnitt $\delta(X)$ mit $|X \cap T|$ ungerade. Man konstruiere einen polynomiellen Algorithmus zur Bestimmung eines T -Schnittes minimaler Kapazität in (G, u) .

Hinweis: Man verwende einen Gomory-Hu-Baum.

(Eine Lösung dieser Aufgabe ist in Abschnitt 12.3 enthalten.)

34. Sei G ein einfacher ungerichteter Graph mit mindestens zwei Knoten. Angenommen, der Grad eines jeden Knotens von G ist mindestens k . Man beweise, dass es zwei Knoten s und t gibt, für die mindestens k paarweise kantendisjunkte s - t -Wege existieren. Was kann man für den Fall sagen, dass es genau einen Knoten mit Grad kleiner als k gibt?

Hinweis: Man betrachte einen Gomory-Hu-Baum für G .

35. Man betrachte das Problem der Bestimmung des Kantenzusammenhanges $\lambda(G)$ eines ungerichteten Graphen (mit Einheitskapazitäten). In Abschnitt 8.7 wurde gezeigt, wie man dieses Problem in $O(mn)$ -Zeit löst, falls man eine MA-Reihenfolge eines ungerichteten Graphen mit Einheitskapazitäten in $O(m+n)$ -Zeit finden kann. Wie lässt sich Letzteres bewerkstelligen?
36. Sei $k \in \mathbb{N}$ und G ein ungerichteter Graph, in dem jeder Knoten mindestens Grad k hat. Man zeige, dass es dann zwei Knoten $s \neq t$ gibt, für die k kantendisjunkte s - t -Wege existieren.

- * 37. Sei G ein einfacher ungerichteter Graph mit einer MA-Reihenfolge v_1, \dots, v_n . Sei κ_{uv}^G die maximale Anzahl intern disjunkter u - v -Wege in G . Man beweise, dass $\kappa_{v_{n-1}v_n}^G = |\delta_G(v_n)|$ (dies ist die knotendisjunkte Variante von Lemma 8.41).

Hinweis: Man beweise mittels Induktion, dass $\kappa_{v_iv_j}^{G_{ij}} = |\delta_{G_{ij}}(v_j)|$ für alle $1 \leq i < j \leq n$, wobei $G_{ij} = G[\{v_1, \dots, v_i\} \cup \{v_j\}]$. Dazu nehme man o. B. d. A. $\{v_i, v_j\} \notin E(G)$ an, wähle eine inklusionsminimale v_j und trennende Menge $Z \subseteq \{v_1, \dots, v_{i-1}\}$ (Satz von Menger (Satz 8.10)) und lasse $h \leq i$ die maximale Zahl sein, so dass $v_h \notin Z$ und v_h mit v_i oder v_j benachbart ist.

(Frank [unveröffentlicht])

- * 38. Ein ungerichteter Graph heißt chordal, falls er keinen Kreis mindestens der Länge vier als induzierten Teilgraphen enthält. Eine Reihenfolge v_1, \dots, v_n eines ungerichteten Graphen G heißt simplizial, falls für zwei Kanten $\{v_i, v_j\}$, $\{v_i, v_k\} \in E(G)$ folgt, dass $\{v_j, v_k\} \in E(G)$ für $i < j < k$.
- (a) Man beweise, dass ein Graph mit einer simplizialen Reihenfolge chordal ist.

- (b) Sei G ein chordaler Graph und v_1, \dots, v_n eine MA-Reihenfolge. Man beweise, dass v_n, v_{n-1}, \dots, v_1 eine simpliziale Reihenfolge ist.

Hinweis: Man benutze Aufgabe 37 und den Satz von Menger (Satz 8.10).

Bemerkung: Die Tatsache, dass ein Graph genau dann chordal ist, wenn er eine simpliziale Reihenfolge besitzt, stammt von Rose [1970].

39. Sei G ein ungerichteter Graph mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$. Sei ferner $\emptyset \neq A \subset V(G)$, so dass $\delta(A)$ ein Schnitt minimaler Kapazität in G ist.
- Man zeige, dass $u(\delta(A)) \leq \frac{2}{n}u(E(G))$. (*Hinweis:* Man betrachte die trivialen Schnitte $\delta(x)$, $x \in V(G)$.)
 - Man nehme o. B. d. A. $u(\delta(A)) > 0$ an und betrachte das folgende Verfahren: Man wähle eine Kante zufällig und kontrahiere sie; jede Kante e wird mit der Wahrscheinlichkeit $\frac{u(e)}{u(E(G))}$ gewählt. Man wiederhole diese Operation bis nur noch zwei Knoten übrig sind. Man beweise, dass die Wahrscheinlichkeit, dass man niemals eine Kante von $\delta(A)$ kontrahiert, mindestens gleich $\frac{2}{(n-1)n}$ ist.
 - Man folgere hieraus: Lässt man den randomisierten Algorithmus in (b) kn^2 mal laufen, so ist die Wahrscheinlichkeit, dass man $\delta(A)$ erhält, mindestens gleich $1 - e^{-2k}$. (Ein solcher Algorithmus, der die korrekte Antwort mit einer positiven Wahrscheinlichkeit liefert, heißt Monte-Carlo-Algorithmus.) (Karger und Stein [1996]; siehe auch Karger [2000])
40. Man folgere aus Übung 39, dass es höchstens $\binom{n}{2}$ Schnitte minimaler Kapazität in einem ungerichteten Graphen mit n Knoten und nichtnegativen Kantenkapazitäten gibt. Man zeige, dass diese Schranke bestmöglich ist.
Bemerkung: Nagamochi, Nishimura und Ibaraki [1997] haben gezeigt, dass es höchstens $\binom{n}{2}$ Schnitte gibt, deren Kapazität um höchstens den Faktor $\frac{4}{3}$ größer ist als die minimale Kapazität eines Schnittes.
41. Man zeige, wie der Knotenzusammenhang eines ungerichteten Graphen in $O(n^{2.5}m)$ -Zeit bestimmt werden kann.
Hinweis: Man beziehe sich auf den Beweis des Satzes von Menger und benutze Aufgabe 21.
Bemerkung: Schnellere Algorithmen wurden von Henzinger, Rao und Gabow [2000] und von Gabow [2006] beschrieben.
42. Sei G ein zusammenhängender ungerichteter Graph mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$. Gesucht wird ein 3-Schnitt minimaler Kapazität, d. h. eine Kantenmenge, deren Entfernen G in mindestens drei Zusammenhangskomponenten zerlegt.
Sei $n := |V(G)| \geq 4$. Sei $\delta(X_1), \delta(X_2), \dots$ eine nach nicht abnehmenden Kapazitäten geordnete Liste der Schnitte: $u(\delta(X_1)) \leq u(\delta(X_2)) \leq \dots$. Angenommen, man kennt die ersten $2n-2$ Elemente dieser Liste (man beachte, dass sie mit einem Verfahren von Vazirani und Yannakakis [1992] in polynomieller Zeit berechnet werden können).
- Man zeige, dass es Indizes $i, j \in \{1, \dots, 2n-2\}$ gibt, für welche die Mengen $X_i \setminus X_j$, $X_j \setminus X_i$, $X_i \cap X_j$ und $V(G) \setminus (X_i \cup X_j)$ nicht leer sind.
 - Man zeige, dass es einen 3-Schnitt mit höchstens der Kapazität $\frac{3}{2}u(\delta(X_{2n-2}))$ gibt.
 - Für jedes $i = 1, \dots, 2n-2$ betrachte man $\delta(X_i)$ plus einen Schnitt minimaler Kapazität von $G - X_i$ und auch $\delta(X_i)$ plus einen Schnitt minimaler

Kapazität von $G[X_i]$. Damit erhält man eine Liste von höchstens $4n - 4$ 3-Schnitten. Man beweise, dass einer von ihnen optimal ist.

(Nagamochi und Ibaraki [2000])

Bemerkung: Dies wurde von Kamidou, Yoshida und Nagamochi [2007] auf k -Schnitte (für ein beliebiges festes k) verallgemeinert; siehe auch Thorup [2008]. Das Problem der Bestimmung des optimalen drei gegebenen Knoten trennenden 3-Schnitte ist viel schwieriger; siehe Dahlhaus et al. [1994] und Cheung, Cunningham und Tang [2006].

43. Sei G ein ungerichteter Graph mit Kapazitäten $\mu : E(G) \rightarrow \mathbb{Z}_+$.
- Man zeige: Sind $\delta(X)$ und $\delta(Y)$ zwei Schnitte minimaler Kapazität und $X \cap Y \neq \emptyset$, $X \cup Y \neq V(G)$, so gilt $\delta(X \setminus Y) \cap \delta(Y \setminus X) = \emptyset$.
 - Angenommen, die minimale Kapazität eines Schnittes sei ungerade. Dann zeige man: Die Familie der Knotenmengen X für die $\delta(X)$ ein Schnitt mit minimaler Kapazität ist, ist kreuzungsfrei. Man leite daraus ab, dass es höchstens $n - 1$ Schnitte mit minimaler Kapazität gibt.

Bemerkung: Dinitz, Karzanov und Lomonosov [1976] haben gezeigt, dass es im Allgemeinen höchstens $\binom{n}{2}$ Schnitte mit minimaler Kapazität gibt (was auch aus Übung 39(b) folgt). Diese können mittels einer sogenannten Kaktus-Repräsentation beschrieben werden, die Baumrepräsentationen verallgemeinern. Siehe auch Frank [2011].

Literatur

Allgemeine Literatur:

- Ahuja, R.K., Magnanti, T.L., und Orlin, J.B. [1993]: Network Flows. Prentice-Hall, Englewood Cliffs 1993
- Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., und Schrijver, A. [1998]: Combinatorial Optimization. Wiley, New York 1998, Kapitel 3
- Cormen, T.H., Leiserson, C.E., Rivest, R.L., und Stein, C. [2009]: Introduction to Algorithms. 3. Aufl. MIT Press, Cambridge 2009, Kapitel 26
- Ford, L.R., und Fulkerson, D.R. [1962]: Flows in Networks. Princeton University Press, Princeton 1962
- Frank, A. [1995]: Connectivity and network flows. In: Handbook of Combinatorics; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, Hrsg.), Elsevier, Amsterdam, 1995
- Frank, A. [2011]: Connections in Combinatorial Optimization. Oxford University Press, Oxford 2011
- Goldberg, A.V., Tardos, É., und Tarjan, R.E. [1990]: Network flow algorithms. In: Paths, Flows, and VLSI-Layout (B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, Hrsg.), Springer, Berlin 1990, pp. 101–164
- Gondran, M., und Minoux, M. [1984]: Graphs and Algorithms. Wiley, Chichester 1984, Kapitel 5
- Jungnickel, D. [2013]: Graphs, Networks and Algorithms. 4. Aufl. Springer, Berlin 2013
- Phillips, D.T., und Garcia-Diaz, A. [1981]: Fundamentals of Network Analysis. Prentice-Hall, Englewood Cliffs 1981
- Ruhe, G. [1991]: Algorithmic Aspects of Flows in Networks. Kluwer Academic Publishers, Dordrecht 1991

- Schrijver, A. [2003]: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin 2003, Kapitel 9,10,13–15
- Tarjan, R.E. [1983]: Data Structures and Network Algorithms. SIAM, Philadelphia 1983, Kapitel 8
- Thulasiraman, K., und Swamy, M.N.S. [1992]: Graphs: Theory and Algorithms. Wiley, New York 1992, Kapitel 12

Zitierte Literatur:

- Ahuja, R.K., Orlin, J.B., und Tarjan, R.E. [1989]: Improved time bounds for the maximum flow problem. *SIAM Journal on Computing* 18 (1989), 939–954
- Borradaile, G. und Klein, P. [2009]: An $O(n \log n)$ algorithm for maximum st -flow in a directed planar graph. *Journal of the ACM* 56 (2009), Article 9
- Cherian, J., und Maheshwari, S.N. [1989]: Analysis of preflow push algorithms for maximum network flow. *SIAM Journal on Computing* 18 (1989), 1057–1086
- Cherian, J., und Mehlhorn, K. [1999]: An analysis of the highest-level selection rule in the preflow-push max-flow algorithm. *Information Processing Letters* 69 (1999), 239–242
- Cherkassky, B.V. [1977]: Algorithm of construction of maximal flow in networks with complexity of $O(V^2\sqrt{E})$ operations. *Mathematical Methods of Solution of Economical Problems* 7 (1977), 112–125 [auf Russisch]
- Cheung, K.K.H., Cunningham, W.H., und Tang, L. [2006]: Optimal 3-terminal cuts and linear programming. *Mathematical Programming* 106 (2006), 1–23
- Cheung, H.Y., Lau, L.C., und Leung, K.M. [2011]: Graph connectivities, network coding, and expander graphs. *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science* (2011), 197–206
- Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., und Yannakakis, M. [1994]: The complexity of multiterminal cuts. *SIAM Journal on Computing* 23 (1994), 864–894
- Dantzig, G.B., und Fulkerson, D.R. [1956]: On the max-flow min-cut theorem of networks. In: *Linear Inequalities and Related Systems* (H.W. Kuhn, A.W. Tucker, Hrsg.), Princeton University Press, Princeton 1956, pp. 215–221
- Dinic, E.A. [1970]: Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Mathematics Doklady* 11 (1970), 1277–1280
- Dinitz, E.A., Karzanov, A.V., und Lomonosov, M.V. [1976]: On the structure of the system of minimum edge cuts of a graph. *Issledovaniya po Diskretnoi Optimizatsii* (A.A. Fridman, Hrsg.), Nauka, Moskau, 1976, pp. 290–306 [auf Russisch]
- Edmonds, J., und Karp, R.M. [1972]: Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM* 19 (1972), 248–264
- Elias, P., Feinstein, A., und Shannon, C.E. [1956]: Note on maximum flow through a network. *IRE Transactions on Information Theory*, IT-2 (1956), 117–119
- Even, S., und Tarjan, R.E. [1975]: Network flow and testing graph connectivity. *SIAM Journal on Computing* 4 (1975), 507–518
- Ford, L.R., und Fulkerson, D.R. [1956]: Maximal Flow Through a Network. *Canadian Journal of Mathematics* 8 (1956), 399–404
- Ford, L.R., und Fulkerson, D.R. [1957]: A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. *Canadian Journal of Mathematics* 9 (1957), 210–218
- Frank, A. [1994]: On the edge-connectivity algorithm of Nagamochi und Ibaraki. Laboratoire Artemis, IMAG, Université J. Fourier, Grenoble, 1994

- Fujishige, S. [2003]: A maximum flow algorithm using MA ordering. *Operations Research Letters* 31 (2003), 176–178
- Gabow, H.N. [1995]: A matroid approach to finding edge connectivity and packing arborescences. *Journal of Computer and System Sciences* 50 (1995), 259–273
- Gabow, H.N. [2006]: Using expander graphs to find vertex-connectivity. *Journal of the ACM* 53 (2006), 800–844
- Galil, Z. [1980]: An $O(V^{\frac{5}{3}}E^{\frac{2}{3}})$ algorithm for the maximal flow problem. *Acta Informatica* 14 (1980), 221–242
- Galil, Z., und Namaad, A. [1980]: An $O(EV \log^2 V)$ algorithm for the maximal flow problem. *Journal of Computer and System Sciences* 21 (1980), 203–217
- Gallai, T. [1958]: Maximum-minimum Sätze über Graphen. *Acta Mathematica Academiae Scientiarum Hungaricae* 9 (1958), 395–434
- Goldberg, A.V., und Rao, S. [1998]: Beyond the flow decomposition barrier. *Journal of the ACM* 45 (1998), 783–797
- Goldberg, A.V., und Tarjan, R.E. [1988]: A new approach to the maximum flow problem. *Journal of the ACM* 35 (1988), 921–940
- Gomory, R.E., und Hu, T.C. [1961]: Multi-terminal network flows. *Journal of SIAM* 9 (1961), 551–570
- Gusfield, D. [1990]: Very simple methods for all pairs network flow analysis. *SIAM Journal on Computing* 19 (1990), 143–155
- Hao, J., und Orlin, J.B. [1994]: A faster algorithm for finding the minimum cut in a directed graph. *Journal of Algorithms* 17 (1994), 409–423
- Henzinger, M.R., Rao, S., und Gabow, H.N. [2000]: Computing vertex connectivity: new bounds from old techniques. *Journal of Algorithms* 34 (2000), 222–250
- Henzinger, M.R., Rao, S., und Wang, D. [2017]: Local flow partitioning for faster edge connectivity. *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms* (2017), 1919–1938
- Hoffman, A.J. [1960]: Some recent applications of the theory of linear inequalities to extremal combinatorial analysis. In: *Combinatorial Analysis* (R.E. Bellman, M. Hall, Hrsg.), AMS, Providence 1960, pp. 113–128
- Hu, T.C. [1969]: Integer Programming and Network Flows. Addison-Wesley, Reading 1969
- Jelinek, F., und Mayeda, W. [1963]: On the maximum number of different entries in the terminal capacity matrix of oriented communication nets. *IEEE Transactions on Circuit Theory* 10 (1963), 307–308
- Kamidou, Y., Yoshida, N., und Nagamochi, H. [2007]: A deterministic algorithm for finding all minimum k -way cuts. *SIAM Journal on Computing* 36 (2007), 1329–1341
- Karger, D.R. [2000]: Minimum cuts in near-linear time. *Journal of the ACM* 47 (2000), 46–76
- Karger, D.R., und Levine, M.S. [1998]: Finding maximum flows in undirected graphs seems easier than bipartite matching. *Proceedings of the 30th Annual ACM Symposium on Theory of Computing* (1998), 69–78
- Karger, D.R., und Stein, C. [1996]: A new approach to the minimum cut problem. *Journal of the ACM* 43 (1996), 601–640
- Karzanov, A.V. [1973]: On finding a maximum flow in a network with special structure and some applications. In: *Matematicheskie Voprosy Upravleniya Proizvodstvom* 5 (L.A. Lyusternik, Hrsg.), Moscow State University Press, Moskau, 1973, pp. 81–94 [auf Russisch]
- Karzanov, A.V. [1974]: Determining a maximal flow in a network by the method of preflows. *Soviet Mathematics Doklady* 15 (1974), 434–437

- Lee, Y.T., und Sidford, A. [2014]: Path finding methods for linear programming. Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (2014), 424–433
- Mader, W. [1972]: Über minimal n -fach zusammenhängende, unendliche Graphen und ein Extremalproblem. Arch. Math. 23 (1972), 553–560
- Mader, W. [1981]: On a property of n edge-connected digraphs. Combinatorica 1 (1981), 385–386
- Malhotra, V.M., Kumar, M.P., und Maheshwari, S.N. [1978]: An $O(|V|^3)$ algorithm for finding maximum flows in networks. Information Processing Letters 7 (1978), 277–278
- Menger, K. [1927]: Zur allgemeinen Kurventheorie. Fundamenta Mathematicae 10 (1927), 96–115
- Nagamochi, H., und Ibaraki, T. [1992]: Computing edge-connectivity in multigraphs and capacitated graphs. SIAM Journal on Discrete Mathematics 5 (1992), 54–66
- Nagamochi, H., und Ibaraki, T. [2000]: A fast algorithm for computing minimum 3-way and 4-way cuts. Mathematical Programming 88 (2000), 507–520
- Nagamochi, H., Nishimura, K., und Ibaraki, T. [1997]: Computing all small cuts in an undirected network. SIAM Journal on Discrete Mathematics 10 (1997), 469–481
- Orlin, J.B. [2013]: Max flows in $O(nm)$ time, or better. Proceedings of the 45th Annual ACM Symposium on Theory of Computing (2013), 765–774
- Phillips, S., und Dessouky, M.I. [1977]: Solving the project time/cost tradeoff problem using the minimal cut concept. Management Science 24 (1977), 393–400
- Picard, J., und Queyranne, M. [1980]: On the structure of all minimum cuts in a network and applications. Mathematical Programming Study 13 (1980), 8–16
- Queyranne, M. [1998]: Minimizing symmetric submodular functions. Mathematical Programming B 82 (1998), 3–12
- Rose, D.J. [1970]: Triangulated graphs and the elimination process. Journal of Mathematical Analysis and Applications 32 (1970), 597–609
- Shiloach, Y. [1978]: An $O(nI \log^2 I)$ maximum-flow algorithm. Technical Report STAN-CS-78-802, Computer Science Department, Stanford University, 1978
- Shiloach, Y. [1979]: Edge-disjoint branching in directed multigraphs. Information Processing Letters 8 (1979), 24–27
- Shioura, A. [2004]: The MA ordering max-flow algorithm is not strongly polynomial for directed networks. Operations Research Letters 32 (2004), 31–35
- Sleator, D.D. [1980]: An $O(nm \log n)$ algorithm for maximum network flow. Technical Report STAN-CS-80-831, Computer Science Department, Stanford University, 1978
- Sleator, D.D., und Tarjan, R.E. [1983]: A data structure for dynamic trees. Journal of Computer and System Sciences 26 (1983), 362–391
- Su, X.Y. [1997]: Some generalizations of Menger's theorem concerning arc-connected digraphs. Discrete Mathematics 175 (1997), 293–296
- Stoer, M., und Wagner, F. [1997]: A simple min cut algorithm. Journal of the ACM 44 (1997), 585–591
- Tarjan, R.E. [1972]: Depth first search and linear graph algorithms. SIAM Journal on Computing 1 (1972), 146–160
- Thorup, M. [2008]: Minimum k-way cuts via deterministic greedy tree packing. Proceedings of the 40th Annual ACM Symposium on Theory of Computing (2008), 159–165
- Tuncel, L. [1994]: On the complexity preflow-push algorithms for maximum flow problems. Algorithmica 11 (1994), 353–359
- Vazirani, V.V., und Yannakakis, M. [1992]: Suboptimal cuts: their enumeration, weight, and number. In: Automata, Languages and Programming; Proceedings of the 19th ICALP conference; LNCS 623 (W. Kuich, Hrsg.), Springer, Berlin 1992, pp. 366–377

- Vygen, J. [2002]: On dual minimum cost flow algorithms. *Mathematical Methods of Operations Research* 56 (2002), 101–126
- Weihe, K. [1997]: Maximum (s, t) -flows in planar networks in $O(|V| \log |V|)$ time. *Journal of Computer and System Sciences* 55 (1997), 454–475
- Whitney, H. [1932]: Congruent graphs and the connectivity of graphs. *American Journal of Mathematics* 54 (1932), 150–168



9 Flüsse mit minimalen Kosten

In diesem Kapitel besprechen wir, wie man vorgeht, wenn zusätzlich die Kanten mit Kosten belegt sind. Zum Beispiel könnte man in der Anwendung des MAXIMUM-FLOW-PROBLEMS auf das JOB-ZUORDNUNGSPROBLEM (siehe Einführung in Kapitel 8) Kosten auf den Kanten einführen, um den Arbeitern verschiedene Gehälter zuzuordnen; das Ziel wäre dann, bis zu einem festgelegten Zeitpunkt und zu minimalen Gesamtkosten alle Jobs erledigt zu haben. Natürlich gibt es etliche weitere Anwendungen.

Eine zweite Verallgemeinerung, nämlich die Einführung mehrerer Quellen und Senken, ist mehr technischer Natur. In Abschnitt 9.1 definieren wir das allgemeine Problem, sowie einen wichtigen Spezialfall. In Abschnitt 9.2 beweisen wir gewisse Optimalitätskriterien, die die Basis der in den Abschnitten 9.3, 9.4, 9.5 und 9.6 zu besprechenden Minimum-Cost-Flow-Algorithmen bilden. Die meisten von ihnen benutzen die in Kapitel 7 besprochenen Algorithmen zur Bestimmung eines Kreises mit minimalem durchschnittlichem Kantengewicht oder eines kürzesten Weges als Subroutine. Abschließend betrachten wir in Abschnitt 9.7 eine Anwendung auf zeitabhängige Flüsse.

9.1 Formulierung des Problems

Gegeben sei wiederum ein Digraph G mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und zusätzlich Zahlen $c : E(G) \rightarrow \mathbb{R}$, die die Kosten der Kanten angeben. Ferner lassen wir mehrfache Quellen und Senken zu:

Definition 9.1. Gegeben sei ein Digraph G , Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und Zahlen $b : V(G) \rightarrow \mathbb{R}$ mit $\sum_{v \in V(G)} b(v) = 0$. Ein **b -Fluss** in (G, u) ist eine Funktion $f : E(G) \rightarrow \mathbb{R}_+$ mit $f(e) \leq u(e)$ für alle $e \in E(G)$ und $\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = b(v)$ für alle $v \in V(G)$.

Ein b -Fluss mit $b \equiv 0$ ist also eine Zirkulation. Die Zahl $b(v)$ heißt die **Balance** des Knotens v . Gelegentlich heißt $|b(v)|$ das **Angebot** (für $b(v) > 0$) oder die **Nachfrage** (für $b(v) < 0$) von v . Knoten v mit $b(v) > 0$ heißen **Quellen**, diejenigen mit $b(v) < 0$ heißen **Senken**.

Satz 9.2. (Gale [1957]) Sei G ein Digraph mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und $b : V(G) \rightarrow \mathbb{R}$ mit $\sum_{v \in V(G)} b(v) = 0$. Ein b -Fluss existiert genau dann, wenn

$$\sum_{e \in \delta^+(X)} u(e) \geq \sum_{v \in X} b(v) \quad \text{für alle } X \subseteq V(G).$$

Man kann in $O(n^2\sqrt{m})$ -Zeit (mit $n := |V(G)|$ und $m := |E(G)|$) einen solchen bestimmten oder entscheiden, dass es keinen solchen gibt.

Beweis: Es gehe G' aus G hervor durch Hinzufügung zweier Knoten s , t und den Kanten (s, v) , (v, t) mit Kapazitäten $u((s, v)) := \max\{0, b(v)\}$ und $u((v, t)) := \max\{0, -b(v)\}$ für alle $v \in V(G)$. Sei $B := \sum_{v \in V(G)} u((s, v)) = \sum_{v \in V(G)} u((v, t))$. Dann sind die b -Flüsse in (G, u) genau die Restriktionen der s - t -Flüsse mit Wert B in (G', u) auf $E(G)$. Nach dem Max-Flow-Min-Cut-Theorem 8.6 gibt es genau dann einen s - t -Fluss mit Wert B in (G', u) , wenn $\sum_{e \in \delta_{G'}^+(\{s\} \cup X)} u(e) \geq B$ für jedes $X \subseteq V(G)$. Beachte, dass $\sum_{e \in \delta_{G'}^+(\{s\} \cup X)} u(e) = \sum_{e \in \delta_G^+(X)} u(e) + B + \sum_{v \in X} (\max\{0, -b(v)\} - \max\{0, b(v)\}) = \sum_{e \in \delta_G^+(X)} u(e) + B - \sum_{v \in X} b(v)$, womit die erste Aussage bewiesen ist.

Man kann einen b -Fluss bestimmen oder entscheiden, dass es keinen solchen gibt, indem man das MAXIMUM-FLOW-PROBLEM in (G', u, s, t) löst. Die Laufzeit folgt mit Satz 8.30. \square

Das Ziel dieses Kapitels ist, einen b -Fluss mit minimalen Kosten zu finden:

MINIMUM-COST-FLOW-PROBLEM

Instanz: Ein Digraph G , Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$, Zahlen $b : V(G) \rightarrow \mathbb{R}$ mit $\sum_{v \in V(G)} b(v) = 0$ und Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Bestimme einen b -Fluss f mit minimalen Kosten $c(f) := \sum_{e \in E(G)} f(e)c(e)$ (oder entscheide, dass es keinen solchen gibt).

Manchmal lässt man auch unendliche Kapazitäten zu. In einem solchen Fall kann eine Instanz auch unbeschränkt sein. Dies kann aber leicht vorab geprüft werden, siehe Aufgabe 5. Beachte, dass man (im Gegensatz zu manch einem anderen Problem, z. B. dem MAXIMUM-FLOW-PROBLEM) o. B. d. A. nicht davon ausgehen kann, dass der Input-Graph beim MINIMUM-COST-FLOW-PROBLEM einfach ist.

Das MINIMUM-COST-FLOW-PROBLEM ist recht allgemein und besitzt einige interessante Spezialfälle. Ohne Kapazitäten (d. h. $u \equiv \infty$) heißt dieses Problem auch das Transshipment-Problem. Ein noch weiter eingeschränktes Problem, auch Transportproblem genannt, wurde recht früh von Hitchcock [1941] und anderen formuliert:

HITCHCOCK-PROBLEM

Instanz: Ein Digraph G mit $V(G) = A \dot{\cup} B$ und $E(G) \subseteq A \times B$. Angebote $b(v) \geq 0$ für $v \in A$ und Nachfragen $-b(v) \geq 0$ für $v \in B$ mit $\sum_{v \in V(G)} b(v) = 0$. Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Bestimme einen b -Fluss f in (G, ∞) mit minimalen Kosten (oder entscheide, dass es keinen solchen gibt).

Für das HITCHCOCK-PROBLEM können wir o. B. d. A. annehmen, dass c nichtnegativ ist: Die Addition einer Konstante α zu jedem Gewicht erhöht die Kosten eines jeden b -Flusses um denselben Betrag, nämlich um $\alpha \sum_{v \in A} b(v)$. Oft wird nur der Spezialfall c nichtnegativ und $E(G) = A \times B$ betrachtet.

Offensichtlich kann jede Instanz des HITCHCOCK-PROBLEMS als eine Instanz des MINIMUM-COST-FLOW-PROBLEMS auf einem bipartiten Graphen mit unendlichen Kapazitäten betrachtet werden. Weniger offensichtlich ist es, dass jede Instanz des MINIMUM-COST-FLOW-PROBLEMS in eine äquivalente (aber größere) Instanz des HITCHCOCK-PROBLEMS transformiert werden kann:

Lemma 9.3. (Orden [1956], Wagner [1959]) *Eine Instanz des MINIMUM-COST-FLOW-PROBLEMS mit n Knoten und m Kanten kann in eine äquivalente Instanz des HITCHCOCK-PROBLEMS mit $n + m$ Knoten und $2m$ Kanten transformiert werden.*

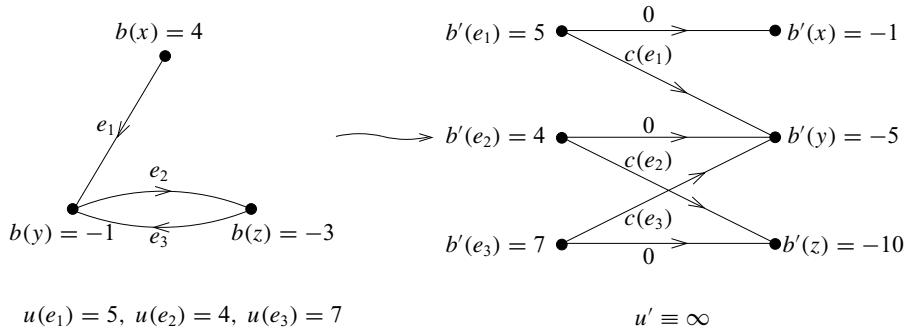


Abbildung 9.1.

Beweis: Sei (G, u, b, c) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS. Wir definieren eine äquivalente Instanz (G', A', B', b', c') des HITCHCOCK-PROBLEMS wie folgt:

Seien $A' := E(G)$, $B' := V(G)$ und $G' := (A' \cup B', E_1 \cup E_2)$, wobei $E_1 := \{(x, y), x\} : (x, y) \in E(G)\}$ und $E_2 := \{((x, y), y) : (x, y) \in E(G)\}$. Seien $c'((e, x)) := 0$ für $(e, x) \in E_1$ und $c'((e, y)) := c(e)$ für $(e, y) \in E_2$. Seien schließlich $b'(e) := u(e)$ für $e \in E(G)$ und

$$b'(x) := b(x) - \sum_{e \in \delta_G^+(x)} u(e) \quad \text{für } x \in V(G).$$

Ein Beispiel wird in Abb. 9.1 gezeigt.

Wir beweisen, dass die beiden Instanzen äquivalent sind. Sei f ein b -Fluss in (G, u) . Setze $f'((e, y)) := f(e)$ und $f'((e, x)) := u(e) - f(e)$ für $e = (x, y) \in E(G)$. Offensichtlich ist f' ein b' -Fluss in G' mit $c'(f') = c(f)$.

Ist umgekehrt f' ein b' -Fluss in G' , so definiert $f((x, y)) := f'(((x, y), y))$ einen b -Fluss in G mit $c(f) = c'(f')$. \square

Dieser Beweis stammt von Ford und Fulkerson [1962].

9.2 Ein Optimalitätskriterium

In diesem Abschnitt beweisen wir einige einfache Resultate, insbesondere ein Optimalitätskriterium, welches den Algorithmen der folgenden Abschnitte zugrunde liegt. Wiederum werden wir die Begriffe des Residualgraphen und der augmentierenden Wege benötigen. Ferner werden wir die Gewichte c auf $\overset{\leftrightarrow}{G}$ erweitern, indem wir $c(\overset{\leftarrow}{e}) := -c(e)$ für jede Kante $e \in E(G)$ setzen. Unsere Definition des Residualgraphen hat den Vorteil, dass das Gewicht einer Kante in einem Residualgraphen G_f unabhängig von dem Fluss f ist.

Das folgende einfache Resultat wird sich als nützlich erweisen:

Proposition 9.4. *Sei (G, u, b, c) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS. Seien f und f' b -Flüsse in (G, u) . Dann ist die Funktion $g : E(G) \rightarrow \mathbb{R}_+$, gegeben durch $g(e) := \max\{0, f'(e) - f(e)\}$ und $g(\overset{\leftarrow}{e}) := \max\{0, f(e) - f'(e)\}$ für $e \in E(G)$, eine Zirkulation in $\overset{\leftrightarrow}{G}$. Ferner ist $g(e) = 0$ für alle $e \notin E(G_f)$ und $c(g) = c(f') - c(f)$.*

Beweis: In jedem Knoten $v \in V(\overset{\leftrightarrow}{G})$ haben wir

$$\begin{aligned} \sum_{e \in \delta_G^+(v)} g(e) - \sum_{e \in \delta_G^-(v)} g(e) &= \sum_{e \in \delta_G^+(v)} (f'(e) - f(e)) - \sum_{e \in \delta_G^-(v)} (f'(e) - f(e)) \\ &= b(v) - b(v) = 0, \end{aligned}$$

also ist g eine Zirkulation in $\overset{\leftrightarrow}{G}$.

Nun betrachten wir für jedes $e \in E(\overset{\leftrightarrow}{G}) \setminus E(G_f)$ zwei Fälle: Ist $e \in E(G)$, so gilt $f(e) = u(e)$, folglich ist $f'(e) \leq f(e)$ und damit $g(e) = 0$. Ist $e = \overset{\leftarrow}{e_0}$ für ein $e_0 \in E(G)$, so gilt $f(e_0) = 0$, folglich ist $g(\overset{\leftarrow}{e_0}) = 0$.

Die letzte Aussage lässt sich leicht beweisen:

$$c(g) = \sum_{e \in E(\overset{\leftrightarrow}{G})} c(e)g(e) = \sum_{e \in E(G)} c(e)f'(e) - \sum_{e \in E(G)} c(e)f(e) = c(f') - c(f).$$

□

Genau so, wie man eulersche Graphen in Kreise partitionieren kann, lassen sich Zirkulationen in Flüsse entlang einzelnen Kreisen zerlegen:

Proposition 9.5. (Ford und Fulkerson [1962]) *Für jede Zirkulation f in einem Digraphen G gibt es eine Familie \mathcal{C} von höchstens $|E(G)|$ Kreisen in G und positive Zahlen $h(C)$ ($C \in \mathcal{C}$) mit $f(e) = \sum_{C \in \mathcal{C}, e \in E(C)} h(C)$ für alle $e \in E(G)$.*

Beweis: Dies ist ein Spezialfall von Satz 8.8.

□

Definition 9.6. Gegeben seien ein Digraph G mit den Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und ein b -Fluss f in (G, u) . Dann ist ein **f -augmentierende Kreis** ein Kreis in G_f .

Nun können wir ein Optimalitätskriterium beweisen:

Satz 9.7. (Klein [1967]) Sei (G, u, b, c) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS. Ein b -Fluss f hat genau dann minimale Kosten, wenn es keinen f -augmentierenden Kreis mit negativem Gesamtgewicht gibt.

Beweis: Gibt es einen f -augmentierenden Kreis C mit Gewicht $\gamma < 0$, so können wir f entlang C um ein $\varepsilon > 0$ augmentieren und einen b -Fluss f' mit um $-\gamma \varepsilon$ gesenkten Kosten erhalten. Somit ist f' kein Fluss mit minimalen Kosten.

Ist f kein b -Fluss mit minimalen Kosten, dann gibt es einen anderen b -Fluss f' mit geringeren Kosten. Betrachte das in Proposition 9.4 definierte g . Dann ist g eine Zirkulation mit $c(g) < 0$. Nach Proposition 9.5 kann man g dann in Flüsse entlang einzelnen Kreisen zerlegen. Da $g(e) = 0$ für alle $e \notin E(G_f)$, sind all diese Kreise f -augmentierend. Mindestens einer von ihnen muss jedoch negatives Gesamtgewicht haben, womit der Satz bewiesen ist. \square

Dieser Satz stammt im Wesentlichen von Tolstoi [1930] und ist mehrere Male in verschiedenen Formen wiederentdeckt worden. Eine äquivalente Formulierung wird in folgendem Korollar gegeben:

Korollar 9.8. (Ford und Fulkerson [1962]) Sei (G, u, b, c) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS. Ein b -Fluss f hat genau dann minimale Kosten, wenn es ein zulässiges Potenzial für (G_f, c) gibt.

Beweis: Nach Satz 9.7 ist f genau dann ein b -Fluss mit minimalen Kosten, wenn G_f keine negativen Kreise enthält. Nach Satz 7.7 enthält (G_f, c) genau dann keinen negativen Kreis, wenn es ein zulässiges Potenzial gibt. \square

Zulässige Potenziale können auch als Lösungen des LP-Duals des MINIMUM-COST-FLOW-PROBLEMS aufgefasst werden. Dies wird aus dem folgenden alternativen Beweis des obigen Optimalitätskriteriums ersichtlich:

Zweiter Beweis von Korollar 9.8: Wir schreiben das MINIMUM-COST-FLOW-PROBLEM als Maximierungsproblem und betrachten das LP

$$\begin{aligned} \max \quad & \sum_{e \in E(G)} -c(e)x_e \\ \text{bzgl.} \quad & \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = b(v) \quad (v \in V(G)) \\ & x_e \leq u(e) \quad (e \in E(G)) \\ & x_e \geq 0 \quad (e \in E(G)) \end{aligned} \tag{9.1}$$

und das dazu duale LP

$$\begin{aligned} \min \quad & \sum_{v \in V(G)} b(v)y_v + \sum_{e \in E(G)} u(e)z_e \\ \text{bzgl.} \quad & y_v - y_w + z_e \geq -c(e) \quad (e = (v, w) \in E(G)) \\ & z_e \geq 0 \quad (e \in E(G)). \end{aligned} \tag{9.2}$$

Sei x ein b -Fluss, d.h. eine zulässige Lösung von (9.1). Nach Korollar 3.23 ist x genau dann eine optimale Lösung, wenn es eine zulässige duale Lösung (y, z) von (9.2) gibt, so dass x und (y, z) die Bedingungen des komplementären Schlupfes erfüllen:

$$z_e(u(e) - x_e) = 0 \text{ und } x_e(c(e) + z_e + y_v - y_w) = 0 \text{ für alle } e = (v, w) \in E(G).$$

Es ist x also genau dann eine optimale Lösung, wenn es ein Vektorenpaar (y, z) gibt, mit

$$\begin{aligned} 0 = -z_e &\leq c(e) + y_v - y_w && \text{für } e = (v, w) \in E(G) \text{ mit } x_e < u(e) \text{ und} \\ c(e) + y_v - y_w &= -z_e \leq 0 && \text{für } e = (v, w) \in E(G) \text{ mit } x_e > 0. \end{aligned}$$

Dies ist wiederum äquivalent mit der Existenz eines Vektors y mit der Eigenschaft, dass $c(e) + y_v - y_w \geq 0$ für alle residuellen Kanten $e = (v, w) \in E(G_x)$, d.h. mit der Existenz eines zulässigen Potenzials y für (G_x, c) . \square

9.3 Der Minimum-Mean-Cycle-Cancelling-Algorithmus

Satz 9.7 legt bereits einen Algorithmus nahe: Finde zunächst einen b -Fluss (benutze einen Maximum-Flow-Algorithmus wie in Satz 9.2 beschrieben) und augmentiere dann schrittweise entlang augmentierenden Kreisen negativen Gewichtes bis keine mehr vorhanden sind. Wir müssen die Kreise jedoch mit Vorsicht wählen, wenn wir eine polynomielle Laufzeit erreichen wollen (siehe Aufgabe 7). Eine gute Strategie besteht darin, jedes Mal einen augmentierenden Kreis mit minimalem durchschnittlichem Kantengewicht zu wählen:

MINIMUM-MEAN-CYCLE-CANCELLING-ALGORITHMUS

Input: Ein Digraph G , Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$, Zahlen $b : V(G) \rightarrow \mathbb{R}$ mit $\sum_{v \in V(G)} b(v) = 0$ und Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Output: Ein b -Fluss f mit minimalen Kosten.

- ① Bestimme einen b -Fluss f .
- ② Bestimme einen Kreis C in G_f mit minimalem durchschnittlichem Kantengewicht.
If C hat nichtnegatives Gesamtgewicht (oder G_f ist azyklisch) **then stop**.
- ③ Berechne $\gamma := \min_{e \in E(C)} u_f(e)$. Augmentiere f entlang C um γ .
Go to ②.

Wie im Beweis von Satz 9.2 beschrieben, kann ① mit jedem Algorithmus für das MAXIMUM-FLOW-PROBLEM implementiert werden. Schritt ② kann mit dem in Abschnitt 7.3 beschriebenen Algorithmus implementiert werden. Wir werden nun beweisen, dass dieser Algorithmus nach einer polynomiellen Anzahl von Iterationen terminiert. Der Beweis verläuft ähnlich wie der in Abschnitt 8.3. Bezeichnen wir mit $\mu(f)$ das minimale durchschnittliche Kantengewicht eines Kreises in G_f , so besagt Satz 9.7, dass ein b -Fluss f genau dann optimal ist, wenn $\mu(f) \geq 0$.

Zunächst zeigen wir, dass $\mu(f)$ während des gesamten Verlaufs des Algorithmus nicht abnimmt. Wir können sogar zeigen, dass $\mu(f)$ mit je $|E(G)|$ Iterationen streng wächst. Wie üblich, bezeichnen wir mit n bzw. m die Knoten- bzw. Kantenanzahl von G .

Lemma 9.9. *Sei (G, u, b, c) eine Instanz des MINIMUM-COST-FLOWS. Sei f_1, f_2, \dots, f_t eine Folge von b -Flüssen, so dass für alle $i = 1, \dots, t-1$ gilt: Es ist $\mu(f_i) < 0$, und f_{i+1} geht aus f_i durch Augmentierung entlang C_i hervor, wobei C_i ein Kreis mit minimalem durchschnittlichem Kantengewicht in G_{f_i} ist. Dann folgt:*

- (a) $\mu(f_k) \leq \mu(f_{k+1})$ für alle k .
- (b) $\mu(f_k) \leq \frac{n}{n-2} \mu(f_l)$ für alle $k < l$ mit der Eigenschaft, dass $C_k \cup C_l$ ein Paar gegenläufiger Kanten enthält.

Beweis: (a): Seien f_k, f_{k+1} zwei aufeinander folgende Flüsse in dieser Folge. Betrachte den eulerschen Graphen H , der aus $(V(G), E(C_k) \dot{\cup} E(C_{k+1}))$ durch das Entfernen von Paaren gegenläufiger Kanten hervorgeht. (Kanten, die sowohl in C_k als auch in C_{k+1} vorkommen, werden zweimal gezählt.) Jeder einfache Teilgraph von H ist ein Teilgraph von G_{f_k} , da jede Kante in $E(G_{f_{k+1}}) \setminus E(G_{f_k})$ die gegenläufige Kante einer Kante in $E(C_k)$ ist. Da H eulersch ist, kann H in Kreise zerlegt werden, und jeder dieser Kreise hat ein durchschnittliches Kantengewicht von mindestens $\mu(f_k)$. Folglich gilt $c(E(H)) \geq \mu(f_k)|E(H)|$.

Da das Gesamtgewicht eines jeden Paares gegenläufiger Kanten verschwindet, haben wir

$$c(E(H)) = c(E(C_k)) + c(E(C_{k+1})) = \mu(f_k)|E(C_k)| + \mu(f_{k+1})|E(C_{k+1})|.$$

Da $|E(H)| \leq |E(C_k)| + |E(C_{k+1})|$, folgt

$$\begin{aligned} \mu(f_k)(|E(C_k)| + |E(C_{k+1})|) &\leq \mu(f_k)|E(H)| \\ &\leq c(E(H)) \\ &= \mu(f_k)|E(C_k)| + \mu(f_{k+1})|E(C_{k+1})|, \end{aligned}$$

und somit gilt $\mu(f_{k+1}) \geq \mu(f_k)$.

(b): Nach (a) genügt es, die Aussage für diejenigen k, l zu beweisen, für die $C_i \cup C_l$ mit $k < i < l$ keine Paare gegenläufiger Kanten enthält.

Wie in dem Beweis von (a), betrachten wir den eulerschen Graphen H , der aus $(V(G), E(C_k) \dot{\cup} E(C_l))$ durch das Entfernen von Paaren gegenläufiger Kanten

hervorgeht. Jeder einfache Teilgraph von H ist ein Teilgraph von G_{f_k} , da jede Kante in $E(C_l) \setminus E(G_{f_k})$ die gegenläufige Kante einer Kante in einem der Kreise $C_k, C_{k+1}, \dots, C_{l-1}$ ist. Nach der besonderen Wahl von k und l folgt aber, dass nur der Kreis C_k unter diesen Kreisen die gegenläufige Kante einer Kante in C_l enthält.

Somit folgt wie in (a), dass $c(E(H)) \geq \mu(f_k)|E(H)|$ und

$$c(E(H)) = \mu(f_k)|E(C_k)| + \mu(f_l)|E(C_l)|.$$

Da $|E(H)| \leq |E(C_k)| + \frac{n-2}{n}|E(C_l)|$ (wir haben mindestens zwei Kanten entfernt), haben wir

$$\begin{aligned} \mu(f_k) \left(|E(C_k)| + \frac{n-2}{n} |E(C_l)| \right) &\leq \mu(f_k)|E(H)| \\ &\leq c(E(H)) \\ &= \mu(f_k)|E(C_k)| + \mu(f_l)|E(C_l)|, \end{aligned}$$

woraus $\mu(f_k) \leq \frac{n}{n-2} \mu(f_l)$ folgt. \square

Korollar 9.10. *Während des Verlaufs des MINIMUM-MEAN-CYCLE-CANCELING-ALGORITHMUS verringert sich $|\mu(f)|$ alle mn Iterationen um mindestens den Faktor 2.*

Beweis: Seien $C_k, C_{k+1}, \dots, C_{k+m}$ die augmentierenden Kreise in aufeinander folgenden Iterationen des Algorithmus. Da jeder dieser Kreise eine bestimmte Kante als Bottleneck-Kante enthält (eine anschließend aus dem Residualgraphen entfernte Kante), gibt es zwei Kreise unter diesen, etwa C_i und C_j mit $k \leq i < j \leq k+m$, deren Vereinigung ein Paar gegenläufiger Kanten enthält. Nach Lemma 9.9 folgt dann

$$\mu(f_k) \leq \mu(f_i) \leq \frac{n}{n-2} \mu(f_j) \leq \frac{n}{n-2} \mu(f_{k+m}).$$

Somit verringert sich $|\mu(f)|$ alle m Iterationen um mindestens den Faktor $\frac{n}{n-2}$. Hiermit folgt das Korollar, da $\left(\frac{n}{n-2}\right)^n > e^2 > 2$. \square

Damit ist bereits bewiesen, dass der Algorithmus in polynomieller Zeit läuft, wenn alle Kantenkosten ganzzahlig sind: Zu Beginn ist $|\mu(f)|$ höchstens $|c_{\min}|$, wobei c_{\min} gleich dem Minimum aller Kantenkosten ist, und $|\mu(f)|$ verringert sich alle mn Iterationen um mindestens den Faktor 2 (oder $\mu(f)$ wird positiv und der Algorithmus terminiert). Nach $O(mn \log(n|c_{\min}|))$ Iterationen ist $\mu(f)$ also größer als $-\frac{1}{n}$. Sind die Kantenkosten ganzzahlig, so folgt $\mu(f) \geq 0$ und der Algorithmus terminiert. Somit haben wir nach Korollar 7.13 eine $O(m^2 n^2 \log(n|c_{\min}|))$ -Laufzeit.

Mehr noch: Wir können eine streng polynomiale Laufzeit für das MINIMUM-COST-FLOW-PROBLEM erhalten (dies gelang zuerst Tardos [1985]):

Satz 9.11. (Goldberg und Tarjan [1989]) *Der MINIMUM-MEAN-CYCLE-CANCELLING-ALGORITHMUS läuft in $O(m^3 n^2 \log n)$ -Zeit.*

Beweis: Wir werden zeigen, dass nach je $mn(\lceil \log n \rceil + 1)$ Iterationen mindestens eine Kante fest bleibt, d. h., dass sich der Fluss auf dieser Kante letztmalig ändert. Somit gibt es höchstens $O(m^2 n \log n)$ Iterationen. Mit Satz 9.2 für ① und Korollar 7.13 für ② folgt der Satz.

Sei f der Fluss bei einer bestimmten Iteration und f' der Fluss $mn(\lceil \log n \rceil + 1)$ Iterationen später. Definiere Gewichte c' durch $c'(e) := c(e) - \mu(f')$ ($e \in E(G_{f'})$). Sei π ein zulässiges Potenzial von $(G_{f'}, c')$ (welches nach Satz 7.7 existiert). Es gilt $0 \leq c'_\pi(e) = c_\pi(e) - \mu(f')$ und damit

$$c_\pi(e) \geq \mu(f') \quad \text{für alle } e \in E(G_{f'}). \quad (9.3)$$

Nun sei C der Kreis mit minimalem durchschnittlichem Kantengewicht in G_f , welcher im Algorithmus gewählt wird, um f zu augmentieren. Da nach Korollar 9.10

$$\mu(f) \leq 2^{\lceil \log n \rceil + 1} \mu(f') \leq 2n\mu(f')$$

gilt (siehe Abb. 9.2), haben wir

$$\sum_{e \in E(C)} c_\pi(e) = \sum_{e \in E(C)} c(e) = \mu(f)|E(C)| \leq 2n\mu(f')|E(C)|.$$

Sei also $e_0 \in E(C)$ mit $c_\pi(e_0) \leq 2n\mu(f')$. Mit (9.3) folgt $e_0 \notin E(G_{f'})$.



Abbildung 9.2.

Behauptung: Für jeden b -Fluss f'' mit $e_0 \in E(G_{f''})$ gilt $\mu(f'') < \mu(f')$.

Nach Lemma 9.9(a) folgt aus der Behauptung, dass e_0 niemals mehr im Residualgraphen sein wird, d. h., dass e_0 und $\overset{\leftarrow}{e_0}$ beide nach $mn(\lceil \log n \rceil + 1)$ Iterationen nachdem e_0 in C gebraucht worden ist, fest bleiben. Damit ist der Beweis zu Ende.

Zum Beweis der Behauptung, sei f'' ein b -Fluss mit $e_0 \in E(G_{f''})$. Wenden wir Proposition 9.4 auf f' und f'' an, so erhalten wir eine Zirkulation g mit $g(e) = 0$ für alle $e \notin E(G_{f'})$ und mit $g(\overset{\leftarrow}{e_0}) > 0$ (da $e_0 \in E(G_{f''}) \setminus E(G_{f'})$).

Nach Proposition 9.5 kann g als Summe von Flüssen in f' -augmentierenden Kreisen geschrieben werden. Einer dieser Kreise, etwa W , enthält $\overset{\leftarrow}{e_0}$. Unter Benutzung von $c_\pi(\overset{\leftarrow}{e_0}) = -c_\pi(e_0) \geq -2n\mu(f')$ und unter Anwendung von (9.3) auf alle $e \in E(W) \setminus \{\overset{\leftarrow}{e_0}\}$ erhalten wir eine untere Schranke für das Gesamtgewicht von W :

$$c(E(W)) = \sum_{e \in E(W)} c_\pi(e) \geq -2n\mu(f') + (n-1)\mu(f') > -n\mu(f').$$

Ersetzen wir jede Kante von W durch ihre gegenläufige Kante, so bekommen wir einen f'' -augmentierenden Kreis (wie man durch Vertauschung der Rollen

von f' und f'' sieht), der ein Gesamtgewicht von weniger als $n\mu(f')$ hat. Dies bedeutet, dass $G_{f''}$ einen Kreis mit durchschnittlichem Kantengewicht kleiner als $\mu(f')$ enthält, womit die Behauptung bewiesen ist. \square

Der MINIMUM-MEAN-CYCLE-CANCELLING-ALGORITHMUS wurde von Karzanov und McCormick [1997] auf verschiedene Weisen verallgemeinert.

9.4 Der Sukzessive-Kürzeste-Wege-Algorithmus

Der folgende Satz führt zu einem weiteren Algorithmus:

Satz 9.12. (Jewell [1958], Iri [1960], Busacker und Gowen [1961]) *Sei (G, u, b, c) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS und f ein b -Fluss mit minimalen Kosten. Sei P ein kürzester (bezüglich c) s - t -Weg in G_f (für irgendwelche s und t). Sei f' ein durch Augmentierung von f entlang P um höchstens die minimale Residualkapazität auf P entstehender Fluss. Dann ist f' ein b' -Fluss (für eine geeignete Wahl von b') mit minimalen Kosten.*

Beweis: Es ist f' ein b' -Fluss für eine geeignete Wahl von b' . Angenommen, es sei f' kein b' -Fluss mit minimalen Kosten. Nach Satz 9.7 gibt es dann einen Kreis C in $G_{f'}$ mit negativem Gesamtgewicht. Betrachte den Graphen H , der aus $(V(G), E(C) \cup E(P))$ durch das Entfernen von Paaren gegenläufiger Kanten hervorgeht. (Wiederum werden Kanten, die sowohl in C als auch in P vorkommen, zweimal gezählt.)

Für jede Kante $e \in E(G_{f'}) \setminus E(G_f)$ liegt die gegenläufige Kante von e in $E(P)$. Also ist jeder einfache Teilgraph von H ein Teilgraph von G_f . Da f ein b -Fluss mit minimalen Kosten ist, hat kein Kreis in G_f und somit auch kein Kreis in H negatives Gesamtgewicht.

Es gilt $c(E(H)) = c(E(C)) + c(E(P)) < c(E(P))$. Ferner ist H die Vereinigung eines s - t -Weges und einigen Kreisen. Keine dieser Kreise hat negatives Gesamtgewicht.

Also enthält H , und somit auch G_f , einen s - t -Weg mit kleinerem Gewicht als P , im Widerspruch zu der Wahl von P . \square

Sind die Gewichte konservativ, so können wir mit $f \equiv 0$ als optimaler Zirkulation (b -Fluss mit $b \equiv 0$) beginnen. Sonst können wir vorab alle Kanten mit negativen Kosten saturieren, d.h. setze $f(e) := u(e)$ für $e \in F := \{e' \in E(G) : c(e') < 0\}$, setze $f(e) := 0$ für $e \in E(G) \setminus F$, und bestimme einen b' -Fluss in (G_f, u_f) mit minimalen Kosten, wobei $b'(v) = b(v) + \text{ex}_f(v)$ für alle $v \in V(G)$. Treten unendliche Kapazitäten auf, so ist der Aufwand größer, siehe Aufgabe 5.

SUKZESSIVE-KÜRZESTE-WEGE-ALGORITHMUS

Input: Ein Digraph G , Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$, Zahlen $b : V(G) \rightarrow \mathbb{R}$ mit $\sum_{v \in V(G)} b(v) = 0$ und konservative Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Output: Ein b -Fluss f mit minimalen Kosten.

- ① Setze $b' := b$ und $f(e) := 0$ für alle $e \in E(G)$.
- ② **If** $b' = 0$ **then stop**, **else**:
Wähle einen Knoten s mit $b'(s) > 0$.
Wähle einen Knoten t mit $b'(t) < 0$, so dass t von s aus in G_f erreichbar ist.
If es gibt kein solches t **then stop**. (Es gibt keinen b -Fluss.)
- ③ Bestimme einen s - t -Weg P in G_f mit minimalem Gewicht.
- ④ Berechne $\gamma := \min \left\{ \min_{e \in E(P)} u_f(e), b'(s), -b'(t) \right\}$.
Setze $b'(s) := b'(s) - \gamma$ und $b'(t) := b'(t) + \gamma$. Augmentiere f entlang P um γ .
Go to ②.

Lassen wir beliebige Kapazitäten zu, so stellen sich dieselben Probleme wie bei dem FORD-FULKERSON-ALGORITHMUS ein (siehe Aufgabe 2, Kapitel 8; setze alle Kosten auf Null). Somit werden wir von nun an u und b als ganzzahlig voraussetzen. Dann haben wir

Satz 9.13. Für gegebene ganze Zahlen u und b arbeitet der SUKZESSIVE-KÜRZESTE-WEGE-ALGORITHMUS korrekt und terminiert nach höchstens $B := \frac{1}{2} \sum_{v \in V(G)} |b(v)|$ Iterationen.

Beweis: Die Schranke für die Anzahl von Iterationen folgt sofort. Der Algorithmus erhält für alle $v \in V(G)$ die Invariante $b(v) = b'(v) - \text{ex}_f(v)$ aufrecht. Nach Satz 9.12 ist der resultierende Fluss optimal, wenn der verschwindende Anfangsfluss optimal war. Dies gilt genau dann, wenn c konservativ ist.

Entscheidet der Algorithmus, dass es keinen b -Fluss gibt, so stimmt das auch: Sei R die Menge derjenigen Knoten, die von einem Knoten s mit $b'(s) > 0$ in G_f aus erreichbar sind, und angenommen, dass $b'(v) \geq 0$ für alle $v \in R$. Dann gilt $b(R) = \sum_{v \in R} (b'(v) - \text{ex}_f(v)) > f(\delta^+(R)) - f(\delta^-(R)) = u(\delta^+(R))$, also gibt es nach Gales Satz 9.2 keinen b -Fluss. \square

Jede Augmentierung benötigt eine Kürzeste-Wege-Berechnung. Da negative Gewichte vorkommen können, müssen wir den MOORE-BELLMAN-FORD-ALGORITHMUS mit $O(nm)$ -Laufzeit anwenden (Satz 7.5), somit resultiert eine $O(Bnm)$ Gesamlaufzeit. Wie im Beweis von Satz 7.8 kann man es jedoch einrichten, dass (außer am Anfang) die kürzesten Wege in einem Graphen mit nichtnegativen Gewichten berechnet werden:

Satz 9.14. (Tomizawa [1971], Edmonds und Karp [1972]) *Sind u und b ganz-zahlig, so kann der SUKZESSIVE-KÜRZESTE-WEGE-ALGORITHMUS mit einer Laufzeit von $O(nm + B(m + n \log n))$ implementiert werden, wobei $B = \frac{1}{2} \sum_{v \in V(G)} |b(v)|$.*

Beweis: Wir führen für jede Iteration i des SUKZESSIVE-KÜRZESTE-WEGE-ALGORITHMUS ein Potenzial $\pi_i : V(G) \rightarrow \mathbb{R}$ ein. Wir beginnen mit irgendeinem zulässigen Potenzial π_0 von (G, c) . Nach Korollar 7.7 gibt es ein solches, und es kann in $O(mn)$ -Laufzeit berechnet werden.

Sei f_{i-1} der Fluss vor der Iteration i . Die Kürzeste-Wege-Berechnung in Iteration i erfolgt mit den reduzierten Kosten $c_{\pi_{i-1}}$ statt c . Ferner erweitern wir $G_{f_{i-1}}$ auf $G'_{f_{i-1}}$ durch Hinzufügung der Kanten (t, v) mit verschwindenden reduzierten Kosten für alle $v \in V(G) \setminus \{s, t\}$ (dies gewährleistet, dass alle Knoten von s aus erreichbar sind). Sei $l_i(v)$ für $v \in V(G)$ die Länge eines kürzesten $s-v$ -Wege in $G'_{f_{i-1}}$ bezüglich der Gewichte $c_{\pi_{i-1}}$. Dann setzen wir $\pi_i(v) := \pi_{i-1}(v) + l_i(v)$.

Wir beweisen nun mittels Induktion über i , dass π_i ein zulässiges Potenzial für (G_{f_i}, c) ist. Dies ist klar für $i = 0$. Für $i > 0$ und jede Kante $e \in E(G)$ haben wir

$$\begin{aligned} c_{\pi_i}(e) &= c(e) + \pi_i(x) - \pi_i(y) = c(e) + \pi_{i-1}(x) + l_i(x) - \pi_{i-1}(y) - l_i(y) \\ &= c_{\pi_{i-1}}(e) + l_i(x) - l_i(y). \end{aligned}$$

Für jede Kante $e = (x, y) \in E(G_{f_{i-1}})$ folgt (nach Definition von l_i und der Induktionsvoraussetzung) $l_i(y) \leq l_i(x) + c_{\pi_{i-1}}(e)$ und somit $c_{\pi_i}(e) \geq 0$. Für jede Kante $e = (x, y) \in P_i$ (wobei P_i der augmentierende Weg in Iteration i ist) folgt $l_i(y) = l_i(x) + c_{\pi_{i-1}}(e)$ (da P_i ein kürzester Weg ist) und somit $c_{\pi_i}(\overset{\leftrightarrow}{e}) = -c_{\pi_i}(e) = 0$. Da jede Kante in $E(G_{f_i}) \setminus E(G_{f_{i-1}})$ die gegenläufige Kante einer Kante in P_i ist, folgt, dass c_{π_i} in der Tat eine nichtnegative Gewichtsfunktion auf $E(G_{f_i})$ ist.

Wir weisen auf folgende Tatsache hin: In Iteration i sind die kürzesten $s-t$ -Wege in $G_{f_{i-1}}$ bezüglich c genau die kürzesten $s-t$ -Wege in $G'_{f_{i-1}}$ bezüglich $c_{\pi_{i-1}}$, da die hinzugefügten in t beginnenden Kanten nicht vorkommen können und $c_{\pi_{i-1}}(P) - c(P) = \pi_{i-1}(s) - \pi_{i-1}(t)$ für jeden $s-t$ -Weg P gilt.

Also können wir für alle Kürzeste-Wege-Berechnungen, außer der ersten, DIJKSTRAS ALGORITHMUS benutzen – dieser läuft nach Satz 7.4 in $O(m + n \log n)$ -Zeit wenn er mit einem Fibonacci-Heap implementiert wird. Da wir höchstens B Iterationen haben, erhalten wir eine Gesamlaufzeit von $O(nm + B(m + n \log n))$. \square

Die Laufzeit in Satz 9.14 ist weiterhin exponentiell, außer wenn man weiß, dass B klein ist. Ist $B = O(n)$, so ist dies der schnellste bekannte Algorithmus. Eine Anwendung befindet sich in Abschnitt 11.1.

Im restlichen Teil dieses Abschnitts zeigen wir, wie man den Algorithmus modifizieren kann, um die Anzahl der Kürzeste-Wege-Berechnungen zu reduzieren. Dazu betrachten wir nur den Fall unendlicher Kapazitäten. Nach Lemma 9.3 kann

jede Instanz des MINIMUM-COST-FLOW-PROBLEMS in eine äquivalente Instanz mit unendlichen Kapazitäten transformiert werden.

Die von Edmonds und Karp [1972] stammende grundlegende Idee ist die folgende. In den frühen Iterationen betrachten wir nur augmentierende Wege mit γ (d.h. die Flussmenge, um die augmentiert werden kann) groß. Wir beginnen mit $\gamma = 2^{\lfloor \log b_{\max} \rfloor}$ und reduzieren γ um den Faktor zwei, wenn keine weiteren Augmentierungen um γ mehr möglich sind. Nach $\lfloor \log b_{\max} \rfloor + 1$ Iterationen haben wir $\gamma = 1$ und terminieren den Algorithmus (wiederum setzen wir b als ganzzahlig voraus). Eine solche Skalierungstechnik hat sich bei vielen Algorithmen als nützlich erwiesen (siehe auch Aufgabe 13). Im Detail lautet der erste Skalierungsalgorithmus folgendermaßen:

KAPAZITÄTS-SKALIERUNGS-ALGORITHMUS

Input: Ein Digraph G mit unendlichen Kapazitäten $u(e) = \infty$ ($e \in E(G)$), Zahlen $b : V(G) \rightarrow \mathbb{Z}$ mit $\sum_{v \in V(G)} b(v) = 0$ und konservative Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Output: Ein b -Fluss f mit minimalen Kosten.

- ① Setze $b' := b$ und $f(e) := 0$ für alle $e \in E(G)$.
Setze $\gamma = 2^{\lfloor \log b_{\max} \rfloor}$, wobei $b_{\max} = \max\{b(v) : v \in V(G)\}$.
- ② **If** $b' = 0$ **then stop**, **else**:
Wähle Knoten s und t mit $b'(s) \geq \gamma$ und $b'(t) \leq -\gamma$, so dass t von s aus in G_f erreichbar ist.
If es gibt kein solches Paar (s, t) **then go to** ⑤.
Bestimme einen s - t -Weg P in G_f mit minimalem Gewicht.
- ④ Setze $b'(s) := b'(s) - \gamma$ und $b'(t) := b'(t) + \gamma$. Augmentiere f entlang P um γ .
Go to ②.
- ⑤ **If** $\gamma = 1$ **then stop**. (Es gibt keinen b -Fluss.)
Else setze $\gamma := \frac{\gamma}{2}$ und **go to** ②.

Satz 9.15. (Edmonds und Karp [1972]) *Der KAPAZITÄTS-SKALIERUNGS-ALGORITHMUS löst das MINIMUM-COST-FLOW-PROBLEM mit ganzzahligem b , unendlichen Kapazitäten und konservativen Gewichten korrekt. Er kann mit $O(n(m + n \log n) \log(2 + b_{\max}))$ -Laufzeit implementiert werden, wobei $b_{\max} = \max\{b(v) : v \in V(G)\}$.*

Beweis: Wie oben, folgt die Korrektheit sofort mit Satz 9.12. Beachte, dass die Residualkapazität einer jeden Kante zu jeder Zeit entweder unendlich oder ein ganzzahliges Vielfaches von γ ist.

Um die Laufzeit zu bestimmen, nennen wir das Zeitintervall, während dem γ konstant bleibt, eine *Phase*. Wir werden beweisen, dass es höchstens n Augmentierungen in jeder Phase gibt. Seien γ , f und b' die Werte am Anfang irgendeiner Phase. Setze $S := \{v \in V(G) : b'(v) \geq \gamma\}$, $S^+ := \{v \in V(G) : b'(v) \geq 2\gamma\}$,

$T := \{v \in V(G) : b'(v) \leq -\gamma\}$ und $T^+ := \{v \in V(G) : b'(v) \leq -2\gamma\}$. Sei R die Menge derjenigen Knoten, die von einem Element aus S^+ in G_f erreichbar sind. Beachte, dass $S^+ \subseteq R$. Ferner gilt $R \cap T^+ = \emptyset$, denn sonst hätte die vorige Phase nicht geendet.

Es seien (s_i, t_i) , $i = 1, \dots, k$, die Paare (s, t) in der betrachteten Phase; Wiederholungen sind natürlich möglich. Da $\delta_{G_f}^+(R) = \emptyset$, folgt $|i : s_i \in R, t_i \notin R| \leq |i : s_i \notin R, t_i \in R|$. Somit können wir die Anzahl k der Iterationen in dieser Phase wie folgt beschränken: $k \leq |i : s_i, t_i \in R| + 2|i : s_i \notin R, t_i \in R| + |i : s_i, t_i \notin R| = |i : t_i \in R| + |i : s_i \notin R| \leq |T \cap R| + |S \setminus R| \leq n$, wobei die zweitletzte Ungleichung deswegen gilt, weil $T^+ \cap R = \emptyset$ und $S^+ \setminus R = \emptyset$.

Dies bedeutet, dass die Gesamtanzahl der Kürzeste-Wege-Berechnungen $O(n \log(2 + b_{\max}))$ ist. In Verbindung mit der Methode des Satzes 9.14 bekommen wir die $O(mn + n \log(2 + b_{\max})(m + n \log n))$ Schranke. \square

Dies war der erste polynomielle Algorithmus für das MINIMUM-COST-FLOW-PROBLEM. Mit einigen weiteren Modifikationen können wir sogar eine stark polynomielle Laufzeit erreichen. Dies ist der Inhalt des nächsten Abschnittes.

9.5 Orlins Algorithmus

Der im vorigen Abschnitt besprochene KAPAZITÄTS-SKALIERUNGS-ALGORITHMUS kann noch verbessert werden. Eine grundlegende Idee in dieser Richtung ist die folgende: Wird eine Kante zu irgendeinem Zeitpunkt des KAPAZITÄTS-SKALIERUNGS-ALGORITHMUS mit mehr als $2ny$ Flusseinheiten belastet, so kann sie kontrahiert werden. Beachte hierzu, dass eine solche Kante immer mit einem positiven Fluss belastet ist (und somit Null reduzierte Kosten bezüglich jedes zulässigen Potenzials im Residualgraph aufweist): Es erfolgen höchstens n weitere Augmentierungen um γ , weitere n um $\frac{\gamma}{2}$ und so weiter; also bleibt der gesamte im restlichen Teil des Algorithmus bewegte Fluss unterhalb $2ny$.

Wir werden ORLINS ALGORITHMUS ohne expliziten Gebrauch von Kontraktionen beschreiben. Dies vereinfacht die Beschreibung, insbesondere hinsichtlich der Implementierung des Algorithmus. Mittels einer Menge F verbuchen wir diejenigen Kanten (und ihre gegenläufigen Kanten), welche kontrahiert werden können. Es wird $(V(G), F)$ immer aus einem Wald hervorgehen, indem man jede Kante in beide Richtungen orientiert. Aus jeder Zusammenhangskomponente von $(V(G), F)$ wird ein Vertreter gewählt, und der Algorithmus zerstört nicht die Eigenschaft, dass jeder dieser Vertreter der einzige nicht balancierte Knoten seiner Zusammenhangskomponente ist. Dies erhöht die Anzahl der Iterationen in jeder Phase, jedoch weiterhin um weniger als $4ny$.

Für jedes x bezeichne $r(x)$ den Vertreter derjenigen Zusammenhangskomponente von $(V(G), F)$, die x enthält. Ferner bezeichne G' denjenigen Teilgraphen von G , der die Kanten von F und alle Kanten (x, y) mit $r(x) \neq r(y)$ enthält. Kanten außerhalb von G' werden nicht mehr benutzt.

In ORLINS ALGORITHMUS wird b nicht als ganzzahlig vorausgesetzt. Er ist jedoch nur für Probleme ohne Kapazitäten geeignet (siehe aber Lemma 9.3).

ORLINS ALGORITHMUS

Input: Ein Digraph G mit unendlichen Kapazitäten $u(e) = \infty$ ($e \in E(G)$), Zahlen $b : V(G) \rightarrow \mathbb{R}$ mit $\sum_{v \in V(G)} b(v) = 0$ und konservative Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Output: Ein b -Fluss f mit minimalen Kosten.

- ① Setze $b' := b$ und $f(e) := 0$ für alle $e \in E(G)$.
Setze $r(v) := v$ für alle $v \in V(G)$. Setze $F := \emptyset$ und $G' := G$.
Setze $\gamma = \max_{v \in V(G)} |b'(v)|$.
- ② **If** $b' = 0$ **then stop.**
- ③ Wähle einen Knoten s mit $b'(s) > \frac{n-1}{n}\gamma$.
If es gibt kein solches s **then go to** ④.
Wähle einen Knoten t mit $b'(t) < -\frac{1}{n}\gamma$, so dass t von s aus in G_f erreichbar ist.
If es gibt kein solches t **then stop.** (Es gibt keinen b -Fluss.)
Go to ⑤.
- ④ Wähle einen Knoten t mit $b'(t) < -\frac{n-1}{n}\gamma$.
If es gibt kein solches t **then go to** ⑥.
Wähle einen Knoten s mit $b'(s) > \frac{1}{n}\gamma$, so dass t von s aus in G_f erreichbar ist.
If es gibt kein solches s **then stop.** (Es gibt keinen b -Fluss.)
- ⑤ Bestimme einen s - t -Weg P in G'_f mit minimalem Gewicht.
Setze $b'(s) := b'(s) - \gamma$ und $b'(t) := b'(t) + \gamma$. Augmentiere f entlang P um γ .
Go to ②.
- ⑥ **If** $f(e) = 0$ für alle $e \in E(G') \setminus F$ **then** setze $\gamma := \min \left\{ \frac{\gamma}{2}, \max_{v \in V(G)} |b'(v)| \right\}$,
else setze $\gamma := \frac{\gamma}{2}$.
- ⑦ **While** es gibt ein $e = (x, y) \in E(G') \setminus F$ mit $f(e) > 6n\gamma$ **do**:
 Setze $F := F \cup \{e, \overleftarrow{e}\}$.
 Sei $x' := r(x)$ und $y' := r(y)$. Sei Q der x' - y' -Weg in F .
 If $b'(x') > 0$ **then** augmentiere f entlang Q um $b'(x')$,
 else augmentiere f entlang dem gegenläufigen Weg des Weges Q um $-b'(x')$.
 Setze $b'(y') := b'(y') + b'(x')$ und $b'(x') := 0$.
 For alle $e' = (v, w) \in E(G') \setminus F$ mit $\{r(v), r(w)\} = \{x', y'\}$ **do**:
 Setze $E(G') := E(G') \setminus \{e'\}$.
 Setze $r(z) := y'$ für alle von y' aus in F erreichbaren Knoten z .
- ⑧ **Go to** ②.

Dieser Algorithmus stammt von Orlin [1993]. Siehe auch (Plotkin und Tardos [1990]). Zunächst werden wir seine Korrektheit beweisen. Dazu werden wir das zwischen zwei Änderungen von γ liegende Zeitintervall eine **Phase** nennen. Wir werden einen Knoten v **wichtig** nennen, falls $|b'(v)| > \frac{n-1}{n}\gamma$. Eine Phase endet wenn es keinen wichtigen Knoten gibt.

Lemma 9.16. *Die Anzahl der Augmentierungen in ⑤ während einer Phase ist höchstens gleich der Anzahl der wichtigen Knoten zu Beginn dieser Phase plus der Anzahl der Augmentierungen in ⑦ zu Beginn dieser Phase.*

Beweis: Jede Augmentierung in ⑤ verringert $\Phi := \sum_{v \in V(G)} \left\lceil \frac{|b'(v)|}{\gamma} - \frac{n-1}{n} \right\rceil$ um mindestens eins, während eine Augmentierung in ⑦ Φ nicht um mehr als eins erhöhen kann. Ferner ist Φ zu Beginn jeder Phase die Anzahl der wichtigen Knoten. \square

Lemma 9.17. *ORLINS ALGORITHMUS löst das MINIMUM-COST-FLOW-PROBLEM ohne Kapazitäten und mit konservativen Gewichten korrekt. Zu jedem Zeitpunkt ist f ein $(b - b')$ -Fluss mit minimalen Kosten.*

Beweis: Zunächst beweisen wir, dass f immer ein $(b - b')$ -Fluss ist. Insbesondere müssen wir zeigen, dass f_k immer nichtnegativ ist. Dazu beachten wir zunächst, dass der Fluss in jeder Kante von $E(G') \setminus F$, und somit auch die Residualkapazität der gegenläufigen Kante, zu jeder Zeit ein ganzzahliges Vielfaches von γ ist.

Ferner behaupten wir, dass jede Kante $e \in F$ immer eine positive Residualkapazität hat. Um dies zu sehen, bemerken wir zunächst, dass es, nachdem e in der γ -Phase ein Element von F wurde, höchstens $n - 1$ Augmentierungen in ⑦ gibt, die zusammen weniger als $2\gamma(n - 1)$ ergeben. Ferner hat jede Phase höchstens $2n - 1$ Augmentierungen um γ in ⑤ (siehe Lemma 9.16); somit ist der gesamte Fluss, welcher bewegt wird nachdem e in der γ -Phase ein Element von F wurde, weniger als $2n\gamma$ in ⑦, weniger als $2n\gamma$ in ⑤ während dieser Phase, weniger als $2n\frac{\gamma}{2}$ in ⑤ während der nächsten Phase, und so weiter, also weniger als $6n\gamma$ insgesamt. Somit besteht immer ein positiver Fluss in jeder Kante in $F \cap E(G)$.

Also ist f immer nichtnegativ und somit ist f immer ein $(b - b')$ -Fluss. Nun beweisen wir, dass f immer ein $(b - b')$ -Fluss mit minimalen Kosten ist. Dies gilt anfänglich da c konservativ ist. So lange f ein optimaler $(b - b')$ -Fluss ist, ist jeder $v-w$ -Weg in $(V(G), F)$ ein kürzester $v-w$ -Weg in G_f (ein kürzerer Weg würde einen negativen Kreis in G_f ergeben). Nun folgt die Behauptung aus Satz 9.12: Es sind P in ⑤ bzw. Q in ⑦ kürzeste Wege in G'_f bzw. $(V(G), F)$ und somit auch in G_f (beachte, dass es für jede Kante $(v, w) \in E(G_f) \setminus E(G'_f)$ einen $v-w$ -Weg in $(V(G), F)$ gibt).

Schließlich zeigen wir: Terminierte der Algorithmus in ③ oder ④ mit $b' \neq 0$, dann gibt es tatsächlich keinen b -Fluss. Angenommen, der Algorithmus terminiert in ③, d. h. es gibt einen Knoten s mit $b'(s) > \frac{n-1}{n}\gamma$, aber kein Knoten t mit $b'(t) < -\frac{1}{n}\gamma$ ist von s aus in G_f erreichbar. Sei R die Menge der von s aus in G_f erreichbaren Knoten. Da f ein $(b - b')$ -Fluss ist, folgt $\sum_{x \in R} (b(x) - b'(x)) = 0$. Also haben wir

$$\sum_{x \in R} b(x) = \sum_{x \in R} b'(x) = b'(s) + \sum_{x \in R \setminus \{s\}} b'(x) > 0,$$

aber $\delta_G^+(R) = \emptyset$. Damit ist bewiesen, dass es keinen b -Fluss gibt (siehe Satz 9.2). Der Fall, dass der Algorithmus in ④ terminiert, wird analog bewiesen. \square

Nun wenden wir uns der Laufzeitanalyse zu.

Lemma 9.18. (Plotkin und Tardos [1990]) *Ist ein Knoten z zu irgendeinem Zeitpunkt des Algorithmus zu Beginn einer Phase wichtig, so wächst die z enthaltende Zusammenhangskomponente von $(V(G), F)$ während der nächsten $\lceil 2 \log n + \log m \rceil + 3$ Phasen.*

Beweis: Sei $|b'(z)| > \frac{n-1}{n}\gamma_1$ für einen Knoten z am Anfang irgendeiner Phase des Algorithmus für die $\gamma = \gamma_1$ gilt. Sei γ_0 der Wert von γ in der vorigen Phase (wobei $\gamma_0 = 2\gamma_1$ falls es keine gibt), und γ_2 der Wert von γ nach $\lceil 2 \log n + \log m \rceil + 2$ weiteren Phasen. Es gilt $\frac{1}{2}\gamma_0 \geq \gamma_1 \geq 4n^2m\gamma_2$. Sei b'_1 bzw. f_1 das b' bzw. das f am Anfang der γ_1 -Phase, und sei b'_2 bzw. f_2 das b' bzw. das f am Ende der γ_2 -Phase.

Sei Z die in der γ_1 -Phase z enthaltende Zusammenhangskomponente von $(V(G), F)$ und angenommen, dass dies während der oben betrachteten $\lceil 2 \log n + \log m \rceil + 3$ Phasen so bleibt. Beachte: Nach ⑦ ist $b'(v) = 0$ für alle Knoten v mit $r(v) \neq v$. Somit ist $b'(v) = 0$ für alle $v \in Z \setminus \{z\}$ und wir haben

$$\sum_{x \in Z} b(x) - b'_1(z) = \sum_{x \in Z} (b(x) - b'_1(x)) = \sum_{e \in \delta^+(Z)} f_1(e) - \sum_{e \in \delta^-(Z)} f_1(e). \quad (9.4)$$

Die rechte Seite ist ein ganzzahliges Vielfaches von γ_0 und

$$\frac{1}{n}\gamma_1 \leq \frac{n-1}{n}\gamma_1 < |b'_1(z)| \leq \frac{n-1}{n}\gamma_0 < \gamma_0 - \frac{1}{n}\gamma_1. \quad (9.5)$$

Damit folgt

$$\left| \sum_{x \in Z} b(x) \right| > \frac{1}{n}\gamma_1. \quad (9.6)$$

Wie in (9.4) folgt $\sum_{e \in \delta^+(Z)} f_2(e) - \sum_{e \in \delta^-(Z)} f_2(e) = \sum_{x \in Z} b(x) - b'_2(z)$. Mit (9.6) und $|b'_2(z)| \leq \frac{n-1}{n}\gamma_2$ haben wir dann

$$\begin{aligned} \sum_{e \in \delta^+(Z) \cup \delta^-(Z)} |f_2(e)| &\geq \left| \sum_{x \in Z} b(x) \right| - |b'_2(z)| > \frac{1}{n}\gamma_1 - \frac{n-1}{n}\gamma_2 \\ &> (4nm - 1)\gamma_2 > m \left(6n \frac{\gamma_2}{2} \right). \end{aligned}$$

Somit gibt es mindestens eine Kante e mit genau einem Endknoten in Z und $f_2(e) > 6n \frac{\gamma_2}{2}$. Mit ⑦ des Algorithmus bedeutet dies, dass Z am Anfang der $\frac{\gamma_2}{2}$ -Phase wächst. \square

Satz 9.19. (Orlin [1993]) ORLINS ALGORITHMUS löst das MINIMUM-COST-FLOW-PROBLEM ohne Kapazitäten und mit konservativen Gewichten korrekt und hat eine $O(n \log n(m + n \log n))$ -Laufzeit.

Beweis: Die Korrektheit folgt nach Lemma 9.17. In jeder Phase braucht ⑦ $O(m(i+1))$ -Laufzeit, wobei i die Anzahl der Iterationen in der While-Schleife ist. Beachte, dass die Gesamtanzahl der Iterationen in dieser While-Schleife höchstens gleich $n - 1$ ist, da die Anzahl der Zusammenhangskomponenten von $(V(G), F)$ jedesmal abnimmt.

Mit ⑥ und ⑦ folgt, dass es höchstens $\lceil \log n \rceil + 3$ aufeinanderfolgende Phasen geben kann, bevor es einen wichtigen Knoten gibt oder man ein Kantenpaar zu F hinzufügt. Somit folgt aus Lemma 9.18, dass die Gesamtanzahl der Phasen $O(n \log m)$ ist.

Nach Lemma 9.16 ist die Gesamtanzahl der Augmentierungen in ⑤ höchstens $n - 1$ plus die Anzahl der Paare (γ, z) , wobei z am Anfang der γ -Phase wichtig ist. Mit Lemma 9.18 und da für alle Knoten v mit $r(v) \neq v$ zu jeder Zeit $b'(v) = 0$ gilt, folgt, dass die Anzahl dieser Paare gleich $O(\log m)$ mal die Anzahl derjenigen Mengen ist, die irgendwann während des Algorithmus Zusammenhangskomponenten von $(V(G), F)$ sind. Da die Familie solcher Mengen laminar ist, gibt es höchstens $2n - 1$ von ihnen (Korollar 2.15), also gibt es insgesamt $O(n \log m)$ Augmentierungen in ⑤.

Unter Anwendung der in Satz 9.14 gegebenen Methode erreichen wir eine Gesamtlaufzeit von $O(mn \log m + (n \log m)(m + n \log n))$. Hier können wir annehmen, dass $m = O(n^2)$ und somit $\log m = O(\log n)$, da wir aus einer Menge von unbeschränkten parallelen Kanten nur eine billigste benötigen. \square

Dies ist die beste bekannte Laufzeit für das MINIMUM-COST-FLOW-PROBLEM ohne Kapazitäten.

Satz 9.20. (Orlin [1993]) Das allgemeine MINIMUM-COST-FLOW-PROBLEM kann mit $O(m \log m(m + n \log n))$ Laufzeit gelöst werden, wobei $n = |V(G)|$ und $m = |E(G)|$.

Beweis: Hier wenden wir die in Lemma 9.3 angegebene Konstruktion an. Wir müssen also ein MINIMUM-COST-FLOW-PROBLEM ohne Kapazitäten auf einem bipartiten Graphen H mit $V(H) = A' \cup B'$ lösen, wobei $A' = E(G)$ und $B' = V(G)$. Da H azyklisch ist, kann man ein zulässiges Anfangspotenzial in $O(|E(H)|) = O(m)$ -Zeit berechnen. Wie oben gezeigt wurde (Satz 9.19), wird die Gesamtlaufzeit dominiert durch die $O(m \log m)$ Kürzeste-Wege-Berechnungen in einem Teilgraphen von $\overset{\leftrightarrow}{H}$ mit nichtnegativen Gewichten.

Bevor wir DIJKSTRAS ALGORITHMUS aufrufen, wenden wir die folgende Operation auf jeden Knoten $a \in A'$ an, der nicht ein Endknoten des von uns gesuchten Weges ist: Füge für jedes Kantenpaar $(b, a), (a, b')$ eine Kante (b, b') hinzu und setze ihr Gewicht auf die Summe der Gewichte der beiden Kanten (b, a) und (a, b') und entferne dann a . Offensichtlich bekommen wir eine äquivalente Instanz des KÜRZESTE-WEGE-PROBLEMS. Da jeder Knoten in A' mit vier Kanten in $\overset{\leftrightarrow}{H}$ inzident ist, hat der resultierende Graph $O(m)$ Kanten und höchstens $n + 2$

Knoten. Die Vorbearbeitung benötigt eine konstante Zeit pro Knoten, nämlich $O(m)$. Dasselbe gilt für die Transformation des kürzesten Weges in den zugehörigen \leftrightarrow Weg in H und für die Berechnung der Distanzmarkierungen der entfernten Knoten. Damit haben wir eine Gesamtlaufzeit von $O((m \log m)(m + n \log n))$. \square

Dies ist der schnellste bekannte streng polynomielle Algorithmus für das allgemeine MINIMUM-COST-FLOW-PROBLEM, solange es nicht bedeutend mehr als n^2 Kanten gibt (siehe Matsui und Scheifele [2016] zu weiteren Details). Ein Algorithmus mit derselben Laufzeit wie der von Orlin, der aber direkt auf Instanzen mit Kapazitäten angewendet wird, wurde von Vygen [2002] beschrieben. Ein schneller schwach polynomieller Algorithmus stammt von Lee und Sidford [2014].

9.6 Der Netzwerk-Simplexalgorithmus

Das MINIMUM-COST-FLOW-PROBLEM ist ein Spezialfall der LINEAREN OPTIMIERUNG. Durch Spezialisierung des SIMPLEXALGORITHMUS und Ausnutzung der besonderen Struktur erhalten wir den sogenannten NETZWERK-SIMPLEXALGORITHMUS. Um diesen Zusammenhang zu erläutern, charakterisieren wir zunächst die Menge der Basislösungen (obwohl wir dies für den Beweis des korrekten Verlaufs nicht benötigen).

Definition 9.21. Sei (G, u, b, c) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS. Ein b -Fluss f in (G, u) ist eine **Baumlösung**, falls $(V(G), \{e \in E(G) : 0 < f(e) < u(e)\})$ keine ungerichteten Kreise enthält.

Proposition 9.22. Eine Instanz des MINIMUM-COST-FLOW-PROBLEMS hat entweder eine optimale Lösung, die eine Baumlösung ist, oder gar keine optimale Lösung.

Beweis: Gegeben sei eine optimale Lösung f und ein ungerichteter Kreis C in $(V(G), \{e \in E(G) : 0 < f(e) < u(e)\})$. Wir haben zwei gerichtete Kreise C' und C'' in G_f mit demselben zugrunde liegenden ungerichteten Graphen wie C . Sei ϵ die minimale Residualkapazität in $E(C') \cup E(C'')$. Mittels Augmentierung von f um ϵ entlang C' bzw. C'' erhalten wir zwei weitere zulässige Lösungen f' und f'' . Da $2c(f) = c(f') + c(f'')$, sind f' und f'' beide auch optimale Lösungen. Mindestens eine von ihnen hat weniger Kanten e mit $0 < f'(e) < u(e)$ als f , somit erhalten wir nach weniger als $|E(G)|$ Schritten eine optimale Baumlösung. \square

Korollar 9.23. Sei (G, u, b, c) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS. Dann sind die Basislösungen von

$$\left\{ \begin{array}{l} x \in \mathbb{R}^{E(G)} : 0 \leq x_e \leq u(e) \ (e \in E(G)), \\ \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = b(v) \ (v \in V(G)) \end{array} \right\}$$

genau die Baumlösungen von (G, u, b, c) .

Beweis: Nach Proposition 9.22 ist jede Basislösung eine Baumlösung.

Gegeben sei eine Baumlösung f . Wir betrachten die Ungleichungen $x_e \geq 0$ für $e \in E(G)$ mit $f(e) = 0$, $x_e \leq u(e)$ für $e \in E(G)$ mit $f(e) = u(e)$, und $\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = b(v)$ für alle v außer einem Knoten pro Zusammenhangskomponente von $(V(G), \{e \in E(G) : 0 < f(e) < u(e)\})$. Diese $|E(G)|$ Ungleichungen werden alle von f mit Gleichheit erfüllt, und die diesen Ungleichungen entsprechende Untermatrix ist nichtsingulär. Somit ist f eine Basislösung. \square

In einer Baumlösung gibt es drei verschiedene Kantenarten: Kanten mit Null-Fluss, mit ausgelasteter Kapazität, oder mit positivem aber die Kapazität nicht auslastendem Fluss. Unter der Annahme, dass G zusammenhängend ist, können wir die Kantenmenge der dritten Kantenart zu einem zusammenhängenden aufspannenden Teilgraphen ohne ungerichtete Kreise erweitern, (d. h. zu einem orientierten aufspannenden Baum; daher der Name Baumlösung oder „spanning tree solution“).

Definition 9.24. Sei (G, u, b, c) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS mit G zusammenhängend. Eine **Baumstruktur** ist ein Quadrupel (r, T, L, U) , wobei $r \in V(G)$, $E(G) = T \cup L \cup U$, $|T| = |V(G)| - 1$ und $(V(G), T)$ keine ungerichteten Kreise enthält.

Der zu der Baumstruktur (r, T, L, U) gehörende b -Fluss wird wie folgt definiert:

- $f(e) := 0$ für $e \in L$,
- $f(e) := u(e)$ für $e \in U$,
- $f(e) := \sum_{v \in C_e} b(v) + \sum_{e' \in U \cap \delta^-(C_e)} u(e') - \sum_{e' \in U \cap \delta^+(C_e)} u(e')$ für $e \in T$, wobei C_e für $e = (v, w)$ die v enthaltende Zusammenhangskomponente von $(V(G), T \setminus \{e\})$ sei.

Die Baumstruktur (r, T, L, U) heiße **zulässig**, falls $0 \leq f(e) \leq u(e)$ für alle $e \in T$.

Eine Kante (v, w) in T heiße **abwärts**, falls v auf dem ungerichteten r - w -Weg in T liegt, sonst **aufwärts**. Die Baumstruktur (r, T, L, U) heiße **stark zulässig**, falls $0 < f(e) \leq u(e)$ für jede Abwärts-Kante $e \in T$ und $0 \leq f(e) < u(e)$ für jede Aufwärts-Kante $e \in T$.

Die eindeutig definierte Funktion $\pi : V(G) \rightarrow \mathbb{R}$ mit $\pi(r) = 0$ und $c_\pi(e) = 0$ für alle $e \in T$ heiße das zu der Baumstruktur (r, T, L, U) gehörende **Potenzial**.

Offensichtlich hat der zu einer Baumstruktur gehörende b -Fluss f die Eigenschaft $\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = b(v)$ für alle $v \in V(G)$ (obwohl f nicht immer ein zulässiger b -Fluss ist). Ferner haben wir:

Proposition 9.25. Für eine gegebene Instanz (G, u, b, c) des MINIMUM-COST-FLOW-PROBLEMS und eine Baumstruktur (r, T, L, U) kann der zu der Baumstruktur gehörende b -Fluss f bzw. das zu der Baumstruktur gehörende Potenzial π in $O(m)$ - bzw. $O(n)$ -Zeit berechnet werden.

Ferner gilt: Sind b und u ganzzahlig, so ist auch f ganzzahlig; ist c ganzzahlig, so ist auch π ganzzahlig.

Beweis: Das zu (r, T, L, U) gehörende Potenzial kann einfach berechnet werden, indem man den GRAPH-SCANNING-ALGORITHMUS auf die Kanten von T und ihre gegenläufigen Kanten anwendet. Der zu (r, T, L, U) gehörende b -Fluss kann in linearer Zeit berechnet werden, indem man die Knoten in einer Reihenfolge des nicht zunehmenden Abstands von r scannt. Die Ganzzahligkeitseigenschaften folgen direkt aus der Definition. \square

Der NETZWERK-SIMPLEXALGORITHMUS hält laufend eine gespeicherte, stark zulässige Baumstruktur bereit und arbeitet auf Optimalität zu. Beachte, dass aus dem Optimalitätskriterium von Korollar 9.8 sofort folgt:

Proposition 9.26. *Sei (r, T, L, U) eine zulässige Baumstruktur und π das zugehörende Potenzial. Wir nehmen nun an, dass*

- $c_\pi(e) \geq 0$ für alle $e \in L$ und
- $c_\pi(e) \leq 0$ für alle $e \in U$.

Dann hat (r, T, L, U) einen zugehörigen optimalen b -Fluss. \square

Beachte, dass $\pi(v)$ die Länge desjenigen r - v -Weges in $\overset{\leftrightarrow}{G}$ ist, der nur Kanten von T oder ihre gegenläufigen Kanten enthält. Für eine Kante $e = (v, w) \in E(\overset{\leftrightarrow}{G})$ sei der *Fundamentalkreis* C von e der Kreis, welcher aus e und demjenigen w - v -Weg besteht, der nur Kanten von T und ihre gegenläufigen Kanten enthält. Derjenige Knoten von C , der r in T am nächsten ist, heißt der *Gipfel* von C .

Für $e = (v, w) \notin T$ sind somit die Kosten, die bei der Beförderung einer Flusseinheit um den Fundamentalkreis von e entstehen, gleich $c_\pi(e) = c(e) + \pi(v) - \pi(w)$.

Es gibt mehrere Wege, um zu einer anfänglichen zulässigen Baumstruktur zu gelangen. Man könnte zum Beispiel einen b -Fluss berechnen (indem man ein MAXIMUM-FLOW-PROBLEM löst), dann das im Beweis von Proposition 9.22 angegebene Verfahren anwendet, ferner r beliebig wählt und T, L und U dem Fluss entsprechend definiert (unter eventuellem Hinzufügen geeigneter Kanten zu T). Andererseits könnte man auch „Phase Eins“ des SIMPLEXALGORITHMUS anwenden.

Dies führt jedoch nicht zwangsläufig zu einer starken zulässigen Baumstruktur. Am einfachsten ist es, wenn man zwischen r und jedem weiteren Knoten eine zusätzliche sehr teure Kante mit genügend Kapazität hinzufügt: Für jede Senke $v \in V(G) \setminus \{r\}$ fügen wir eine Kante (r, v) mit Kapazität $-b(v)$ hinzu und für jeden anderen Knoten $v \in V(G) \setminus \{r\}$ eine Kante (v, r) mit Kapazität $b(v)+1$. Die Kosten jeder dieser Zusatzkanten müssen so hoch sein, dass sie niemals in einer optimalen Lösung vorkommen, z. B. $1 + (|V(G)| - 1) \max_{e \in E(G)} |c(e)|$ (Aufgabe 19). Dann können wir als T die Menge aller Zusatzkanten nehmen, als L die Menge der ursprünglichen Kanten, und wir können $U := \emptyset$ setzen, um eine anfängliche stark zulässige Baumstruktur zu erhalten.

NETZWERK-SIMPLEXALGORITHMUS

Input: Eine Instanz (G, u, b, c) des MINIMUM-COST-FLOW-PROBLEMS und eine stark zulässige Baumstruktur (r, T, L, U) .

Output: Eine optimale Lösung f .

- ① Berechne den b -Fluss f und das zu (r, T, L, U) gehörende Potenzial π .
- ② Sei $e \in L$ mit $c_\pi(e) < 0$ oder $e \in U$ mit $c_\pi(e) > 0$.
If es gibt keine solche Kante e **then stop**.
- ③ Sei C der Fundamentalkreis von e (falls $e \in L$) oder von \overleftarrow{e} (falls $e \in U$).
Sei $\gamma := c_\pi(e)$.
- ④ Sei $\delta := \min_{e' \in E(C)} u_f(e')$ und e' die letzte Kante, in der dieses Minimum beim Durchlaufen von C in Orientierungsrichtung und im Gipfel beginnend angenommen wird.
Sei $e_0 \in E(G)$ mit der Eigenschaft: e' ist e_0 oder $\overleftarrow{e_0}$.
- ⑤ Entferne e aus L oder U .
Setze $T := (T \cup \{e\}) \setminus \{e_0\}$.
If $e' = e_0$ **then** füge e_0 zu U hinzu **else** füge e_0 zu L hinzu.
- ⑥ Augmentiere f um δ entlang C .
Sei X die r enthaltende Zusammenhangskomponente von $(V(G), T \setminus \{e\})$.
If $e \in \delta^+(X)$ **then** setze $\pi(v) := \pi(v) + \gamma$ für $v \in V(G) \setminus X$.
If $e \in \delta^-(X)$ **then** setze $\pi(v) := \pi(v) - \gamma$ für $v \in V(G) \setminus X$.
Go to ②.

Bemerke, dass man ⑥ einfach durch Rückkehr nach ① ersetzen kann, da f und π , wie in ⑥ berechnet, zu der neuen Baumstruktur gehören. Bemerke ferner, dass $e = e_0$ möglich ist; in diesem Fall ist $X = V(G)$, und es bleiben T und π unverändert, aber e geht von L nach U , oder umgekehrt, und $\delta = u(e)$.

Satz 9.27. (Dantzig [1951], Cunningham [1976]) *Der NETZWERK-SIMPLEXALGORITHMUS terminiert nach endlich vielen Iterationen und liefert eine optimale Lösung.*

Beweis: Zunächst bemerken wir, dass die Eigenschaft, dass f bzw. π der zu (r, T, L, U) gehörende b -Fluss bzw. das zu (r, T, L, U) gehörende Potenzial ist, in ⑥ erhalten bleibt.

Nun beweisen wir, dass die Baumstruktur immer stark zulässig ist. Nach der Wahl von δ bleibt die Bedingung $0 \leq f(e) \leq u(e)$ für alle e erhalten, somit bleibt die Baumstruktur zulässig.

Da auf keiner der Kanten des Teilweges von C , der von dem Knoten, in dem e' ankommt, bis zum Gipfel von C läuft, das Minimum in ④ angenommen wurde, werden diese Kanten nach der Augmentierung weiterhin positive Residualkapazität haben.

Für die Kanten des Teilweges von C , der vom Gipfel von C bis zu dem Knoten läuft, in dem e' beginnt, müssen wir sicherstellen, dass ihre gegenläufigen Kanten nach der Augmentierung noch positive Residualkapazität haben. Dies ist klar, falls $\delta > 0$. Andernfalls (falls $\delta = 0$) haben wir: Aus der Tatsache, dass die Baumstruktur vorher stark zulässig war, folgt, dass weder e noch \overleftarrow{e} diesem Teilweg angehören kann (d. h. $e = e_0$ oder $\delta^-(X) \cap E(C) \cap \{e, \overleftarrow{e}\} \neq \emptyset$), und dass die gegenläufigen Kanten des Teilweges von C , der vom Gipfel von C bis zu dem Knoten läuft, in dem e oder \overleftarrow{e} beginnt, positive Residualkapazität hatten.

Nach Proposition 9.26 ist der berechnete Fluss f bei Terminierung des Algorithmus optimal. Wir werden nun zeigen, dass es nicht zwei Iterationen mit demselben Paar (f, π) geben kann, somit kommt jede Baumstruktur höchstens einmal vor.

In jeder Iteration werden die Kosten des Flusses um $|\gamma| \delta$ reduziert. Da $\gamma \neq 0$, brauchen wir nur die Iterationen mit $\delta = 0$ zu betrachten. Hier bleiben die Kosten des Flusses konstant. Ist $e \neq e_0$, so ist $e \in L \cap \delta^-(X)$ oder $e \in U \cap \delta^+(X)$, also wächst $\sum_{v \in V(G)} \pi(v)$ streng (um mindestens $|\gamma|$). Ist schließlich $\delta = 0$ und $e = e_0$, so ist $u(e) = 0$, $X = V(G)$, π bleibt konstant, und $|\{e \in L : c_\pi(e) < 0\}| + |\{e \in U : c_\pi(e) > 0\}|$ fällt streng. Damit haben wir gezeigt, dass keine zwei Iterationen dieselbe Baumstruktur beinhalten. \square

Obwohl der NETZWERK-SIMPLEXALGORITHMUS nicht polynomiell ist, ist er in der Praxis recht effizient. Orlin [1997] hat eine Variante vorgeschlagen, die in polynomieller Zeit läuft. Polynomielle duale Netzwerk-Simplexalgorithmen wurden von Orlin, Plotkin und Tardos [1993] und von Armstrong und Jin [1997] beschrieben.

9.7 Zeitabhängige Flüsse

Wir kommen nun zur Betrachtung von zeitabhängigen Flüssen (auch als dynamische Flüsse bekannt), d. h. die Werte des Flusses entlang den Kanten können zeitlich variieren und der in eine Kante hineinfließende Fluss kommt erst nach einer vorgegebenen Zeitspanne am anderen Endpunkt der Kante an:

Definition 9.28. Sei (G, u, s, t) ein Netzwerk mit den Transitzeiten $l : E(G) \rightarrow \mathbb{R}_+$ und einem Zeithorizont $T \in \mathbb{R}_+$. Dann besteht ein **zeitabhängiger s-t-Fluss** f aus einer Lebesgue-messbaren Funktion $f_e : [0, T] \rightarrow \mathbb{R}_+$ für jedes $e \in E(G)$, mit $f_e(\tau) \leq u(e)$ für alle $\tau \in [0, T]$ und $e \in E(G)$, und

$$\text{ex}_f(v, a) := \sum_{e \in \delta^-(v)} \int_0^{\max\{0, a - l(e)\}} f_e(\tau) d\tau - \sum_{e \in \delta^+(v)} \int_0^a f_e(\tau) d\tau \geq 0 \quad (9.7)$$

für alle $v \in V(G) \setminus \{s\}$ und $a \in [0, T]$.

Die Zahl $f_e(\tau)$ ist die Flussrate des Flusses, der zum Zeitpunkt τ in e hineinfließt (und diese Kante $l(e)$ Zeiteinheiten später wieder verlässt). Die Ungleichung (9.7) erlaubt Zwischenlagerungen in Knoten, wie bei $s-t$ -Präflüssen. Es liegt nahe, den in die Senke t einfließenden Fluss zu maximieren:

MAXIMUM-FLOW-OVER-TIME-PROBLEM

Instanz: Ein Netzwerk (G, u, s, t) . Transitzeiten $l : E(G) \rightarrow \mathbb{R}_+$ und ein Zeithorizont $T \in \mathbb{R}_+$.

Aufgabe: Bestimme einen zeitabhängigen s - t -Fluss f mit maximalem Wert $\text{value}(f) := \text{ex}_f(t, T)$.

Ford und Fulkerson [1958] folgend, werden wir zeigen, dass dieses Problem auf das MINIMUM-COST-FLOW-PROBLEM zurückgeführt werden kann.

Satz 9.29. Das MAXIMUM-FLOW-OVER-TIME-PROBLEM kann mit Laufzeit $O(m \log m(m + n \log n))$ gelöst werden.

Beweis: Für eine gegebene Instanz (G, u, s, t, l, T) wie oben, definieren wir eine neue Kante $e' = (t, s)$ und den Graphen $G' := G + e'$. Setze $u(e') := u(E(G)) + 1$, $c(e') := -T$ und $c(e) := l(e)$ für $e \in E(G)$. Betrachte die Instanz $(G', u, 0, c)$ des MINIMUM-COST-FLOW-PROBLEMS. Sei f' eine optimale Lösung, d. h. eine Zirkulation in (G', u) mit minimalen Kosten bezüglich c . Nach Proposition 9.5 kann f' in Flüsse entlang Kreisen zerlegt werden, d. h. es gibt eine Menge \mathcal{C} von Kreisen in G' und positive Zahlen $h : \mathcal{C} \rightarrow \mathbb{R}_+$ mit $f'(e) = \sum\{h(C) : C \in \mathcal{C}, e \in E(C)\}$. Es gilt $c(E(C)) \leq 0$ für alle $C \in \mathcal{C}$, da f' eine Zirkulation mit minimalen Kosten ist.

Sei $C \in \mathcal{C}$ mit $c(E(C)) < 0$. Der Kreis C enthält e' . Für $e = (v, w) \in E(C) \setminus \{e'\}$ sei d_e^C die Distanz von s nach v in (C, c) . Setze

$$f_e^*(\tau) := \sum \left\{ h(C) : C \in \mathcal{C}, c(E(C)) < 0, e \in E(C), d_e^C \leq \tau \leq d_e^C - c(E(C)) \right\}$$

für $e \in E(G)$ und $\tau \in [0, T]$. Hiermit erhalten wir einen zeitabhängigen s - t -Fluss ohne Zwischenlagerungen (d. h. $\text{ex}_f(v, a) = 0$ für alle $v \in V(G) \setminus \{s, t\}$ und alle $a \in [0, T]$). Ferner gilt

$$\begin{aligned} \text{value}(f^*) &= \sum_{e \in \delta^-(t)} \int_0^{T-l(e)} f_e^*(\tau) d\tau \\ &= \sum_{C \in \mathcal{C}} h(C)(-c(E(C))) \\ &= - \sum_{e \in E(G')} c(e)f'(e). \end{aligned}$$

Sei $\pi(v) := T + \min\{0, \text{dist}_{(G'_f, c)}(t, v)\}$ für $v \in V(G)$. Wir haben $\pi(t) = T$, $\pi(s) = 0$ und $0 \leq \pi(v) \leq T$ für alle $v \in V(G)$.

Behauptung: π ist ein zulässiges Potenzial in (G'_f, c) .

Um dies zu beweisen, bemerken wir zunächst: Für alle $e = (x, y) \in E(G')$ mit $f'(e) > 0$ gilt $\text{dist}_{(G'_f, c)}(t, y) \leq 0$ und somit $\pi(y) = T + \text{dist}_{(G'_f, c)}(t, y)$. Also gilt für jedes $e = (v, w) \in E(G'_f)$ entweder $\pi(v) = T + \text{dist}_{(G'_f, c)}(t, v)$ oder

$(\pi(v) = T \text{ und } c(e) \geq 0)$. Im Fall (1) folgt $c_\pi(e) = c(e) + T + \text{dist}_{(G'_{f'}, c)}(t, v) - \pi(w) \geq c(e) + T + \text{dist}_{(G'_{f'}, c)}(t, v) - (T + \text{dist}_{(G'_{f'}, c)}(t, w)) \geq 0$, weil $G'_{f'}$ keine negativen Kreise enthält (siehe Satz 9.7). Im Fall (2) gilt $c_\pi(e) = c(e) + T - \pi(w) \geq c(e) + T - T = c(e) \geq 0$. Damit ist die Behauptung bewiesen.

Schließlich beweisen wir, dass f^* optimal ist. Dazu sei f irgendein zeitabhängiger s - t -Fluss. Setze $f_e(\tau) := 0$ für $e \in E(G)$ und $\tau \notin [0, T]$. Nach (9.7) gilt

$$\begin{aligned} \text{value}(f) &= \text{ex}_f(t, T) \\ &\leq \sum_{v \in V(G)} \text{ex}_f(v, \pi(v)) \\ &= \sum_{e=(v,w) \in E(G)} \left(\int_0^{\pi(w)-l(e)} f_e(\tau) d\tau - \int_0^{\pi(v)} f_e(\tau) d\tau \right) \\ &\leq \sum_{e=(v,w) \in E(G): \pi(w)-l(e) > \pi(v)} (\pi(w) - l(e) - \pi(v)) u(e) \\ &= \sum_{e=(v,w) \in E(G)} (\pi(w) - l(e) - \pi(v)) f'(e). \end{aligned}$$

Die letzte Gleichung gilt, da π ein zulässiges Potenzial in $(G'_{f'}, c)$ ist.

Daraus folgt nun, dass

$$\begin{aligned} \text{value}(f) &\leq \sum_{e=(v,w) \in E(G)} (\pi(w) - l(e) - \pi(v)) f'(e) \\ &= \sum_{e=(v,w) \in E(G')} (\pi(w) - c(e) - \pi(v)) f'(e) \\ &= - \sum_{e=(v,w) \in E(G')} c(e) f'(e) \\ &= \text{value}(f^*). \end{aligned}$$

□

Der Beweis zeigt auch: Es gibt stets einen optimalen zeitabhängigen Fluss mit stückweise konstanten Funktionen und mit durchgehend verschwindendem Überschuss an allen Knoten außer s und t .

Es gibt andere zeitabhängige Flussprobleme, die weitaus schwieriger sind. Hoppe und Tardos [2000] haben das sogenannte Schnellste-Transshipment-Problem (mit mehreren Quellen und Senken) mit ganzzahligen Transitzeiten mittels der Minimierung submodularer Funktionen (siehe Kapitel 14) gelöst. Die Bestimmung zeitabhängiger Flüsse mit minimalen Kosten ist *NP*-schwer (Klinz und Woeginger [2004]). Siehe auch Fleischer und Skutella [2007] für weitere Informationen und zum Thema Approximationsalgorithmen.

Aufgaben

- Man zeige, dass man das MAXIMUM-FLOW-PROBLEM als einen Spezialfall des MINIMUM-COST-FLOW-PROBLEMS auffassen kann.
- Sei G ein Digraph mit unteren und oberen Kapazitäten $l, u : E(G) \rightarrow \mathbb{R}_+$, wobei $l(e) \leq u(e)$ für alle $e \in E(G)$, und seien $b_1, b_2 : V(G) \rightarrow \mathbb{R}$ mit $b_1(v) \leq b_2(v)$ für alle $v \in V(G)$. Man beweise, dass es genau dann einen Fluss f mit $l(e) \leq f(e) \leq u(e)$ für alle $e \in E(G)$ und

$$b_1(v) \leq \sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) \leq b_2(v) \quad \text{für alle } v \in V(G)$$

gibt, wenn

$$\sum_{e \in \delta^+(X)} u(e) \geq \max \left\{ \sum_{v \in X} b_1(v), - \sum_{v \in V(G) \setminus X} b_2(v) \right\} + \sum_{e \in \delta^-(X)} l(e)$$

für alle $X \subseteq V(G)$.

Bemerkung: Diese Aufgabe verallgemeinert Aufgabe 4, Kapitel 8, und Satz 9.2.

(Hoffman [1960])

- Man beweise den folgenden Satz von Ore [1956]. Gegeben sei ein Digraph G und nichtnegative ganze Zahlen $a(x), b(x)$ für jedes $x \in V(G)$. Dann besitzt G einen aufspannenden Teilgraphen H mit $|\delta_H^+(x)| = a(x)$ und $|\delta_H^-(x)| = b(x)$ für alle $x \in V(G)$ genau dann, wenn

$$\sum_{x \in V(G)} a(x) = \sum_{x \in V(G)} b(x) \quad \text{und}$$

$$\sum_{x \in X} a(x) \leq \sum_{y \in V(G)} \min\{b(y), |\delta_G^+(X \setminus \{y\}) \cap \delta_G^-(y)|\} \quad \text{für alle } X \subseteq V(G).$$

(Ford und Fulkerson [1962])

- Sei (G, u, c, b) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS mit $c(e) \geq 0$ für alle $e \in E(G)$, und sei F die Menge derjenigen Knoten $e \in E(G)$, für die es eine optimale Lösung f mit $f(e) > 0$ gibt. Man zeige, dass jeder Kreis in $(V(G), F)$ nur aus Kanten e mit $c(e) = 0$ besteht.
- Man betrachte das MINIMUM-COST-FLOW-PROBLEM mit möglichen unendlichen Kapazitäten (d. h. $u(e) = \infty$ für manche Kanten e).
 - Man zeige, dass eine Instanz genau dann unbeschränkt ist, wenn sie zulässig ist und es einen negativen Kreis gibt, dessen Kanten alle unendliche Kapazität haben.
 - Man zeige, wie man in $O(n^3 + m)$ -Zeit entscheiden kann, ob eine Instanz unbeschränkt ist.

- (c) Man zeige, dass in einer nicht unbeschränkten Instanz jede unendliche Kapazität auf äquivalente Weise durch eine endliche Kapazität ersetzt werden kann.
- * 6. Sei (G, u, c, b) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS. Man nennt eine Funktion $\pi : V(G) \rightarrow \mathbb{R}$ ein optimales Potenzial, falls es einen b -Fluss f mit minimalen Kosten gibt, so dass π ein zulässiges Potenzial bezüglich (G_f, c) ist.
 - (a) Man beweise, dass eine Funktion $\pi : V(G) \rightarrow \mathbb{R}$ genau dann ein optimales Potenzial ist, wenn für jedes $X \subseteq V(G)$ die folgende Ungleichung gilt:

$$b(X) + \sum_{e \in \delta^-(X) : c_\pi(e) < 0} u(e) \leq \sum_{e \in \delta^+(X) : c_\pi(e) \leq 0} u(e).$$

- (b) Man zeige, wie man für eine gegebene Funktion $\pi : V(G) \rightarrow \mathbb{R}$ entweder eine die Bedingung in (a) nicht erfüllende Menge X findet, oder entscheidet, dass es keine solche gibt.
- (c) Man zeige, wie man für ein gegebenes optimales Potenzial einen b -Fluss mit minimalen Kosten in $O(n^3)$ -Zeit findet.

Bemerkung: Dies führt zu den sogenannten Cut-Cancelling-Algorithmen für das MINIMUM-COST-FLOW-PROBLEM.

(Hassin [1983])

- 7. Man betrachte das folgende Algorithmenschema für das MINIMUM-COST-FLOW-PROBLEM: Man finde zunächst irgendeinen b -Fluss, dann augmentiere man den Fluss entlang einem negativen augmentierenden Kreis (um die größtmögliche Menge), so lange es einen solchen Kreis noch gibt. In Abschnitt 9.3 wurde gezeigt, dass man eine streng polynomiale Laufzeit erreicht, wenn man immer einen Kreis mit minimalem durchschnittlichem Kantengewicht wählt. Man beweise, dass man ohne diese Vorgabe nicht sicher sein kann, dass der Algorithmus terminieren wird.

(Man verwende die in Aufgabe 2, Kapitel 8, angegebene Konstruktion.)

- 8. Man betrachte das in Aufgabe 2 beschriebene Problem mit der Gewichtsfunktion $c : E(G) \rightarrow \mathbb{R}$. Kann man einen die Bedingungen von Aufgabe 2 erfüllenden Fluss mit minimalen Kosten finden? (Man führe dieses Problem auf ein MINIMUM-COST-FLOW-PROBLEM zurück.)
- 9. Das GERICHTETE CHINESISCHE POSTBOTEN-PROBLEM kann folgendermaßen formuliert werden: Für einen gegebenen stark zusammenhängenden einfachen Digraphen G mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ finde man eine Funktion $f : E(G) \rightarrow \mathbb{N}$, so dass der Graph, welcher $f(e)$ Kopien von jeder Kante $e \in E(G)$ enthält, eulersch ist und $\sum_{e \in E(G)} c(e)f(e)$ minimiert. Wie löst man dieses Problem mit einem polynomiellen Algorithmus?
(Zum UNGERICHTETEN CHINESISCHEN POSTBOTEN-PROBLEM siehe Abschnitt 12.2.)
- * 10. Das GEBROCHENE b -MATCHING-PROBLEM wird folgendermaßen definiert: Gegeben sei ein ungerichteter Graph G , Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$, Zahlen $b : V(G) \rightarrow \mathbb{R}_+$ und Gewichte $c : E(G) \rightarrow \mathbb{R}$. Man finde eine Funktion

$f : E(G) \rightarrow \mathbb{R}_+$ mit $f(e) \leq u(e)$ für alle $e \in E(G)$ und $\sum_{e \in \delta(v)} f(e) \leq b(v)$ für alle $v \in V(G)$, die $\sum_{e \in E(G)} c(e)f(e)$ maximiert.

- (a) Man zeige, wie man dieses Problem durch Zurückführung auf ein MINIMUM-COST-FLOW-PROBLEM lösen kann.
- (b) Seien nun b und u ganzzahlig. Man zeige, dass das GEBROCHENE b -MATCHING-PROBLEM dann immer eine halbganzzahlige Lösung f hat (d. h. $2f(e) \in \mathbb{Z}$ für alle $e \in E(G)$).

Bemerkung: Das (ganzzahlige) MAXIMUM-WEIGHT- b -MATCHING-PROBLEM wird in Abschnitt 12.1 betrachtet.

- * 11. Man beschreibe einen polynomiellen kombinatorischen Algorithmus für das in Aufgabe 16, Kapitel 5 angegebene Intervall-Packungsproblem.
(Arkin und Silverberg [1987])
- 12. Man betrachte ein LP $\max\{cx : Ax \leq b\}$, wobei alle Elemente von A gleich $-1, 0$, oder 1 sind und jede Spalte von A höchstens eine 1 und höchstens eine -1 enthält. Man zeige, dass ein solches LP mit einer Instanz des MINIMUM-COST-FLOW-PROBLEMS äquivalent ist.
- 13. Die in den Abschnitten 8.4 und 9.4 eingeführte Skalierungstechnik kann in einem sehr allgemeinen Rahmen betrachtet werden: Sei Ψ eine Familie von Mengensystemen, die alle die leere Menge enthalten. Man nehme an, dass es einen Algorithmus zur Lösung des folgenden Problems gibt: Für ein gegebenes $(E, \mathcal{F}) \in \Psi$ mit Gewichten $c : E \rightarrow \mathbb{Z}_+$ und einer Menge $X \in \mathcal{F}$ finde man ein $Y \in \mathcal{F}$ mit $c(Y) > c(X)$ oder entscheide, dass es kein solches Y gibt. Angenommen, dieser Algorithmus hat eine in $\text{size}(c)$ polynomielle Laufzeit. Dann beweise man, dass es einen in $\text{size}(c)$ polynomiellen Algorithmus zur Bestimmung einer Menge $X \in \mathcal{F}$ mit maximalem Gewicht für gegebene $(E, \mathcal{F}) \in \Psi$ und $c : E \rightarrow \mathbb{Z}_+$ gibt.
(Grötschel und Lovász [1995]; siehe auch Schulz, Weismantel und Ziegler [1995] und Schulz und Weismantel [2002])
- 14. Man beweise, dass ORLINS ALGORITHMUS dahingehend präzisiert werden kann, dass er stets eine Baumlösung berechnet.
- 15. Man beweise, dass man in ⑦ von ORLINS ALGORITHMUS die $6n\gamma$ -Schranke auf $5n\gamma$ verbessern kann.
- 16. Man betrachte die Kürzeste-Wege-Berechnungen mit nichtnegativen Gewichten (mittels DIJKSTRAS ALGORITHMUS) in den Algorithmen der Abschnitte 9.4 und 9.5. Man zeige, dass jede dieser Berechnungen sogar für Graphen mit parallelen Kanten in $O(n^2)$ -Zeit durchgeführt werden kann, unter der Voraussetzung, dass die Inzidenzliste von G , nach den Kosten der Kanten sortiert, vorliegt. Man folgere daraus, dass ORLINS ALGORITHMUS eine $O(mn^2 \log m)$ -Laufzeit hat.
- * 17. Der PUSH-RELABEL-ALGORITHMUS (Abschnitt 8.5) kann für das MINIMUM-COST-FLOW-PROBLEM verallgemeinert werden. Für eine gegebene Instanz (G, u, b, c) mit ganzzahligen Kosten c sucht man einen b -Fluss f und ein zulässiges Potenzial π in (G_f, c) . Zunächst setzt man $\pi := 0$ und saturiert alle Kanten e mit negativen Kosten. Dann führt man ③ des PUSH-RELABEL-ALGORITHMUS mit den folgenden Änderungen durch: Eine Kante e ist erlaubt,

falls $e \in E(G_f)$ und $c_\pi(e) < 0$. Ein Knoten v ist aktiv, falls $b(v) + \text{ex}_f(v) > 0$. In $\text{RELABEL}(v)$ setzt man $\pi(v) := \max\{\pi(w) - c(e) - 1 : e = (v, w) \in E(G_f)\}$. In $\text{PUSH}(e)$ für $e \in \delta^+(v)$ setzt man $\gamma := \min\{b(v) + \text{ex}_f(v), u_f(e)\}$.

(a) Man beweise, dass die Anzahl der RELABEL -Operationen $O(n^2|c_{\max}|)$ ist, wobei $c_{\max} = \max_{e \in E(G)} c(e)$.

Hinweis: Es muss irgendein Knoten w mit $b(w) + \text{ex}_f(w) < 0$ von jedem aktiven Knoten v aus in G_f erreichbar sein. Man beachte, dass $b(w)$ niemals verändert worden ist. Ferner beachte man die Beweise der Lemmata 8.25 und 8.27.

- (b) Man zeige, dass die Gesamlaufzeit $O(n^2 m c_{\max})$ ist.
- (c) Man beweise, dass der Algorithmus eine optimale Lösung berechnet.
- (d) Man wende Skalierung an, um einen $O(n^2 m \log c_{\max})$ -Algorithmus für das MINIMUM-COST-FLOW-PROBLEM mit ganzzahligen Kosten c zu bekommen.

(Goldberg und Tarjan [1990])

18. Sei (G, u, b, c) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS, wobei u und b ganzzahlig sind. Man beweise die folgende Aussage entweder (a) mittels des MINIMUM-MEAN-CYCLE-CANCELLING-ALGORITHMUS oder (b) mittels des SUKZESSIVE-KÜRZESTE-WEGE-ALGORITHMUS, oder (c) mittels totaler Unimodularität. Gibt es einen b -Fluss in (G, u) , so gibt es auch einen ganzzahligen b -Fluss mit minimalen Kosten.
19. Sei (G, u, c, b) eine Instanz des MINIMUM-COST-FLOW-PROBLEMS. Sei $\bar{e} \in E(G)$ mit $c(\bar{e}) > (|V(G)| - 1) \max_{e \in E(G) \setminus \{\bar{e}\}} |c(e)|$. Man beweise: Gibt es einen b -Fluss f in (G, u) mit $f(\bar{e}) = 0$, so gilt $f(\bar{e}) = 0$ für jede optimale Lösung f .
20. Gegeben sei ein Netzwerk (G, u, s, t) mit ganzzahligen Transitzeiten $l : E(G) \rightarrow \mathbb{Z}_+$, einem Zeithorizont $T \in \mathbb{N}$, einer Zahl $V \in \mathbb{R}_+$ und Kosten $c : E(G) \rightarrow \mathbb{R}_+$. Man sucht einen zeitabhängigen $s-t$ -Fluss f mit $\text{value}(f) = V$ und minimalen Kosten $\sum_{e \in E(G)} c(e) \int_0^T f_e(\tau) d\tau$. Man zeige, wie man dies in polynomieller Zeit erreicht, falls T eine Konstante ist.
Hinweis: Man betrachte ein zeiterweitertes Netzwerk mit einer Kopie von G für jeden diskreten Zeit-Schritt.

Literatur

Allgemeine Literatur:

- Ahuja, R.K., Magnanti, T.L., und Orlin, J.B. [1993]: Network Flows. Prentice-Hall, Englewood Cliffs 1993
- Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., und Schrijver, A. [1998]: Combinatorial Optimization. Wiley, New York 1998, Kapitel 4
- Goldberg, A.V., Tardos, É., und Tarjan, R.E. [1990]: Network flow algorithms. In: Paths, Flows, and VLSI-Layout (B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, Hrsg.), Springer, Berlin 1990, pp. 101–164
- Gondran, M., und Minoux, M. [1984]: Graphs and Algorithms. Wiley, Chichester 1984, Chapter 5

- Jungnickel, D. [2013]: Graphs, Networks and Algorithms. 4. Aufl. Springer, Berlin 2013, Kapitel 10 und 11
- Lawler, E.L. [1976]: Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston, New York 1976, Kapitel 4
- Ruhe, G. [1991]: Algorithmic Aspects of Flows in Networks. Kluwer Academic Publishers, Dordrecht 1991
- Skutella, M. [2009]: An introduction to network flows over time. In: Research Trends in Combinatorial Optimization (W.J. Cook, L. Lovász, J. Vygen, Hrsg.), Springer, Berlin 2009, pp. 451–482

Zitierte Literatur:

- Arkin, E.M., und Silverberg, E.B. [1987]: Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics* 18 (1987), 1–8
- Armstrong, R.D., und Jin, Z. [1997]: A new strongly polynomial dual network simplex algorithm. *Mathematical Programming* 78 (1997), 131–148
- Busacker, R.G., und Gowen, P.J. [1961]: A procedure for determining a family of minimum-cost network flow patterns. ORO Technical Paper 15, Operational Research Office, Johns Hopkins University, Baltimore 1961
- Cunningham, W.H. [1976]: A network simplex method. *Mathematical Programming* 11 (1976), 105–116
- Dantzig, G.B. [1951]: Application of the simplex method to a transportation problem. In: *Activity Analysis and Production and Allocation* (T.C. Koopmans, Hrsg.), Wiley, New York 1951, pp. 359–373
- Edmonds, J., und Karp, R.M. [1972]: Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM* 19 (1972), 248–264
- Fleischer, L., und Skutella, M. [2007]: Quickest flows over time. *SIAM Journal on Computing* 36 (2007), 1600–1630
- Ford, L.R., und Fulkerson, D.R. [1958]: Constructing maximal dynamic flows from static flows. *Operations Research* 6 (1958), 419–433
- Ford, L.R., und Fulkerson, D.R. [1962]: Flows in Networks. Princeton University Press, Princeton 1962
- Gale, D. [1957]: A theorem on flows in networks. *Pacific Journal of Mathematics* 7 (1957), 1073–1082
- Goldberg, A.V., und Tarjan, R.E. [1989]: Finding minimum-cost circulations by cancelling negative cycles. *Journal of the ACM* 36 (1989), 873–886
- Goldberg, A.V., und Tarjan, R.E. [1990]: Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research* 15 (1990), 430–466
- Grötschel, M., und Lovász, L. [1995]: Combinatorial optimization. In: *Handbook of Combinatorics*; Vol. 2 (R.L. Graham, M. Grötschel, L. Lovász, Hrsg.), Elsevier, Amsterdam 1995
- Hassin, R. [1983]: The minimum cost flow problem: a unifying approach to dual algorithms and a new tree-search algorithm. *Mathematical Programming* 25 (1983), 228–239
- Hitchcock, F.L. [1941]: The distribution of a product from several sources to numerous localities. *Journal of Mathematical Physics* 20 (1941), 224–230
- Hoffman, A.J. [1960]: Some recent applications of the theory of linear inequalities to extremal combinatorial analysis. In: *Combinatorial Analysis* (R.E. Bellman, M. Hall, Hrsg.), AMS, Providence 1960, pp. 113–128

- Hoppe, B., und Tardos, É. [2000]: The quickest transshipment problem. *Mathematics of Operations Research* 25 (2000), 36–62
- Iri, M. [1960]: A new method for solving transportation-network problems. *Journal of the Operations Research Society of Japan* 3 (1960), 27–87
- Jewell, W.S. [1958]: Optimal flow through networks. *Interim Technical Report* 8, MIT 1958
- Karzanov, A.V., und McCormick, S.T. [1997]: Polynomial methods for separable convex optimization in unimodular linear spaces with applications. *SIAM Journal on Computing* 26 (1997), 1245–1275
- Klein, M. [1967]: A primal method for minimum cost flows, with applications to the assignment and transportation problems. *Management Science* 14 (1967), 205–220
- Klinz, B., und Woeginger, G.J. [2004]: Minimum cost dynamic flows: the series-parallel case. *Networks* 43 (2004), 153–162
- Lee, Y.T., und Sidford, A. [2014]: Path finding methods for linear programming. *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science* (2014), 424–433
- Matsui, T., und Scheifele, R. [2016]: A linear time algorithm for the unbalanced Hitchcock transportation problem. *Networks* 67 (2016), 170–182
- Orden, A. [1956]: The transshipment problem. *Management Science* 2 (1956), 276–285
- Ore, O. [1956]: Studies on directed graphs I. *Annals of Mathematics* 63 (1956), 383–406
- Orlin, J.B. [1993]: A faster strongly polynomial minimum cost flow algorithm. *Operations Research* 41 (1993), 338–350
- Orlin, J.B. [1997]: A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming* 78 (1997), 109–129
- Orlin, J.B., Plotkin, S.A., und Tardos, É. [1993]: Polynomial dual network simplex algorithms. *Mathematical Programming* 60 (1993), 255–276
- Plotkin, S.A., und Tardos, É. [1990]: Improved dual network simplex. *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms* (1990), 367–376
- Schulz, A.S., Weismantel, R., und Ziegler, G.M. [1995]: 0/1-Integer Programming: optimization and augmentation are equivalent. In: *Algorithms – ESA '95; LNCS* 979 (P. Spirakis, Hrsg.), Springer, Berlin 1995, pp. 473–483
- Schulz, A.S., und Weismantel, R. [2002]: The complexity of generic primal algorithms for solving general integer problems. *Mathematics of Operations Research* 27 (2002), 681–692
- Tardos, É. [1985]: A strongly polynomial minimum cost circulation algorithm. *Combinatorica* 5 (1985), 247–255
- Tolstoĭ, A.N. [1930]: Metody nakhozhdeniya naimen'shego summovogo kilometrazha pri planirovanií perevozok v prostanstve. In: *Planirovanie Perevozok, Sbornik pervyi, Transpechat' NKPS*, Moskow 1930, pp. 23–55. (See A. Schrijver, On the history of the transportation and maximum flow problems, *Mathematical Programming* 91 (2002), 437–445)
- Tomizawa, N. [1971]: On some techniques useful for solution of transportation network problems. *Networks* 1 (1971), 173–194
- Vygen, J. [2002]: On dual minimum cost flow algorithms. *Mathematical Methods of Operations Research* 56 (2002), 101–126
- Wagner, H.M. [1959]: On a class of capacitated transportation problems. *Management Science* 5 (1959), 304–318



10 Kardinalitätsmaximale Matchings

Die Matching-Theorie ist eines der klassischen und wichtigsten Gebiete der Kombinatorik und der kombinatorischen Optimierung. In diesem Kapitel sind sämtliche Graphen ungerichtet. Wir erinnern daran, dass ein Matching aus einer Menge von paarweise disjunkten Kanten besteht. Das grundlegende Problem ist:

KARDINALITÄTS-MATCHING-PROBLEM

Instanz: Ein ungerichteter Graph G .

Aufgabe: Bestimme ein kardinalitätsmaximales Matching in G .

Da die gewichtete Variante dieses Problems wesentlich schwieriger ist, werden wir sie erst später in Kapitel 11 betrachten. Aber bereits die obige Kardinalitätsvariante hat diverse Anwendungen: In dem JOB-ZUORDNUNGSPROBLEM können wir annehmen, dass jeder Job die gleiche Zeitspanne, etwa eine Stunde, benötigt und können fragen, ob alle Jobs in einer Stunde erledigt werden können. Anders ausgedrückt: Für einen gegebenen bipartiten Graphen G mit Bipartition $V(G) = A \cup B$ suchen wir Zahlen $x : E(G) \rightarrow \mathbb{R}_+$ mit $\sum_{e \in \delta(a)} x(e) = 1$ für jeden Job $a \in A$ und $\sum_{e \in \delta(b)} x(e) \leq 1$ für jeden Angestellten $b \in B$. Wir können dies in der Form eines linearen Ungleichungssystems $x \geq 0$, $Mx \leq \mathbf{1}$, $M'x \geq \mathbf{1}$ schreiben, wobei die Zeilen von M und M' Zeilen der Knoten-Kanten-Inzidenzmatrix von G sind. Nach Satz 5.26 sind diese Matrizen vollständig-unimodular. Mit Satz 5.21 folgt sodann: Gibt es eine Lösung x , so gibt es auch eine ganzzahlige Lösung. Beachte nun, dass die ganzzahligen Lösungen des obigen linearen Ungleichungssystems genau die Inzidenzvektoren der Matchings in G sind, die A überdecken.

Definition 10.1. Sei G ein Graph und M ein Matching in G . Wir sagen: Ein Knoten v wird von M **überdeckt**, wenn $v \in e$ für ein $e \in M$, anderenfalls ist v **exponiert** bezüglich M . Wir nennen M ein **perfektes Matching**, falls alle Knoten von M überdeckt werden.

In Abschnitt 10.1 betrachten wir Matchings in bipartiten Graphen. Algorithmmisch gesehen kann dieses Problem auf das MAXIMUM-FLOW-PROBLEM zurückgeführt werden. Das Max-Flow-Min-Cut-Theorem und auch der Begriff der augmentierenden Wege haben schöne Interpretationen im vorliegenden Kontext.

Der Begriff des Matching in allgemeinen, nicht bipartiten Graphen lässt sich nicht direkt auf Netzwerkflüsse zurückführen. In den Abschnitten 10.2 und 10.3

führen wir zwei notwendige und hinreichende Bedingungen dafür ein, dass ein allgemeiner Graph ein perfektes Matching besitzt. In Abschnitt 10.4 betrachten wir faktorkritische Graphen, die für jedes $v \in V(G)$ ein Matching haben, welches alle Knoten außer v überdeckt. Diese Graphen spielen eine wichtige Rolle in dem Algorithmus von Edmonds für das KARDINALITÄTS-MATCHING-PROBLEM, den wir in Abschnitt 10.5 beschreiben, und in der gewichteten Variante, die wir erst später in den Abschnitten 11.2 und 11.3 besprechen werden.

10.1 Bipartite Matchings

Da das KARDINALITÄTS-MATCHING-PROBLEM in einem bipartiten Graphen G einfacher ist, werden wir zunächst diesen Fall betrachten. In diesem Abschnitt setzen wir einen bipartiten Graphen G mit Bipartition $V(G) = A \dot{\cup} B$ voraus. Da wir annehmen können, dass G zusammenhängend ist, können wir diese Bipartition als eindeutig bestimmt betrachten (Aufgabe 26, Kapitel 2).

Für einen gegebenen Graphen G bezeichne $\nu(G)$ die maximale Kardinalität eines Matchings in G und $\tau(G)$ die minimale Kardinalität einer Knotenüberdeckung in G .

Satz 10.2. (König [1931]) *Ist G bipartit, so gilt $\nu(G) = \tau(G)$.*

Beweis: Betrachte den Graphen $G' = (V(G) \dot{\cup} \{s, t\}, E(G) \cup \{\{s, a\} : a \in A\} \cup \{\{b, t\} : b \in B\})$. Dann ist $\nu(G)$ die maximale Anzahl intern disjunkter s - t -Wege und $\tau(G)$ die minimale Anzahl von Knoten, deren Entfernen t von s aus unerreichbar macht. Der Satz folgt nun sofort aus dem Satz von Menger (Satz 8.10). \square

Es gilt offensichtlich $\nu(G) \leq \tau(G)$ für jeden Graph (bipartit oder nicht), aber Gleichheit gilt im Allgemeinen nicht (das Dreieck K_3 ist ein Gegenbeispiel).

Mehrere Aussagen sind äquivalent zum Satz von König. Der Satz von Hall ist sicherlich die bekannteste Version.

Satz 10.3. (Hall [1935]) *Sei G ein bipartiter Graph mit Bipartition $V(G) = A \dot{\cup} B$. Dann hat G ein A überdeckendes Matching genau dann, wenn*

$$|\Gamma(X)| \geq |X| \quad \text{für alle } X \subseteq A. \quad (10.1)$$

Beweis: Die Notwendigkeit der Bedingung ist klar. Um zu zeigen, dass sie auch hinreichend ist, nehmen wir an, dass G kein A überdeckendes Matching habe, d. h. $\nu(G) < |A|$. Mit Satz 10.2 folgt dann $\tau(G) < |A|$.

Sei $A' \subseteq A$ und $B' \subseteq B$, so dass $A' \cup B'$ sämtliche Kanten überdeckt und $|A' \cup B'| < |A|$ gilt. Offensichtlich ist $\Gamma(A \setminus A') \subseteq B'$. Also folgt $|\Gamma(A \setminus A')| \leq |B'| < |A| - |A'| = |A \setminus A'|$, und somit ist die Hall-Bedingung (10.1) verletzt. \square

Wir weisen darauf hin, dass es nicht sehr schwierig ist, den Satz von Hall direkt zu beweisen. Der folgende Beweis stammt von Halmos und Vaughan [1950]:

Zweiter Beweis des Satzes 10.3: Wir werden zeigen, dass jeder die Hall-Bedingung (10.1) erfüllende Graph ein A überdeckendes Matching besitzt. Dazu benutzen wir Induktion über $|A|$. Die Fälle $|A| = 0$ und $|A| = 1$ sind trivial.

Für $|A| \geq 2$ betrachten wir zwei Fälle: gilt $|\Gamma(X)| > |X|$ für jede nichtleere echte Teilmenge X von A , dann nehmen wir irgendeine Kante $\{a, b\}$ ($a \in A$, $b \in B$), entfernen die beiden Endknoten und wenden Induktion an. Der kleinere Graph erfüllt die Hall-Bedingung, da $|\Gamma(X)| - |X|$ für kein $X \subseteq A \setminus \{a\}$ um mehr als eins abnehmen kann.

Für den anderen Fall nehmen wir an, dass es eine nichtleere echte Teilmenge X von A mit $|\Gamma(X)| = |X|$ gibt. Nach der Induktionsvoraussetzung gibt es ein X überdeckendes Matching in $G[X \cup \Gamma(X)]$. Wir behaupten nun, dass wir dieses Matching zu einem A überdeckenden Matching in G erweitern können. Aufgrund der Induktionsvoraussetzung genügt es zu zeigen, dass $G[(A \setminus X) \cup (B \setminus \Gamma(X))]$ die Hall-Bedingung erfüllt. Dazu bemerken wir: Für jedes $Y \subseteq A \setminus X$ gilt (im anfänglichen Graphen G):

$$|\Gamma(Y) \setminus \Gamma(X)| = |\Gamma(X \cup Y)| - |\Gamma(X)| \geq |X \cup Y| - |X| = |Y|. \quad \square$$

Ein Spezialfall des Satzes von Hall ist der sogenannte „Heiratssatz“:

Satz 10.4. (Frobenius [1917]) Sei G ein bipartiter Graph mit Bipartition $V(G) = A \dot{\cup} B$. Dann hat G ein perfektes Matching genau dann, wenn $|A| = |B|$ und $|\Gamma(X)| \geq |X|$ für alle $X \subseteq A$. \square

Die Vielzahl der Anwendungen des Satzes von Hall ist aus den Aufgaben 4–7 ersichtlich.

Der Beweis des Satzes von König (Satz 10.2) zeigt den Weg zur algorithmischen Lösung des bipartiten Matching-Problems an:

Satz 10.5. Das KARDINALITÄTS-MATCHING-PROBLEM für bipartite Graphen G kann in $O(nm)$ -Zeit gelöst werden, wobei $n = |V(G)|$ und $m = |E(G)|$.

Beweis: Sei G ein bipartiter Graph mit Bipartition $V(G) = A \dot{\cup} B$. Füge einen Knoten s hinzu und verbinde ihn mit jedem Knoten in A . Füge einen weiteren Knoten t hinzu und verbinde ihn mit jedem Knoten in B . Orientiere die Kanten von s nach A , von A nach B und von B nach t . Es seien alle Kapazitäten gleich 1. Dann entspricht ein s - t -Fluss mit maximalem ganzzahligem Wert einem Matching maximaler Kardinalität (und umgekehrt).

Also wenden wir den FORD-FULKERSON-ALGORITHMUS an und bestimmen einen s - t -Fluss mit maximalem Wert (und damit auch ein kardinalitätsmaximales Matching) nach höchstens n Augmentierungen. Da jede Augmentierung $O(m)$ -Zeit benötigt, sind wir fertig. \square

Dieses Resultat stammt im Wesentlichen von Kuhn [1955]. In der Tat kann man hier wieder den Begriff des kürzesten augmentierenden Weges verwenden (siehe den EDMONDS-KARP-ALGORITHMUS). Auf diese Weise erhält man

den $O(\sqrt{n}(m+n))$ -Algorithmus von Hopcroft und Karp [1973] und Karzanov [1973]. Dieser Algorithmus wird in den Aufgaben 11 und 12 besprochen. Kleinere Verbesserungen im HOPCROFT-KARP-ALGORITHMUS ergeben Laufzeiten von $O(n\sqrt{\frac{mn}{\log n}})$ (Alt et al. [1991]) und $O(m\sqrt{n}\frac{\log(n^2/m)}{\log n})$ (Feder und Motwani [1995]). Letztere Schranke ist die bisher beste für dichte Graphen. Für dünne Graphen hat Mądry [2013] einen gänzlich anderen, schnelleren Algorithmus angegeben, der eine Laufzeit von $O(m^{10/7} \log^{O(1)} m)$ erreicht.

Wir werden nun den Begriff des augmentierenden Weges für unseren Kontext adaptieren.

Definition 10.6. Sei G ein Graph (bipartit oder nicht) und M ein Matching in G . Ein Weg P heißt ein **M -alternierender Weg**, falls $E(P) \setminus M$ ein Matching ist. Ein M -alternierender Weg heißt **M -augmentierend**, falls er positive Länge hat und seine Endknoten bezüglich M exponiert sind.

Es folgt unmittelbar, dass M -augmentierende Wege ungerade Länge haben.

Satz 10.7. (Petersen [1891], Berge [1957]) Sei G ein Graph (bipartit oder nicht) mit einem Matching M . Dieses ist genau dann kardinalitätsmaximal, wenn es keinen M -augmentierenden Weg gibt.

Beweis: Gibt es einen M -augmentierenden Weg P , so bildet die symmetrische Differenz $M \triangle E(P)$ ein Matching mit größerer Kardinalität als die von M . So mit ist M nicht kardinalitätsmaximal. Gibt es andererseits ein Matching M' mit $|M'| > |M|$, so ist die symmetrische Differenz $M \triangle M'$ gleich der knotendisjunkten Vereinigung der alternierenden Kreise und Wege, wobei mindestens ein Weg M -augmentierend sein muss. \square

Dieses Resultat wurde bereits von Petersen beobachtet und dann von Berge wiederentdeckt, und ist allgemein als der Satz von Berge bekannt. Im bipartiten Fall folgt er natürlich auch aus Satz 8.5.

10.2 Die Tutte-Matrix

Nun betrachten wir kardinalitätsmaximale Matchings aus algebraischer Sicht. Sei G ein einfacher ungerichteter Graph und G' der aus G durch eine beliebige Kan tenorientierung hervorgehende gerichtete Graph. Für jeden Vektor $x = (x_e)_{e \in E(G)}$ von Variablen definieren wir die **Tutte-Matrix**

$$T_G(x) = (t_{vw}^x)_{v,w \in V(G)}$$

durch

$$t_{vw}^x := \begin{cases} x_{\{v,w\}} & \text{für } (v,w) \in E(G') \\ -x_{\{v,w\}} & \text{für } (w,v) \in E(G') \\ 0 & \text{sonst} \end{cases}$$

Eine solche Matrix M , mit der Eigenschaft $M = -M^\top$, heißt **schiefsymmetrisch**. Die Matrix $T_G(x)$, jedoch nicht ihr Rang, hängt von der Orientierung G' von G ab. Beachte, dass $\det T_G(x)$ ein Polynom in den Variablen x_e ($e \in E(G)$) ist.

Satz 10.8. (Tutte [1947]) *G hat genau dann ein perfektes Matching, wenn $\det T_G(x)$ nicht identisch Null ist.*

Beweis: Sei $V(G) = \{v_1, \dots, v_n\}$ und S_n die Menge aller Permutationen der Zahlenmenge $\{1, \dots, n\}$. Definitionsgemäß gilt

$$\det T_G(x) = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^n t_{v_i, v_{\pi(i)}}^x.$$

Sei $S'_n := \left\{ \pi \in S_n : \prod_{i=1}^n t_{v_i, v_{\pi(i)}}^x \neq 0 \right\}$. Jede Permutation $\pi \in S_n$ entspricht einem gerichteten Graphen $H_\pi := (V(G), \{(v_i, v_{\pi(i)}) : i = 1, \dots, n\})$, wobei $|\delta_{H_\pi}^-(x)| = |\delta_{H_\pi}^+(x)| = 1$ für jeden Knoten x . Für jede Permutation $\pi \in S'_n$ ist H_π ein Teilgraph von G' .

Gibt es eine Permutation $\pi \in S'_n$, so dass H_π allein aus geraden Kreisen besteht, so erhalten wir ein perfektes Matching in G , indem wir in jedem Kreis nur jede zweite Kante nehmen (und die Orientierungen ignorieren).

Gibt es keine solche Permutation, so gibt es für jedes $\pi \in S'_n$ eine Permutation $r(\pi) \in S'_n$ mit der Eigenschaft, dass man $H_{r(\pi)}$ durch die Umorientierung des ersten ungeraden Kreises in H_π erhält, d.h. desjenigen ungeraden Kreises, der den Knoten mit minimalem Index enthält. Offensichtlich gilt $r(r(\pi)) = \pi$.

Beachte, dass $\operatorname{sgn}(\pi) = \operatorname{sgn}(r(\pi))$, d.h. die zwei Permutationen haben dasselbe Signum: Besteht der erste ungerade Kreis aus den Knoten $v_{i_1}, \dots, v_{i_{2k+1}}$ with $\pi(i_j) = i_{j+1}$ ($j = 1, \dots, 2k$) und $\pi(i_{2k+1}) = i_1$, so erhalten wir $r(\pi)$ mittels 2k Transpositionen: Für $j = 1, \dots, k$ vertausche $\pi(i_{2j-1})$ mit $\pi(i_{2j})$ und vertausche dann $\pi(i_{2j})$ mit $\pi(i_{2k+1})$.

Ferner gilt $\prod_{i=1}^n t_{v_i, v_{\pi(i)}}^x = -\prod_{i=1}^n t_{v_i, v_{r(\pi)(i)}}^x$. Also heben sich die beiden entsprechenden Terme in der Summe

$$\det T_G(x) = \sum_{\pi \in S'_n} \operatorname{sgn}(\pi) \prod_{i=1}^n t_{v_i, v_{\pi(i)}}^x$$

gegenseitig auf. Dies gilt für alle Paare $\pi, r(\pi) \in S'_n$, somit folgt, dass $\det T_G(x)$ identisch Null ist.

Hat also G kein perfektes Matching, so ist $\det T_G(x)$ identisch Null. Hat aber andererseits G ein perfektes Matching M , so betrachten wir die durch $\pi(i) := j$ und $\pi(j) := i$ für alle $\{v_i, v_j\} \in M$ definierte Permutation. Der entsprechende Term $\prod_{i=1}^n t_{v_i, v_{\pi(i)}}^x = \prod_{e \in M} (-x_e^2)$ hebt sich aber mit keinem anderen Term auf, also ist $\det T_G(x)$ nicht identisch Null. \square

Ursprünglich benutzte Tutte Satz 10.8, um sein Hauptergebnis über Matchings zu beweisen, nämlich Satz 10.13. Es liefert Satz 10.8 aber keine gute Charakterisierung dafür, dass ein Graph ein perfektes Matching besitzt. Das Problem liegt

darin, dass die Determinante leicht zu berechnen ist, wenn ihre Elemente Zahlen sind (Satz 4.10), nicht aber wenn sie Variablen sind. Der Satz deutet jedoch einen randomisierten Algorithmus für das KARDINALITÄTS-MATCHING-PROBLEM an:

Korollar 10.9. (Lovász [1979]) *Sei $x = (x_e)_{e \in E(G)}$ ein Zufallsvektor, dessen Komponenten unabhängig und in $[0, 1]$ gleichverteilt sind. Dann ist der Rang von $T_G(x)$ mit Wahrscheinlichkeit 1 genau gleich zweimal die Größe eines kardinalitätsmaximalen Matchings.*

Beweis: Angenommen, der Rang von $T_G(x)$ ist k ; seien etwa die ersten k Zeilen linear unabhängig. Dann folgt $T_G(x) = \begin{pmatrix} A & B \\ -B^\top & C \end{pmatrix}$, wobei A eine schiefsymmetrische $(k \times k)$ -Matrix ist und $\begin{pmatrix} A & B \end{pmatrix}$ Rang k hat. Somit gibt es eine Matrix D mit $D \begin{pmatrix} A & B \end{pmatrix} = \begin{pmatrix} -B^\top & C \end{pmatrix}$. Es folgt $AD^\top = -(DA)^\top = B$, also hat A Rang k . Somit ist die Hauptuntermatrix A von $T_G(x)$ nichtsingulär und nach Satz 10.8 hat der von den entsprechenden Knoten induzierte Teilgraph ein perfektes Matching. Insbesondere ist k gerade und G hat ein Matching der Kardinalität $\frac{k}{2}$.

Hat G andererseits ein Matching der Kardinalität k , so ist die Determinante derjenigen Hauptuntermatrix T' , deren Zeilen und Spalten den $2k$ von M überdeckten Knoten entsprechen, nach Satz 10.8 nicht identisch Null. Die Menge der Vektoren x mit $\det T'(x) = 0$ muss dann das Maß Null haben. Also ist der Rang von $T_G(x)$ mit Wahrscheinlichkeit 1 mindestens gleich $2k$. \square

Offensichtlich ist es nicht möglich, Zufallszahlen aus $[0, 1]$ mit einem digitalen Rechner zu erzeugen. Es kann jedoch gezeigt werden, dass es genügt, Zufallszahlen aus der endlichen Menge $\{1, 2, \dots, N\}$ zu wählen. Für genügend großes N wird die Fehlerwahrscheinlichkeit beliebig klein (siehe Lovász [1979]). Man kann Lovász' Algorithmus zur Bestimmung eines kardinalitätsmaximalen Matchings anwenden (nicht nur seiner Kardinalität). Siehe Rabin und Vazirani [1989], Mulmuley, Vazirani und Vazirani [1987] und Mucha und Sankowski [2004] für weitere randomisierte Algorithmen zur Bestimmung eines kardinalitätsmaximalen Matchings in einem Graphen. Ferner bemerken wir, dass Geelen [2000] gezeigt hat, wie man Lovász' Algorithmus derandomisieren kann. Obwohl seine Laufzeit schlechter als die für Edmonds' Matching-Algorithmus ist (siehe Abschnitt 10.5), ist er für gewisse Verallgemeinerungen des KARDINALITÄTS-MATCHING-PROBLEMS von Bedeutung (siehe z. B. Geelen und Iwata [2005]).

10.3 Der Satz von Tutte

Nun wenden wir uns dem KARDINALITÄTS-MATCHING-PROBLEM in allgemeinen Graphen zu. Eine notwendige Bedingung dafür, dass ein Graph ein perfektes Matching besitzt, ist, dass jede Zusammenhangskomponente gerade ist (d. h. eine gerade Anzahl von Knoten hat). Diese Bedingung ist jedoch nicht hinreichend: Ein Gegenbeispiel ist der in Abb. 10.1(a) dargestellte Graph $K_{1,3}$.

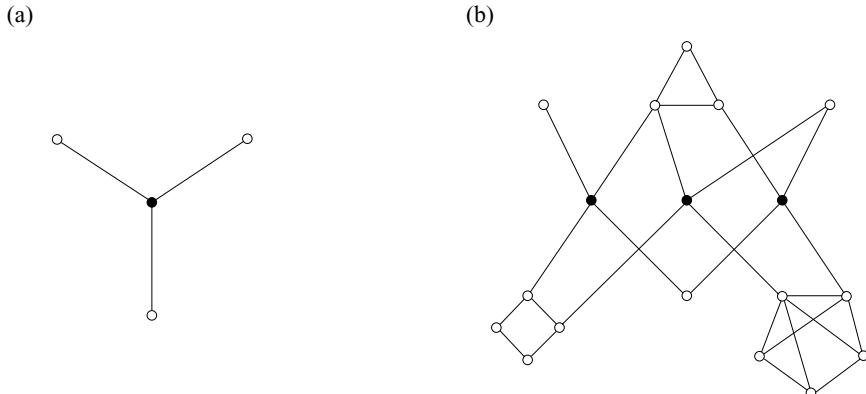


Abbildung 10.1.

Dass $K_{1,3}$ kein perfektes Matching besitzt, liegt daran, dass es einen Knoten (den schwarzen) gibt, dessen Entfernen drei ungerade Zusammenhangskomponenten ergibt. Der in Abb. 10.1(b) dargestellte Graph ist komplizierter. Besitzt dieser ein perfektes Matching? Entfernen wir die drei schwarzen Knoten, so bekommen wir fünf ungerade Zusammenhangskomponenten (und eine gerade Zusammenhangskomponente). Gäbe es ein perfektes Matching, so müsste mindestens ein Knoten aus jeder der fünf ungeraden Zusammenhangskomponenten mit einem der schwarzen Knoten verbunden sein. Dies ist jedoch unmöglich, da die Anzahl der ungeraden Zusammenhangskomponenten diejenige der schwarzen Knoten übersteigt.

Allgemeiner sei $q_G(X)$ für jede Knotenmenge $X \subseteq V(G)$ die Anzahl ungerader Zusammenhangskomponenten von $G - X$. Dann kann ein Graph mit $q_G(X) > |X|$ für ein bestimmtes $X \subseteq V(G)$ kein perfektes Matching besitzen: Sonst würde es für jede ungerade Zusammenhangskomponente in $G - X$ mindestens eine Matching-Kante geben, welche diese Zusammenhangskomponente mit X verbindet. Dies ist jedoch unmöglich, wenn die Anzahl der ungeraden Zusammenhangskomponenten diejenige der Elemente in X übersteigt. Der Satz von Tutte besagt, dass diese notwendige Bedingung auch hinreichend ist:

Definition 10.10. Ein Graph G erfüllt die **Tutte-Bedingung**, falls $q_G(X) \leq |X|$ für alle $X \subseteq V(G)$. Eine nichtleere Knotenmenge $X \subseteq V(G)$ ist eine **Barriere**, falls $q_G(X) = |X|$.

Um zu zeigen, dass die Tutte-Bedingung hinreichend ist, benötigen wir ein einfaches Resultat und eine wichtige Definition:

Proposition 10.11. Sei G ein Graph und $X \subseteq V(G)$. Dann gilt

$$q_G(X) - |X| \equiv |V(G)| \pmod{2}.$$

□

Definition 10.12. Ein Graph G heißt **faktorkritisch**, falls $G - v$ ein perfektes Matching für jedes $v \in V(G)$ besitzt. Ein Matching heißt **fast perfekt**, falls es alle Knoten bis auf einen überdeckt.

Nun können wir den Satz von Tutte beweisen:

Satz 10.13. (Tutte [1947]) Ein Graph G hat genau dann ein perfektes Matching, wenn er die Tutte-Bedingung erfüllt:

$$q_G(X) \leq |X| \quad \text{für alle } X \subseteq V(G).$$

Beweis: Die Notwendigkeit der Tutte-Bedingung haben wir bereits eingesehen. Dass sie auch hinreichend ist, beweisen wir mittels Induktion über $|V(G)|$. Der Fall $|V(G)| \leq 2$ ist trivial.

Sei G ein die Tutte-Bedingung erfüllender Graph. Es kann $|V(G)|$ nicht ungerade sein, da sonst die Tutte-Bedingung verletzt sein würde, weil $q_G(\emptyset) \geq 1$.

Also muss $|X| - q_G(X)$ nach Proposition 10.11 für jedes $X \subseteq V(G)$ gerade sein. Da $|V(G)|$ gerade ist und die Tutte-Bedingung erfüllt ist, ist jede einelementige Knotenmenge eine Barriere.

Nun wählen wir eine inklusionsmaximale Barriere X . Es hat $G - X$ genau $|X|$ ungerade Zusammenhangskomponenten. Ferner hat $G - X$ keine geraden Zusammenhangskomponenten, da sonst $X \cup \{v\}$, wobei v ein Knoten irgendeiner geraden Zusammenhangskomponente ist, eine Barriere wäre (beachte: $G - (X \cup \{v\})$ hat $|X| + 1$ ungerade Zusammenhangskomponenten), im Widerspruch zur Maximalität von X .

Wir behaupten nun, dass jede ungerade Zusammenhangskomponente von $G - X$ faktorkritisch ist. Um dies zu beweisen, sei C eine ungerade Zusammenhangskomponente von $G - X$ und $v \in V(C)$. Angenommen, $C - v$ hätte kein perfektes Matching. Dann folgt mit der Induktionsvoraussetzung, dass es ein $Y \subseteq V(C) \setminus \{v\}$ mit $q_{C-v}(Y) > |Y|$ gibt. Nach Proposition 10.11 ist $q_{C-v}(Y) - |Y|$ gerade, also folgt

$$q_{C-v}(Y) \geq |Y| + 2.$$

Da X , Y und $\{v\}$ paarweise disjunkt sind, haben wir

$$\begin{aligned} q_G(X \cup Y \cup \{v\}) &= q_G(X) - 1 + q_C(Y \cup \{v\}) \\ &= |X| - 1 + q_{C-v}(Y) \\ &\geq |X| - 1 + |Y| + 2 \\ &= |X \cup Y \cup \{v\}|. \end{aligned}$$

Somit ist $X \cup Y \cup \{v\}$ eine Barriere, im Widerspruch zur Maximalität von X .

Nun betrachten wir den bipartiten Graphen G' mit Bipartition $V(G') = X \dot{\cup} Z$, der dadurch entsteht, dass wir Kanten mit beiden Endknoten in X entfernen und die ungeraden Zusammenhangskomponenten von $G - X$ auf einzelne Knoten kontrahieren (die eine Menge Z bilden).

Wir müssen nur noch zeigen, dass G' ein perfektes Matching besitzt. Wäre das Gegenteil der Fall, so gäbe es nach dem Satz von Frobenius (Satz 10.4) ein $A \subseteq Z$ mit $|\Gamma_{G'}(A)| < |A|$. Damit folgt $q_G(\Gamma_{G'}(A)) \geq |A| > |\Gamma_{G'}(A)|$, ein Widerspruch. \square

Dieser Beweis stammt von Anderson [1971]. Die Tutte-Bedingung liefert eine gute Charakterisierung des perfekten Matching-Problems: Entweder ein Graph hat ein perfektes Matching, oder er hat eine sogenannte **Tutte-Menge** X , die den Beweis dafür liefert, dass er kein perfektes Matching besitzt. Eine wichtige Folgerung des Satzes von Tutte ist die sogenannte Berge-Tutte-Formel:

Satz 10.14. (Berge [1958])

$$2v(G) + \max_{X \subseteq V(G)} (q_G(X) - |X|) = |V(G)|.$$

Beweis: Für jedes $X \subseteq V(G)$ gilt, jedes Matching muss mindestens $q_G(X) - |X|$ Knoten unüberdeckt lassen. Damit folgt $2v(G) + q_G(X) - |X| \leq |V(G)|$.

Um die umgekehrte Ungleichung zu beweisen, sei

$$k := \max_{X \subseteq V(G)} (q_G(X) - |X|).$$

Wir konstruieren einen neuen Graphen H , indem wir k neue Knoten zu G hinzufügen, von denen ein jeder mit allen alten Knoten verbunden ist.

Wenn wir beweisen können, dass H ein perfektes Matching besitzt, dann folgt

$$2v(G) + k \geq 2v(H) - k = |V(H)| - k = |V(G)|,$$

womit der Satz bewiesen ist.

Angenommen, H hätte kein perfektes Matching. Mit dem Satz von Tutte folgt dann, es gibt eine Menge $Y \subseteq V(H)$ mit $q_H(Y) > |Y|$. Nach Proposition 10.11 hat k dieselbe Parität wie $|V(G)|$, also ist $|V(H)|$ gerade. Daraus folgt $Y \neq \emptyset$ und somit ist $q_H(Y) > 1$. Dann enthält Y aber sämtliche neuen Knoten, also haben wir

$$q_G(Y \cap V(G)) = q_H(Y) > |Y| = |Y \cap V(G)| + k,$$

im Widerspruch zur Definition von k . \square

Wir schließen diesen Abschnitt mit einem später gebrauchten Resultat.

Proposition 10.15. Sei G ein Graph und $X \subseteq V(G)$ mit $|V(G)| - 2v(G) = q_G(X) - |X|$. Dann enthält jedes kardinalitätsmaximale Matching von G in jeder geraden Zusammenhangskomponente von $G - X$ ein perfektes Matching und in jeder ungeraden Zusammenhangskomponente von $G - X$ ein fast perfektes Matching und paart sämtliche Knoten in X mit Knoten in paarweise verschiedenen ungeraden Komponenten von $G - X$. \square

Später (in Satz 10.32) werden wir erfahren, dass X so gewählt werden kann, dass jede ungerade Zusammenhangskomponente von $G - X$ faktorkritisch ist.

10.4 Ohrenzerlegungen faktorkritischer Graphen

Dieser Abschnitt enthält einige Resultate über faktorkritische Graphen, die wir später benötigen. In Aufgabe 22, Kapitel 2, haben wir bereits gesehen, dass Graphen mit einer Ohrenzerlegung genau die 2-fach kantenzusammenhängenden Graphen sind. Hier betrachten wir nur ungerade Ohrenzerlegungen.

Definition 10.16. Eine Ohrenzerlegung heißt **ungerade**, falls jedes Ohr ungerade Länge hat.

Satz 10.17. (Lovász [1972]) Ein Graph ist genau dann faktorkritisch, wenn er eine ungerade Ohrenzerlegung hat. Ferner kann der Anfangsknoten der Ohrenzerlegung beliebig gewählt werden.

Beweis: Sei G ein Graph mit einer festen ungeraden Ohrenzerlegung. Wir werden mittels Induktion über die Ohrenanzahl beweisen, dass G faktorkritisch ist. Sei P das letzte Ohr in der ungeraden Ohrenzerlegung, P gehe etwa von x nach y , und sei G' der Graph bevor P hinzugefügt wurde. Wir müssen zeigen, dass $G - v$ für jeden Knoten $v \in V(G)$ ein perfektes Matching enthält. Ist v kein innerer Knoten von P , so folgt dies mittels Induktion (nehme jede zweite Kante von P zum perfekten Matching in $G' - v$ hinzu). Ist andererseits v ein innerer Knoten von P , so ist entweder $P_{[v,x]}$ oder $P_{[v,y]}$ gerade, etwa $P_{[v,x]}$. Mit der Induktionsvoraussetzung gibt es ein perfektes Matching in $G' - x$. Indem wir jede zweite Kante sowohl von $P_{[y,v]}$ als auch von $P_{[v,x]}$ hinzunehmen, bekommen wir ein perfektes Matching in $G - v$.

Nun beweisen wir die umgekehrte Richtung. Wir wählen den Anfangsknoten z der Ohrenzerlegung beliebig. Sei M ein $V(G) \setminus \{z\}$ überdeckendes fast perfektes Matching in G . Angenommen, wir haben bereits eine ungerade Ohrenzerlegung eines Teilgraphen G' von G mit der Eigenschaft, dass $z \in V(G')$ ist und $M \cap E(G')$ ein fast perfektes Matching in G' bildet. Ist dann auch noch $G = G'$, so sind wir fertig.

Gilt Letzteres nicht, so gibt es, da G zusammenhängend ist, eine Kante $e = \{x, y\} \in E(G) \setminus E(G')$ mit $x \in V(G')$. Ist $y \in V(G')$, so ist e das nächste Ohr. Andernfalls sei N ein $V(G) \setminus \{y\}$ überdeckendes fast perfektes Matching in G . Dann enthält $M \triangle N$ offensichtlich die Kanten eines y - z -Weges P . Sei w der erste zu $V(G')$ gehörende Knoten von P (von y ausgehend). Die letzte Kante von $P' := P_{[y,w]}$ kann nicht zu M gehören (da keine Kante von M in $V(G')$ beginnt), und die erste Kante kann nicht zu N gehören. Da P' M - N -alternierend ist, ist $|E(P')|$ gerade, somit bildet P' zusammen mit e das nächste Ohr. \square

Wir haben in der Tat eine besondere ungerade Ohrenzerlegung konstruiert:

Definition 10.18. Gegeben sei ein faktorkritischer Graph G und ein fast perfektes Matching M . Eine **M -alternierende Ohrenzerlegung** von G ist eine ungerade Ohrenzerlegung, bei der jedes Ohr ein M -alternierender Weg oder ein Kreis C mit $|E(C) \cap M| + 1 = |E(C) \setminus M|$ ist.

Es ist klar, dass der Anfangsknoten einer M -alternierenden Ohrenzerlegung der bezüglich M exponierte Knoten ist. Aus dem Beweis von Satz 10.17 folgt sofort:

Korollar 10.19. Für jeden faktorkritischen Graphen G und jedes fast perfekte Matching M in G gibt es eine M -alternierende Ohrenzerlegung. \square

Von nun an betrachten wir nur M -alternierende Ohrenzerlegungen. Eine interessante Art, eine M -alternierende Ohrenzerlegung effizient zu speichern, stammt von Lovász und Plummer [1986]:

Definition 10.20. Sei G ein faktorkritischer Graph und M ein fast perfektes Matching in G . Sei ferner r, P_1, \dots, P_k eine M -alternierende Ohrenzerlegung von G und $\mu, \varphi : V(G) \rightarrow V(G)$ zwei Funktionen. Wir sagen, dass μ und φ zu der Ohrenzerlegung r, P_1, \dots, P_k gehören, falls

- $\mu(x) = y$ für $\{x, y\} \in M$,
- $\varphi(x) = y$ für $\{x, y\} \in E(P_i) \setminus M$ und $x \notin \{r\} \cup V(P_1) \cup \dots \cup V(P_{i-1})$,
- $\mu(r) = \varphi(r) = r$.

Ist M gegeben, so sagen wir auch, dass φ zu r, P_1, \dots, P_k gehört.

Ist M ein festes fast perfektes Matching und gehören μ und φ zu zwei M -alternierenden Ohrenzerlegungen, so sind diese bis auf die Reihenfolge der Ohren miteinander identisch. Ferner kann man eine explizite Liste der Ohren in linearer Zeit erstellen:

OHRENZERLEGUNGSALGORITHMUS

Input: Ein faktorkritischer Graph G , zu einer M -alternierenden Ohrenzerlegung gehörende Funktionen μ, φ .
Output: Eine M -alternierende Ohrenzerlegung r, P_1, \dots, P_k .

- ① Sei am Anfang $X := \{r\}$, wobei r der Knoten mit $\mu(r) = r$ ist.
 Sei $k := 0$, und der Stack sei leer.
- ② **If** $X = V(G)$ **then go to** ⑤.
If der Stack ist nicht leer
 then sei $v \in V(G) \setminus X$ ein Endknoten des obersten Elements des Stacks,
 else wähle $v \in V(G) \setminus X$ beliebig.
- ③ Setze $x := v$, $y := \mu(v)$ und $P := (\{x, y\}, \{\{x, y\}\})$.
While $\varphi(\varphi(x)) = x$ **do**:
 Setze $P := P + \{x, \varphi(x)\} + \{\varphi(x), \mu(\varphi(x))\}$ und $x := \mu(\varphi(x))$.
While $\varphi(\varphi(y)) = y$ **do**:
 Setze $P := P + \{y, \varphi(y)\} + \{\varphi(y), \mu(\varphi(y))\}$ und $y := \mu(\varphi(y))$.
 Setze $P := P + \{x, \varphi(x)\} + \{y, \varphi(y)\}$. P ist dasjenige Ohr, welches y als inneren Knoten enthält. Platziere P oben auf den Stack.
- ④ **While** beide Endknoten des obersten Elements P des Stacks sind in X **do**:
 Entferne P aus dem Stack, setze $k := k + 1$, $P_k := P$ und
 $X := X \cup V(P)$.
Go to ②.
- ⑤ **For** alle $\{y, z\} \in E(G) \setminus (E(P_1) \cup \dots \cup E(P_k))$ **do**:
 Setze $k := k + 1$ und $P_k := (\{y, z\}, \{\{y, z\}\})$.

Proposition 10.21. Sei G ein faktorkritischer Graph und μ, φ zu einer M -alternierenden Ohrenzerlegung gehörende Funktionen. Dann ist diese Ohrenzerlegung bis auf die Reihenfolge der Ohren eindeutig bestimmt. Der OHRENZERLEGUNGSAORITHMUS bestimmt eine explizite Liste der Ohren korrekt und läuft in linearer Zeit.

Beweis: Sei \mathcal{D} eine M -alternierende Ohrenzerlegung, zu der μ und φ gehören. Die Eindeutigkeit von \mathcal{D} und die Korrektheit des Algorithmus folgen aus der offensichtlichen Tatsache, dass P , wie in ③ berechnet, tatsächlich ein Ohr von \mathcal{D} ist. Die Laufzeit von ① – ④ ist offensichtlich $O(|V(G)|)$, und ⑤ benötigt $O(|E(G)|)$ -Zeit. \square

Die wichtigste Eigenschaft der zu einer alternierenden Ohrenzerlegung gehörenden Funktionen ist die folgende:

Lemma 10.22. Sei G ein faktorkritischer Graph und μ, φ zwei zu einer M -alternierenden Ohrenzerlegung gehörende Funktionen. Sei r der bezüglich M exponierte Knoten. Dann definiert der maximale durch eine Anfangsteilfolge von

$$x, \mu(x), \varphi(\mu(x)), \mu(\varphi(\mu(x))), \varphi(\mu(\varphi(\mu(x)))), \dots$$

gegebene Weg einen M -alternierenden x - r -Weg gerader Länge für alle $x \in V(G)$.

Beweis: Sei $x \in V(G) \setminus \{r\}$ und P_i das erste x enthaltende Ohr. Offensichtlich ist irgendeine Anfangsteilfolge von

$$x, \mu(x), \varphi(\mu(x)), \mu(\varphi(\mu(x))), \varphi(\mu(\varphi(\mu(x)))), \dots$$

ein von x nach y laufender Teilweg Q von P_i , wobei $y \in \{r\} \cup V(P_1) \cup \dots \cup V(P_{i-1})$. Da wir eine M -alternierende Ohrenzerlegung haben, ist die letzte Kante von Q nicht in M ; somit hat Q gerade Länge. Falls $y = r$, sind wir fertig, anderenfalls wenden wir Induktion über i an. \square

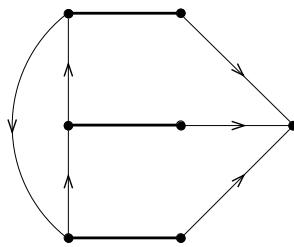


Abbildung 10.2.

Die Umkehrung von Lemma 10.22 gilt nicht: In dem in Abb. 10.2 dargestellten Gegenbeispiel (fett durchgezogene Kanten sind Matching-Kanten und eine von u

nach v gerichtete Kante bedeutet $\varphi(u) = v$) definieren μ und φ auch alternierende Wege zu dem bezüglich des Matchings exponierten Knoten. Es gehören μ und φ jedoch nicht zu irgendeiner alternierenden Ohrenzerlegung.

Für den GEWICHTETEN MATCHING-ALGORITHMUS (Abschnitt 11.3) benötigen wir eine schnelle Routine zur Update-Erstellung einer alternierenden Ohrenzerlegung wenn sich das Matching verändert. Obwohl der Beweis von Satz 10.17 algorithmischer Natur ist (vorausgesetzt wir können ein kardinalitätsmaximales Matching in einem Graphen bestimmen), ist dieser Weg viel zu ineffizient. Wir verwenden die alte Ohrenzerlegung:

Lemma 10.23. *Gegeben sei ein faktorkritischer Graph G , zwei fast perfekte Matchings M und M' und die zu einer M -alternierenden Ohrenzerlegung gehörenden Funktionen μ und φ . Dann kann man zwei zu einer M' -alternierenden Ohrenzerlegung gehörende Funktionen μ' , φ' in $O(|V(G)|)$ -Zeit bestimmen.*

Beweis: Sei v der bezüglich M exponierte Knoten und v' der bezüglich M' exponierte Knoten. Sei P der $v'-v$ -Weg in $M \triangle M'$, etwa $P = x_0, x_1, \dots, x_k$ mit $x_0 = v'$ und $x_k = v$.

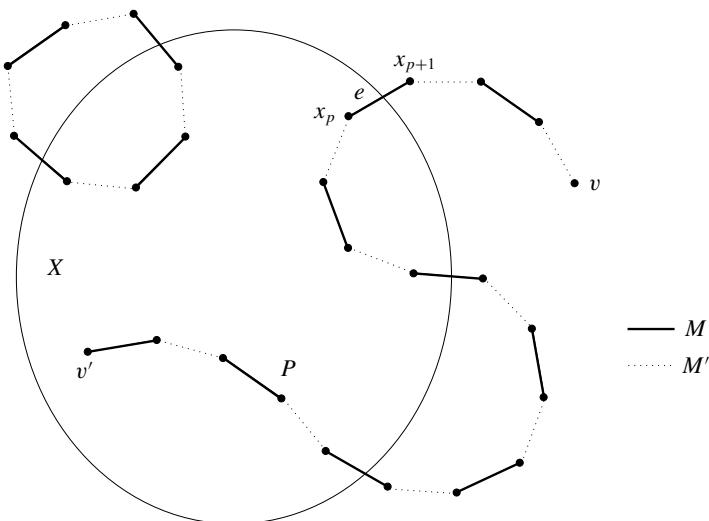


Abbildung 10.3.

Eine explizite Liste der Ohren der alten Ohrenzerlegung kann in linearer Zeit mittels des OHRENZERLEGUNGSALGORITHMUS aus μ und φ erstellt werden (Proposition 10.21). Da wir Ohren der Länge 1 nicht zu berücksichtigen brauchen, können wir sogar ⑤ auslassen: Damit ist die Gesamtanzahl der zu betrachtenden Kanten höchstens gleich $\frac{3}{2}(|V(G)| - 1)$ (siehe Aufgabe 21).

Angenommen, wir haben bereits eine M' -alternierende Ohrenzerlegung eines aufspannenden Teilgraphen von $G[X]$ für ein $X \subseteq V(G)$ mit $v' \in X$ (anfänglich

$X := \{v'\}$) konstruiert. Natürlich beginnt keine M' -Kante in X . Sei $p := \max\{i \in \{0, \dots, k\} : x_i \in X\}$ (wie in Abb. 10.3 dargestellt). Bei jedem Schritt speichern wir p und die Kantenmenge $\delta(X) \cap M$. Deren Update bei der Erweiterung von X ist offensichtlich in linearer Gesamtzeit möglich.

Nun zeigen wir, wie man die Ohrenzerlegung erweitert. In jedem Schritt werden wir ein oder mehrere Ohren hinzufügen. Die pro Schritt benötigte Zeit wird zu der Gesamtanzahl der Kanten in den neuen Ohren proportional sein.

Fall 1: $|\delta(X) \cap M| \geq 2$. Sei $f \in \delta(X) \cap M$ mit $x_p \notin f$. Offensichtlich liegt f auf einem M - M' -alternierenden Weg, der als nächstes Ohr hinzugefügt werden kann. Die zum Auffinden dieses Ohres benötigte Zeit ist zu dessen Länge proportional.

Fall 2: $|\delta(X) \cap M| = 1$. Dann ist $v \notin X$ und $e = \{x_p, x_{p+1}\}$ ist die einzige Kante in $\delta(X) \cap M$. Sei R' der durch μ und φ bestimmte x_{p+1} - v -Weg (siehe Lemma 10.22). Es ist e die erste Kante von R' . Sei q der kleinste Index $i \in \{p+2, p+4, \dots, k\}$ mit $x_i \in V(R')$ und $V(R'_{[x_{p+1}, x_i]}) \cap \{x_{i+1}, \dots, x_k\} = \emptyset$ (siehe Abb. 10.4). Sei $R := R'_{[x_p, x_q]}$. Somit hat R die Knoten $x_p, \varphi(x_p), \mu(\varphi(x_p)), \varphi(\mu(\varphi(x_p))), \dots, x_q$ und kann in einer zu seiner Länge proportionalen Zeit durchlaufen werden.

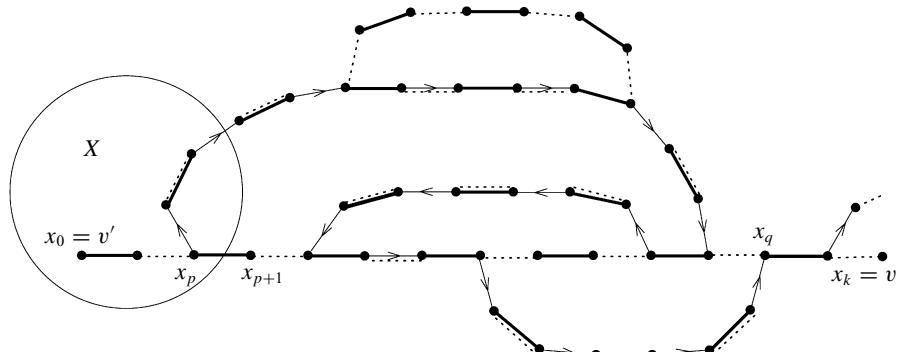


Abbildung 10.4.

Sei $S := E(R) \setminus E(G[X])$, $D := (M \triangle M') \setminus (E(G[X]) \cup E(P_{[x_q, v]}))$ und $Z := S \triangle D$. Es bestehen S und D aus M -alternierenden Wegen und Kreisen. Beachte, dass jeder Knoten außerhalb von X Grad 0 oder 2 bezüglich Z hat. Ferner haben wir: Für jeden Knoten außerhalb von X mit zwei aus Z inzidenten Kanten, ist eine dieser Kanten aus M' . (Hier ist die Wahl von q entscheidend.)

Damit können alle Zusammenhangskomponenten C von $(V(G), Z)$ mit $E(C) \cap \delta(X) \neq \emptyset$ als nächste Ohren hinzugefügt werden, und nach deren Hinzufügen ist $S \setminus Z = S \cap (M \triangle M')$ die knotendisjunkte Vereinigung von Wegen, die danach einzeln als nächstes Ohr hinzugefügt werden können. Da $e \in D \setminus S \subseteq Z$, folgt $Z \cap \delta(X) \neq \emptyset$, so haben wir mindestens ein Ohr hinzugefügt.

Es bleibt zu zeigen, dass die für die obige Konstruktion benötigte Zeit zur Gesamtanzahl der Kanten in den neuen Ohren proportional ist. Offensichtlich genügt es, S in $O(|S|)$ -Zeit zu bestimmen.

Dies wird durch die Teilwege von R innerhalb von X erschwert. Es ist uns jedoch im Grunde genommen egal, wie diese aussehen. Also würden wir diese Wege gerne womöglich umgehen. Um dies zu erreichen, ändern wir die φ -Variablen.

Dazu gehen wir folgendermaßen vor: Bei jeder Anwendung von Fall 2 sei $R_{[a,b]}$ ein maximaler Teilweg von R innerhalb von X mit $a \neq b$. Sei $y := \mu(b)$; es ist y der Vorgänger von b auf R . Nun setzen wir $\varphi(x) := y$ für alle Knoten x auf $R_{[x,y]}$, für die $R_{[x,y]}$ ungerade Länge hat. Ob x und y durch eine Kante verbunden sind, spielt keine Rolle. Ein Beispiel hierzu wird in Abb. 10.5 gegeben.

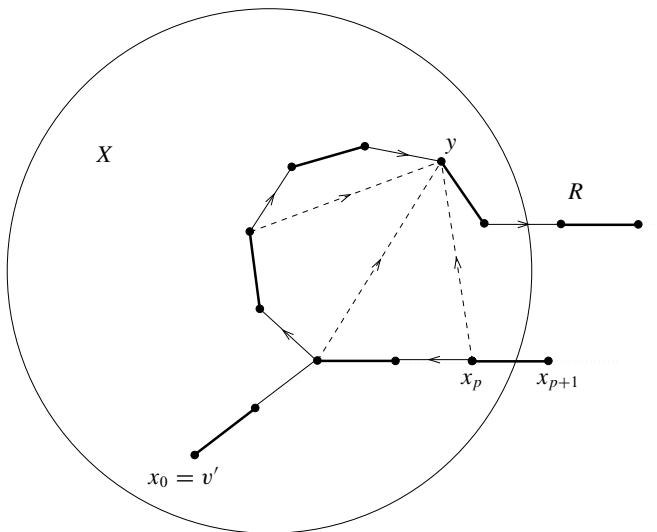


Abbildung 10.5.

Die zur Update-Erstellung der φ -Variablen benötigte Zeit ist zur Anzahl der betrachteten Kanten proportional. Beachte, dass diese φ -Änderungen die in Lemma 10.22 angegebene Eigenschaft nicht zerstören, und dass die φ -Variablen nur noch dazu benutzt werden, in Fall 2 M -alternierende Wege nach v zu bestimmen.

Damit ist gewährleistet, dass die zur Bestimmung der Teilwege von R innerhalb von X benötigte Zeit zur Anzahl der Teilwege plus der Anzahl der zum ersten Male innerhalb von X betrachteten Kanten proportional ist. Da die Anzahl der Teilwege innerhalb von X kleiner oder gleich der Anzahl der neuen Ohren in diesem Schritt ist, bekommen wir eine lineare Gesamlaufzeit.

Fall 3: $\delta(X) \cap M = \emptyset$. Dann ist $v \in X$. Wir betrachten die Ohren der (alten) M -alternierenden Ohrenzerlegung der Reihenfolge nach. Sei R das erste Ohr mit $V(R) \setminus X \neq \emptyset$.

Wie in Fall 2 setzen wir $S := E(R) \setminus E(G[X])$, $D := (M \triangle M') \setminus E(G[X])$ und $Z := S \triangle D$. Wiederum können alle Zusammenhangskomponenten C von $(V(G), Z)$ mit $E(C) \cap \delta(X) \neq \emptyset$ als nächste Ohren hinzugefügt werden, und nach deren Hinzufügen ist $S \setminus Z$ die knotendisjunkte Vereinigung von Wegen, die danach einzeln als nächstes Ohr hinzugefügt werden können. Die für Fall 3 benötigte Gesamtzeit ist offensichtlich linear. \square

10.5 Edmonds' Matching-Algorithmus

Wir erinnern an den Satz von Berge (Satz 10.7): Ein Matching in einem Graphen ist genau dann kardinalitätsmaximal, wenn es keinen augmentierenden Weg gibt. Da dies auch für nicht bipartite Graphen gilt, wird unser Matching-Algorithmus wieder auf augmentierenden Wegen basieren.

Es ist jedoch überhaupt nicht klar, wie man einen augmentierenden Weg bestimmen soll (oder entscheiden soll, dass es keinen solchen gibt). Im bipartiten Fall (Satz 10.5) genügte es, diejenigen Knoten zu markieren, die von einem bezüglich des Matchings exponierten Knoten aus mittels einer alternierenden Kantenfolge erreichbar sind. Da es im bipartiten Fall keine ungeraden Kreise gab, waren die mittels einer alternierenden Kantenfolge erreichbaren Knoten auch mittels eines alternierenden Weges erreichbar. Bei allgemeinen Graphen ist dies nicht der Fall.

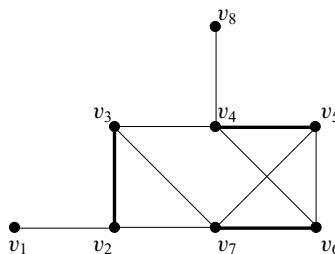


Abbildung 10.6.

Betrachte hierzu das in Abb. 10.6 dargestellte Beispiel (die fett durchgezogenen Kanten bilden ein Matching M). Beginnend in v_1 , haben wir eine alternierende Kantenfolge $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_5, v_4, v_8$, die aber keinen Weg bildet. Wir haben einen ungeraden Kreis durchlaufen, nämlich v_5, v_6, v_7 . Beachte, dass es in diesem Beispiel einen augmentierenden Weg gibt ($v_1, v_2, v_3, v_7, v_6, v_5, v_4, v_8$), dass es aber gar nicht klar ist, wie man diesen finden soll.

Es stellt sich die Frage, was zu tun ist, wenn man einen ungeraden Kreis antrifft. Erstaunlicherweise genügt es, ihn zu einem einzigen Knoten zu schrumpfen. Es stellt sich heraus, dass der kleinere Graph genau dann ein perfektes Matching besitzt, wenn dasselbe für den Ursprungsgraphen gilt. Dies verkörpert die allgemeine Idee von EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS. Diesen Algorithmus werden wir nach der folgenden Definition in Lemma 10.25 formulieren:

Definition 10.24. Sei G ein Graph und M ein Matching in G . Eine Blüte in G bezüglich M ist ein faktorkritischer Teilgraph C von G mit $|M \cap E(C)| = \frac{|V(C)|-1}{2}$. Der bezüglich $M \cap E(C)$ exponierte Knoten von C heißt die Basis von C .

Die im obigen Beispiel (Abb. 10.6) angetroffene Blüte wird durch $\{v_5, v_6, v_7\}$ induziert. Beachte, dass dieses Beispiel weitere Blüten enthält. Jeder einzelne Knoten ist auch eine Blüte laut unserer Definition. Nun können wir das Blütenlemma formulieren:

Lemma 10.25. Sei G ein Graph, M ein Matching in G und C eine Blüte in G (bezüglich M). Angenommen, es gibt einen M -alternierenden v - r -Weg Q mit gerader Länge von einem bezüglich M exponierten Knoten v aus bis zur Blütenbasis r von C , wobei $E(Q) \cap E(C) = \emptyset$.

Es gehe G' bzw. M' aus G bzw. M durch Schrumpfung von $V(C)$ zu einem einzigen Knoten hervor. Dann ist M ein kardinalitätsmaximales Matching in G genau dann, wenn M' ein kardinalitätsmaximales Matching in G' ist.

Beweis: Angenommen, M ist kein kardinalitätsmaximales Matching in G . Das Matching $N := M \triangle E(Q)$ hat dieselbe Kardinalität, ist also auch nicht kardinalitätsmaximal. Nach dem Satz von Berge (Satz 10.7) gibt es somit einen N -augmentierenden Weg P in G . Beachte, dass N nicht r überdeckt.

Wenigstens einer der Endknoten von P , etwa x , liegt nicht in C . Sind P und C disjunkt, so sei y der andere Endknoten von P . Sind anderenfalls P und C nicht disjunkt, so sei y der erste Knoten auf P – von x ausgehend – der in C liegt. Es gehe P' aus $P_{[x,y]}$ durch Schrumpfung von $V(C)$ in G hervor. Die Endknoten von P' sind bezüglich N' exponiert (N' ist das Matching in G' , welches N entspricht). Also ist P' ein N' -augmentierender Weg in G' . Damit ist N' kein kardinalitätsmaximales Matching in G' , also ist M' auch keines (M' hat dieselbe Kardinalität wie N').

Zum Beweis der Umkehrung nehmen wir an, dass M' kein kardinalitätsmaximales Matching in G' ist. Sei N' ein größeres Matching in G' . Es entspricht N' einem Matching N_0 in G , welches höchstens einen Knoten von C in G überdeckt. Da C faktorkritisch ist, kann man N_0 mit $k := \frac{|V(C)|-1}{2}$ Kanten zu einem Matching N in G erweitern, wobei

$$|N| = |N_0| + k = |N'| + k > |M'| + k = |M|,$$

womit bewiesen ist, dass M kein kardinalitätsmaximales Matching in G ist. \square

Man muss notwendigerweise fordern, dass die Blütenbasis mittels eines M -alternierenden Weges gerader Länge und disjunkt von der Blüte von einem Knoten aus erreichbar ist, der bezüglich M exponiert ist. Zum Beispiel kann die durch $\{v_4, v_6, v_7, v_2, v_3\}$ in Abb. 10.6 induzierte Blüte nicht geschrumpft werden, ohne den einzigen augmentierenden Weg zu zerstören.

Bei der Suche nach einem augmentierenden Weg werden wir einen alternierenden Wald aufbauen:

Definition 10.26. Gegeben sei ein Graph G und ein Matching M in G . Ein alternierender Wald bezüglich M in G ist ein Wald F in G mit den folgenden Eigenschaften:

- (a) $V(F)$ enthält alle bezüglich M exponierten Knoten. Jede Zusammenhangskomponente von F enthält genau einen bezüglich M exponierten Knoten, ihre Wurzel.
- (b) Ein Knoten $v \in V(F)$ heißt ein **äußerer (innerer)** Knoten, falls er eine gerade (ungerade) Distanz zur Wurzel der v enthaltenden Zusammenhangskomponente hat. (Insbesondere sind Wurzeln äußere Knoten). Alle inneren Knoten haben Grad 2 in F .
- (c) Für jedes $v \in V(F)$ ist der eindeutig bestimmte Weg von v aus zur Wurzel der v enthaltenden Zusammenhangskomponente M -alternierend.

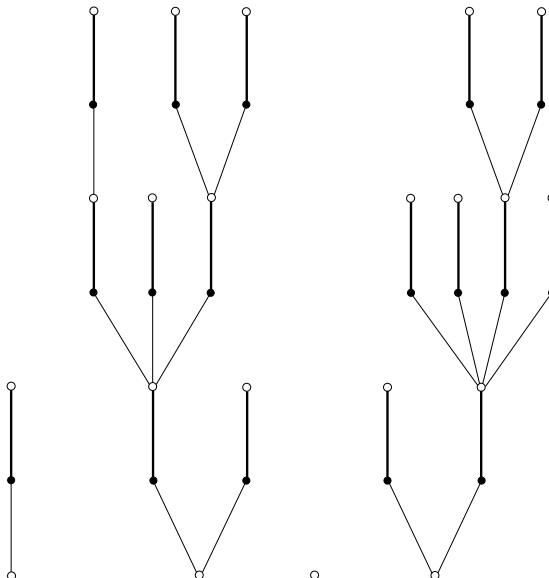


Abbildung 10.7.

Abbildung 10.7 zeigt einen alternierenden Wald. Die fett durchgezogenen Kanten liegen im Matching. Die schwarzen Knoten sind innere, die weißen äußere.

Proposition 10.27. In jedem alternierenden Wald ist die Anzahl der äußeren Knoten, die nicht Wurzeln sind, gleich der Anzahl der inneren Knoten.

Beweis: Jeder äußere Knoten, der nicht Wurzel ist, hat genau einen Nachbarn, der ein innerer Knoten ist und dessen Distanz zur Wurzel geringer ist. Dies liefert offensichtlich eine Bijektion zwischen den äußeren Knoten, die nicht Wurzeln sind, und den inneren Knoten. \square

In Worten ausgedrückt, funktioniert EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS folgendermaßen. Für ein gegebenes Matching M konstruieren wir schrittweise einen M -alternierenden Wald F . Wir beginnen mit der Menge S der bezüglich M exponierten Knoten und ohne jegliche Kanten.

Wir betrachten nun einen Nachbarn y eines äußeren Knotens x zu irgendeinem Zeitpunkt des Algorithmus. Sei $P(x)$ der eindeutig bestimmte Weg in F von x zu einer Wurzel. Es liegen drei interessante Fälle vor, entsprechend dreier Operationen („anwachsen“, „augmentieren“ und „schrumpfen“):

Fall 1: $y \notin V(F)$. Dann wird der Wald anwachsen, wenn wir $\{x, y\}$ und die y überdeckende Matching-Kante hinzufügen.

Fall 2: Es ist y ein äußerer Knoten in einer anderen Zusammenhangskomponente von F . Dann augmentieren wir M entlang $P(x) \cup \{x, y\} \cup P(y)$.

Fall 3: Es ist y ein äußerer Knoten in derselben Zusammenhangskomponente von F (mit Wurzel q). Sei r der erste Knoten von $P(x)$ (ausgehend von x), der auch auf $P(y)$ liegt. (Der Knoten r kann auch x oder y sein.) Ist r keine Wurzel, so hat r mindestens den Grad 3. Somit ist r ein äußerer Knoten. Also ist $C := P(x)_{[x,r]} \cup \{x, y\} \cup P(y)_{[y,r]}$ eine Blüte mit mindestens drei Knoten. Wir schrumpfen C .

Liegt keiner dieser drei Fälle vor, so sind alle Nachbarn äußerer Knoten innere Knoten. Wir behaupten, dass M kardinalitätsmaximal ist. Sei X die Menge der inneren Knoten, $s := |X|$ und t die Anzahl der äußeren Knoten. $G - X$ hat t ungerade Komponenten (jeder äußere Knoten ist isoliert in $G - X$), somit ist $q_G(X) - |X| = t - s$. Nach dem trivialen Teil der Berge-Tutte-Formel folgt sodann, dass jedes Matching mindestens $t - s$ Knoten nicht überdeckt. Andererseits ist aber die Anzahl der bezüglich M exponierten Knoten, d. h. die Anzahl der Wurzeln von F , nach Proposition 10.27 genau gleich $t - s$. Also ist M tatsächlich kardinalitätsmaximal.

Wir werden einige Zeit für die Details der Implementierung aufwenden, da dies keineswegs eine triviale Sache ist. Die schwierigste Frage dabei ist, wie man die Schrumpfung effizient erledigt, so dass man den ursprünglichen Graphen danach wieder herstellen kann. Natürlich können mehrere Schrumpfungen denselben Knoten betreffen. Unsere Darstellung basiert auf der Arbeit von Lovász und Plummer [1986].

Anstatt die Schrumpfung tatsächlich zu vollziehen, erlauben wir, dass unser Wald Blüten enthält.

Definition 10.28. Gegeben sei ein Graph G und ein Matching M in G . Ein Teilgraph F von G heißt ein **allgemeiner Blütenwald** (bezüglich M), falls es eine Partition $V(F) = V_1 \dot{\cup} V_2 \dot{\cup} \dots \dot{\cup} V_k$ gibt, so dass $F_i := F[V_i]$ ein inklusionsmaximaler faktorkritischer Teilgraph von F mit $|M \cap E(F_i)| = \frac{|V_i|-1}{2}$ ($i = 1, \dots, k$) ist und wir nach der Kontraktion aller V_1, \dots, V_k einen alternierenden Wald F' erhalten.

Es heißt F_i eine **äußere Blüte (innere Blüte)**, falls V_i ein äußerer (innerer) Knoten in F' ist. Jeder Knoten einer äußeren (inneren) Blüte heißt **äußerer (innerer) Knoten**. Ein allgemeiner Blütenwald, in dem jede innere Blüte ein einziger Knoten ist, heißt **ein spezieller Blütenwald**.

Abbildung 10.8 zeigt eine Zusammenhangskomponente eines speziellen Blütenwaldes mit fünf nicht-trivialen äußeren Blüten. Diese entspricht einer der Zusammenhangskomponenten des in Abb. 10.7 dargestellten alternierenden Waldes. Die Orientierungen der Kanten werden später erklärt. Alle nicht zu dem speziellen Blütenwald gehörenden Knoten von G heißen Blüten **außerhalb des Waldes**.

Beachte, dass sich das Blütenlemma (Lemma 10.25) nur auf äußere Blüten bezieht. Wir werden es jedoch in diesem Abschnitt nur mit speziellen Blütenwäldern zu tun haben. Allgemeine Blütenwälder werden nur im Zusammenhang mit dem GEWICHTETEN MATCHING-ALGORITHMUS in Kapitel 11 auftreten.

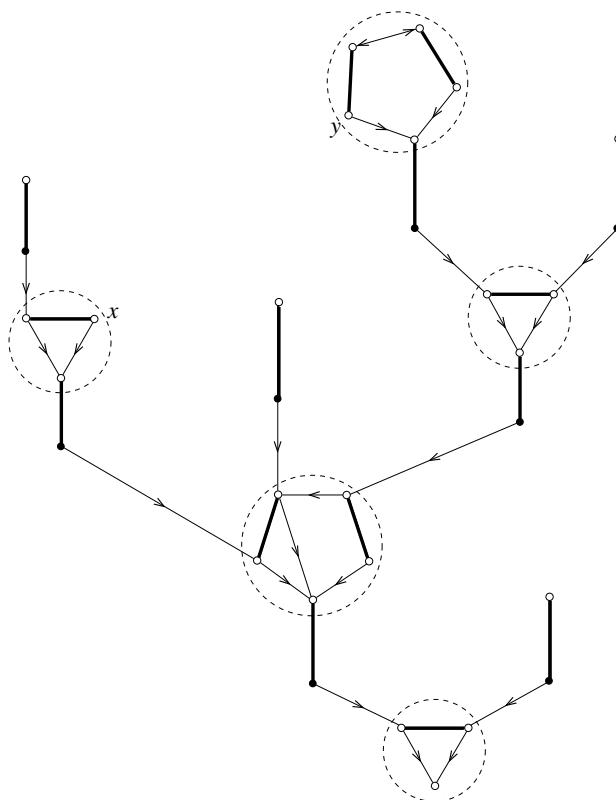


Abbildung 10.8.

Um einen speziellen Blütenwald F zu speichern, führen wir die folgenden Datenstrukturen ein. Für jeden Knoten $x \in V(G)$ haben wir die drei Variablen $\mu(x)$, $\varphi(x)$ und $\rho(x)$ mit den folgenden Eigenschaften:

$$\mu(x) = \begin{cases} x & \text{falls } x \text{ bez\"uglich } M \text{ exponiert ist} \\ y & \text{f\"ur } \{x, y\} \in M \end{cases} \quad (10.2)$$

$$\varphi(x) = \begin{cases} x & \text{falls } x \notin V(F) \text{ oder } x \text{ die Basis einer \"au\ss{}eren Bl\"ute ist} \\ y & \text{f\"ur } \{x, y\} \in E(F) \setminus M, \text{ falls } x \text{ ein innerer Knoten ist} \\ y & \text{f\"ur } \{x, y\} \in E(F) \setminus M \text{ mit } \mu \text{ und } \varphi \text{ zugeh\"orig zu einer } M\text{-alternierenden Ohrenzerlegung der } x \text{ enthaltenden Bl\"ute, falls } x \text{ ein \"au\ss{}erer Knoten ist} \end{cases} \quad (10.3)$$

$$\rho(x) = \begin{cases} x & \text{falls } x \text{ kein \"au\ss{}erer Knoten ist} \\ y & \text{falls } x \text{ ein \"au\ss{}erer Knoten und } y \text{ die Basis der } x \text{ enthaltenden \"au\ss{}eren Bl\"ute in } F \text{ ist.} \end{cases} \quad (10.4)$$

F\"ur jeden \"au\ss{}eren Knoten v sei $P(v)$ der maximale durch eine Anfangsteilfolge von

$$v, \mu(v), \varphi(\mu(v)), \mu(\varphi(\mu(v))), \varphi(\mu(\varphi(\mu(v)))), \dots$$

gegebene Weg. Es gelten die folgenden Eigenschaften:

Proposition 10.29. *Sei F ein spezieller Bl\"utenwald bez\"uglich eines Matchings M und seien $\mu, \varphi : V(G) \rightarrow V(G)$ zwei (10.2) und (10.3) erfüllende Funktionen. Dann haben wir:*

- (a) *F\"ur jeden \"au\ss{}eren Knoten v ist $P(v)$ ein alternierender v - q -Weg, wobei q die Wurzel des v enthaltenden Baumes von F ist.*
- (b) *Ein Knoten x ist ein*
 - *\"au\ss{}erer Knoten genau dann, wenn entweder $\mu(x) = x$ oder $\varphi(\mu(x)) \neq \mu(x)$;*
 - *innerer Knoten genau dann, wenn $\varphi(\mu(x)) = \mu(x)$ und $\varphi(x) \neq x$;*
 - *Knoten au\ss{}erhalb des Waldes genau dann, wenn $\mu(x) \neq x$ und $\varphi(x) = x$ und $\varphi(\mu(x)) = \mu(x)$.*

Beweis: (a): Nach (10.3) und Lemma 10.22 ist eine geeignete Anfangsteilfolge von

$$v, \mu(v), \varphi(\mu(v)), \mu(\varphi(\mu(v))), \varphi(\mu(\varphi(\mu(v)))), \dots$$

ein M -alternierender Weg mit gerader L\"ange, der bis zur Basis r der v enthaltenden Bl\"ute geht. Ist r nicht die Wurzel des v enthaltenden Baumes, so wird r von M \"uberdeckt. Somit setzt sich die obige Folge mit der Matching-Kante $\{r, \mu(r)\}$ und auch mit $\{\mu(r), \varphi(\mu(r))\}$ fort, da $\mu(r)$ ein innerer Knoten ist. Aber $\varphi(\mu(r))$ ist wieder ein \"au\ss{}erer Knoten, somit folgt der Beweis mittels Induktion.

(b): Ist x ein \"au\ss{}erer Knoten, so ist er entweder eine Wurzel (d. h. $\mu(x) = x$) oder $P(x)$ ist ein Weg der L\"ange mindestens 2, d. h. $\varphi(\mu(x)) \neq \mu(x)$.

Ist x ein innerer Knoten, so ist $\mu(x)$ die Basis einer \"au\ss{}eren Bl\"ute, somit gilt $\varphi(\mu(x)) = \mu(x)$ nach (10.3). Ferner ist $P(\mu(x))$ ein Weg der L\"ange mindestens 2, also folgt $\varphi(x) \neq x$.

Ist x ein Knoten außerhalb des Waldes, so wird x definitionsgemäß von M überdeckt, somit gilt $\mu(x) \neq x$ nach (10.2). Natürlich ist $\mu(x)$ auch ein Knoten außerhalb des Waldes, also folgt $\varphi(x) = x$ und $\varphi(\mu(x)) = \mu(x)$ nach (10.3).

Da jeder Knoten ein äußerer oder innerer Knoten oder ein Knoten außerhalb des Waldes ist und jeder Knoten genau eine der Bedingungen auf den drei rechten Seiten erfüllt, ist der Beweis zu Ende. \square

In Abb. 10.8 ist eine Kante von u nach v orientiert worden, falls $\varphi(u) = v$. Wir sind nun in der Lage, eine detaillierte Beschreibung des Algorithmus zu geben.

EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS

Input: Ein Graph G .

Output: Ein durch die Kanten $\{x, \mu(x)\}$ gegebenes kardinalitätsmaximales Matching in G .

- ① Setze $\mu(v) := v$, $\varphi(v) := v$, $\rho(v) := v$ und $\text{scanned}(v) := \text{false}$ für alle $v \in V(G)$.
- ② If alle äußereren Knoten sind gescannt worden
then stop,
else sei x ein äußerer Knoten mit $\text{scanned}(x) = \text{false}$.
- ③ Sei y ein Nachbar von x , so dass y ein Knoten außerhalb des Waldes ist oder (y ist ein äußerer Knoten und $\rho(y) \neq \rho(x)$).
If es gibt kein solches y **then** setze $\text{scanned}(x) := \text{true}$ und **go to** ②.
 ④ („anwachsen“)
If y ist ein Knoten außerhalb des Waldes **then** setze $\varphi(y) := x$ und **go to** ③.
- ⑤ („augmentieren“)
If $P(x)$ und $P(y)$ sind knotendisjunkt **then**:
Setze $\mu(\varphi(v)) := v$, $\mu(v) := \varphi(v)$ für alle $v \in V(P(x)) \cup V(P(y))$ mit ungerader Distanz zu x bzw. y auf $P(x)$ bzw. $P(y)$.
Setze $\mu(x) := y$.
Setze $\mu(y) := x$.
Setze $\varphi(v) := v$, $\rho(v) := v$, $\text{scanned}(v) := \text{false}$ für alle $v \in V(G)$.
Go to ②.
- ⑥ („schrumpfen“)
Sei r der erste Knoten aus $V(P(x)) \cap V(P(y))$ mit $\rho(r) = r$.
For $v \in V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})$ mit ungerader Distanz zu x bzw. y auf $P(x)_{[x,r]}$ bzw. $P(y)_{[y,r]}$ und $\rho(\varphi(v)) \neq r$ **do**:
Setze $\varphi(\varphi(v)) := v$.
If $\rho(x) \neq r$ **then** setze $\varphi(x) := y$.
If $\rho(y) \neq r$ **then** setze $\varphi(y) := x$.
For alle $v \in V(G)$ mit $\rho(v) \in V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})$ **do**:
Setze $\rho(v) := r$.
Go to ③.

Um eine Vorstellung der Auswirkung des Schrumpfens auf die φ -Werte zu erhalten, siehe Abb. 10.9, wo Schritt ⑥ des Algorithmus auf x und y aus Abb. 10.8 angewendet wurde.

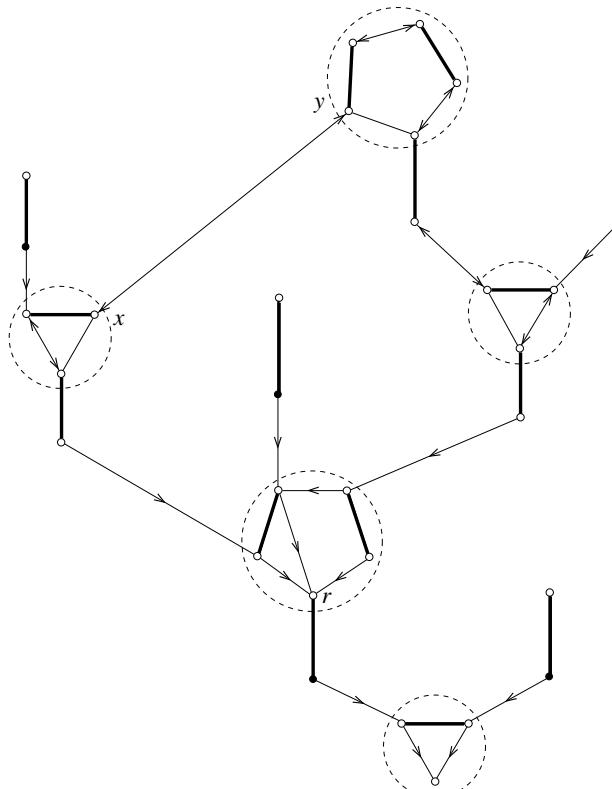


Abbildung 10.9.

Lemma 10.30. Die folgenden drei Aussagen gelten zu jedem Zeitpunkt von EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS:

- (a) Die Kanten $\{x, \mu(x)\}$ bilden ein Matching M ;
- (b) Die Kanten $\{x, \mu(x)\}$ und $\{x, \varphi(x)\}$ bilden einen speziellen Blütenwald F bezüglich M (plus einigen isolierten Matching-Kanten);
- (c) Die Eigenschaften (10.2), (10.3) und (10.4) sind bezüglich F erfüllt.

Beweis: (a): Die einzige Stelle, wo μ verändert wird, ist ⑤, und dort wird die Augmentierung offensichtlich korrekt ausgeführt.

(b): Da wir nach ① und ⑤ trivialerweise einen Blütenwald ohne Kanten haben und dieser in ④ korrekt um zwei Kanten anwächst, müssen wir nur ⑥ prüfen.

Entweder ist r eine Wurzel oder r hat mindestens den Grad 3, also ist r ein äußerer Knoten. Die neue Blüte besteht aus den Knoten $B := \{v \in V(G) : \rho(v) \in V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})\}$. Betrachte eine Kante $\{u, v\}$ des Blütenwaldes mit $u \in B$ und $v \notin B$. Da $F[B]$ ein fast perfektes Matching enthält, ist die Kante $\{u, v\}$ nur dann eine Matching-Kante, wenn sie $\{r, \mu(r)\}$ ist. Ferner war u ein äußerer Knoten vor der Ausführung von ⑥. Die Tatsache, dass $F[B]$ faktorkritisch ist, folgt aus der Existenz einer M -alternierenden Ohrendekomposition (siehe (c)) und mit Satz 10.17. Damit folgt, dass F weiterhin ein spezieller Blütenwald bleibt.

(c): Die einzige nicht-triviale Tatsache ist hier, dass μ und φ nach einer Schrumpfung zu einer alternierenden Ohrenzerlegung der neuen Blüte gehören. Seien also x und y zwei äußere Knoten in derselben Zusammenhangskomponente des speziellen Blütenwaldes, und sei r der erste Knoten aus $V(P(x)) \cap V(P(y))$ mit $\rho(r) = r$. Sei B wie oben die Knotenmenge der neuen Blüte.

Beachte, dass $\varphi(v)$ für kein $v \in B$ mit $\rho(v) = r$ verändert wird. Somit ist die Ohrenzerlegung der alten Blüte $B' := \{v \in V(G) : \rho(v) = r\}$ der Anfang der Ohrenzerlegung von B . Das nächste Ohr besteht aus $P(x)_{[x,x']}$, $P(y)_{[y,y']}$, und der Kante $\{x, y\}$, wobei x' bzw. y' der erste in B' liegende Knoten auf $P(x)$ bzw. $P(y)$ ist. Schließlich ist $Q \setminus (E(P(x)) \cup E(P(y)))$ für jedes Ohr Q einer alten äußeren Blüte $B'' \subseteq B$, ein Ohr der neuen Ohrenzerlegung von B . \square

Satz 10.31. (Edmonds [1965]) EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS bestimmt ein kardinalitätsmaximales Matching korrekt in $O(n^3)$ -Zeit, wobei $n = |V(G)|$.

Beweis: Nach Lemma 10.30 und Proposition 10.29 arbeitet der Algorithmus korrekt. Betrachte die Lage bei Terminierung des Algorithmus. Sei M das Matching und F der spezielle Blütenwald entsprechend Lemma 10.30(a) und (b). Offensichtlich ist jeder Nachbar eines äußeren Knotens x entweder ein innerer Knoten oder ein in derselben Blüte liegender Knoten y (d. h. $\rho(y) = \rho(x)$).

Um zu zeigen, dass M ein kardinalitätsmaximales Matching ist, sei X die Menge der inneren Knoten und B die Menge derjenigen Knoten, die Basis einer äußeren Blüte in F sind. Dann liegt jeder vom Matching ungepaarte Knoten in B , und die gepaarten Knoten von B sind mit Elementen aus X gepaart:

$$|B| = |X| + |V(G)| - 2|M|. \quad (10.5)$$

Andererseits sind die äußeren Blüten in F ungerade Zusammenhangskomponenten in $G - X$. Damit werden für jedes Matching mindestens $|B| - |X|$ Knoten nicht überdeckt. Nach (10.5) folgt: Genau $|B| - |X|$ Knoten werden von M nicht überdeckt und somit ist M kardinalitätsmaximal.

Nun betrachten wir die Laufzeit. Nach Proposition 10.29(b) kann der Status (innerer oder äußerer Knoten, oder Knoten außerhalb des Waldes) eines jeden Knotens in konstanter Zeit geprüft werden. Jeder der Schritte ④, ⑤ und ⑥ kann in $O(n)$ -Zeit ausgeführt werden. Zwischen zwei Augmentierungen werden ④ oder ⑥ höchstens $O(n)$ mal ausgeführt, da die Anzahl der Fixpunkte von φ jedes Mal abnimmt. Zwischen zwei Augmentierungen wird auch kein Knoten zweimal

gescannt. Somit ist die Zeitspanne zwischen zwei Augmentierungen $O(n^2)$, womit wir eine $O(n^3)$ Gesamtaufzeit erhalten. \square

Micali und Vazirani [1980] haben die Laufzeit auf $O(\sqrt{n}m)$ verbessert. Sie benutzten die in Aufgabe 11 angegebenen Resultate, aber die Existenz von Blüten macht die Suche einer maximalen Menge paarweise knotendisjunkter augmentierender Wege mit minimaler Länge schwieriger als im bipartiten Fall (der schon früher von Hopcroft und Karp [1973] und Karzanov [1973] gelöst worden war, siehe Aufgabe 12). Siehe auch Vazirani [1994]. Die momentan beste Zeitkomplexität für das KARDINALITÄTS-MATCHING-PROBLEM ist $O\left(m\sqrt{n}\frac{\log(n^2/m)}{\log n}\right)$, genau wie im bipartiten Fall. Diese ist von Goldberg und Karzanov [2004] und von Fremuth-Paeger und Jungnickel [2003] erreicht worden.

Mit dem Matching-Algorithmus können wir auch leicht den Gallai-Edmonds-Struktursatz beweisen. Dies wurde zuerst von Gallai getan, aber EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS liefert einen konstruktiven Beweis.

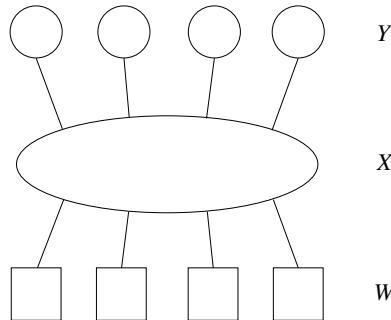


Abbildung 10.10.

Satz 10.32. (Gallai [1964]) Sei G ein Graph. Es bezeichne Y die Menge derjenigen Knoten, die bezüglich mindestens einem kardinalitätsmaximalen Matching exponiert sind. Ferner bezeichne X die Menge der Nachbarn von Y in $V(G) \setminus Y$ und W die Menge der restlichen Knoten. Dann gilt:

- (a) Jedes kardinalitätsmaximale Matching in G enthält ein perfektes Matching von $G[W]$ und fast perfekte Matchings der Zusammenhangskomponenten von $G[Y]$ und paart alle Knoten in X mit paarweise verschiedenen Zusammenhangskomponenten von $G[Y]$;
- (b) Die Zusammenhangskomponenten von $G[Y]$ sind faktorkritisch;
- (c) $2v(G) = |V(G)| - q_G(X) + |X|$.

Wir nennen das Tripel W, X, Y die **Gallai-Edmonds-Dekomposition** von G (siehe Abb. 10.10).

Beweis: Wir wenden EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS an und betrachten das Matching M und den speziellen Blütenwald F bei Termi-

nierung. Sei X' die Menge der inneren Knoten, Y' die Menge der äußeren Knoten und W' die Menge der Knoten außerhalb des Waldes. Zunächst beweisen wir, dass X', Y', W' die Aussagen (a)–(c) erfüllen, und stellen sodann fest, dass $X = X'$, $Y = Y'$ und $W = W'$.

Aus dem Beweis von Satz 10.31 folgt $2v(G) = |V(G)| - q_G(X') + |X'|$. Nun wenden wir Proposition 10.15 auf X' an. Da die ungeraden Zusammenhangskomponenten von $G - X'$ gerade die äußeren Blüten in F sind, gilt (a) für X', Y' und W' . Da die äußeren Blüten faktorkritisch sind, gilt (b) auch.

Da (a) für X', Y' und W' gilt, wissen wir, dass jedes kardinalitätsmaximale Matching sämtliche Knoten in $V(G) \setminus Y'$ überdeckt. Anders ausgedrückt, $Y \subseteq Y'$. Wir behaupten, dass $Y' \subseteq Y$ auch gilt. Sei v ein äußerer Knoten in F . Dann ist $M \triangle E(P(v))$ ein kardinalitätsmaximales Matching M' , und M' überdeckt v nicht. Somit ist $v \in Y$.

Also ist $Y = Y'$. Daraus folgt $X = X'$ und $W = W'$, womit der Satz bewiesen ist. \square

Aufgaben

1. Sei G ein Graph und M_1, M_2 zwei inklusionsmaximale Matchings in G . Man beweise, dass $|M_1| \leq 2|M_2|$.
2. Es bezeichne $\alpha(G)$ die Größe einer kardinalitätsmaximalen stabilen Menge in G und $\zeta(G)$ die minimale Kardinalität einer Kantenüberdeckung. Man beweise:
 - (a) $\alpha(G) + \tau(G) = |V(G)|$ für jeden Graphen G .
 - (b) $v(G) + \zeta(G) = |V(G)|$ für jeden Graphen G ohne isolierte Knoten.
 - (c) $\zeta(G) = \alpha(G)$ für jeden bipartiten Graphen G ohne isolierte Knoten.
(König [1933], Gallai [1959])
3. Man beweise, dass ein k -regulärer bipartiter Graph k paarweise disjunkte perfekte Matchings besitzt. Man folgere daraus, dass die Kantenmenge eines bipartiten Graphen mit maximalem Grad k in k Matchings partitioniert werden kann.
(König [1916]; siehe Rizzi [1998] oder Satz 16.16)
- * 4. Eine partiell geordnete Menge („poset“) ist eine Menge S mit einer partiellen Ordnung auf S , d. h. mit einer Relation $R \subseteq S \times S$, die reflexiv ($(x, x) \in R$ für alle $x \in S$), antisymmetrisch (aus $(x, y) \in R$ und $(y, x) \in R$ folgt $x = y$) und transitiv (aus $(x, y) \in R$ und $(y, z) \in R$ folgt $(x, z) \in R$) ist. Es heißen $x, y \in S$ vergleichbar, falls $(x, y) \in R$ oder $(y, x) \in R$. Eine Kette (eine Antikette) ist eine Menge paarweise vergleichbarer (nicht vergleichbarer) Elemente von S . Man verwende den Satz von König (Satz 10.2) um den folgenden Satz von Dilworth [1950] zu beweisen:
In einer endlichen partiell geordneten Menge S ist die maximale Größe einer Antikette gleich der minimalen Anzahl von Ketten, die eine Partition von S bilden.
Hinweis: Man nehme zwei Kopien v' und v'' eines jeden $v \in S$ und betrachte den Graphen mit Kante $\{v', v''\}$ für jede Kante $(v, w) \in R$.
(Fulkerson [1956])

5. (a) Sei $S = \{1, 2, \dots, n\}$ und $0 \leq k < \frac{n}{2}$. Sei A bzw. B die Familie aller k -elementigen bzw. $(k+1)$ -elementigen Teilmengen von S . Man bilde den bipartiten Graphen

$$G = (A \dot{\cup} B, \{\{a, b\} : a \in A, b \in B, a \subseteq b\})$$

und beweise, dass G ein A überdeckendes Matching besitzt.

- * (b) Man beweise Sperner's Lemma: Die maximale Anzahl der Teilmengen einer n -elementigen Menge mit der Eigenschaft, dass keine der Teilmengen eine andere enthält, ist $\binom{n}{\lfloor \frac{n}{2} \rfloor}$.

(Sperner [1928])

6. Sei (U, \mathcal{S}) ein Mengensystem. Eine injektive Funktion $\Phi : \mathcal{S} \rightarrow U$ mit $\Phi(S) \in S$ für alle $S \in \mathcal{S}$ heißt ein Repräsentantensystem für \mathcal{S} . Man beweise:

- (a) \mathcal{S} hat ein Repräsentantensystem genau dann, wenn die Vereinigungsmenge von jeder Wahl von k Mengen aus \mathcal{S} mindestens die Kardinalität k hat.
(Hall [1935])

- (b) Für $u \in U$ setze man $r(u) := |\{S \in \mathcal{S} : u \in S\}|$. Sei $n := |\mathcal{S}|$ und $N := \sum_{S \in \mathcal{S}} |S| = \sum_{u \in U} r(u)$. Angenommen, $|S| < \frac{N}{n-1}$ für $S \in \mathcal{S}$ und $r(u) < \frac{N}{n-1}$ für $u \in U$. Dann hat \mathcal{S} ein Repräsentantensystem.
(Mendelsohn und Dulmage [1958])

7. Sei G ein bipartiter Graph mit Bipartition $V(G) = A \dot{\cup} B$. Angenommen, dass $S \subseteq A$ und $T \subseteq B$ ist und dass es ein S überdeckendes Matching gibt und auch ein T überdeckendes Matching. Man beweise, dass es dann auch ein $S \cup T$ überdeckendes Matching gibt.
(Mendelsohn und Dulmage [1958])

8. Jeder Student aus einer gegebenen Menge von Studenten bewirbt sich um Teilnahme an genau drei Seminaren seiner Wahl aus einer gegebenen Menge von Seminaren. Zwei der Seminare werden von 40 Studenten gewählt, alle anderen von weniger als 40 Studenten.

- (a) Man beweise, dass jeder Student an einem der Seminare seiner Wahl teilnehmen kann und dass dabei in keinem der Seminare mehr als 13 Studenten teilnehmen.
(b) Man zeige, wie man eine solche Zuordnung in $O(n^2)$ -Zeit berechnen kann, wobei n die Anzahl der Seminare ist.

9. Wie lässt sich überprüfen, ob ein gegebener ungerichteter Graph G eine Orientierung hat, so dass kein Knoten einen größeren Eingangsgrad hat als eine gegebene Zahl k ? Beschreiben Sie einen polynomiellen Algorithmus sowie eine notwendige und hinreichende Bedingung ("für alle $X \subseteq V(G)$ gilt ...").

10. Man zeige, dass jeder einfache Graph mit n Knoten und minimalem Grad k ein Matching der Kardinalität $\min\{k, \lfloor \frac{n}{2} \rfloor\}$ hat.

Hinweis: Man verwende den Satz von Berge (Satz 10.7).

11. Sei G ein Graph und M ein nicht kardinalitätsmaximales Matching in G .

- (a) Man zeige, dass es $v(G) - |M|$ paarweise knotendisjunkte M -augmentierende Wege in G gibt.

Hinweis: Man beachte den Beweis des Satzes von Berge (Satz 10.7).

- (b) Man beweise, dass es in G einen M -augmentierenden Weg mit Länge höchstens $\frac{v(G)+|M|}{v(G)-|M|}$ gibt.

- (c) Sei P ein kürzester M -augmentierender Weg in G und P' ein $(M \triangle E(P))$ -augmentierender Weg. Dann gilt $|E(P')| \geq |E(P)| + |E(P \cap P')|$.

Man betrachte den folgenden generischen Algorithmus. Man beginne mit dem leeren Matching und augmentiere das Matching in jeder Iteration entlang einem kürzesten augmentierenden Weg. Sei P_1, P_2, \dots die gewählte Folge von augmentierenden Wegen. Mit (c) gilt $|E(P_k)| \leq |E(P_{k+1})|$ für alle k .

- (d) Man zeige: Ist $|E(P_i)| = |E(P_j)|$ für $i \neq j$, so sind P_i und P_j knotendisjunkt.

- (e) Man benutze (b) um zu beweisen, dass die Folge $|E(P_1)|, |E(P_2)|, \dots$ höchstens $2\sqrt{v(G)} + 2$ verschiedene Zahlen enthält.

(Hopcroft und Karp [1973]), Karzanov [1973])

- * 12. Sei G ein bipartiter Graph. Man betrachte den in Aufgabe 11 beschriebenen generischen Algorithmus.

- (a) Man beweise, dass man für ein gegebenes Matching M die Vereinigung aller kürzesten M -augmentierenden Wege in G in $O(n+m)$ -Zeit bestimmen kann.

Hinweis: Man verwende eine Art von Breadth-First-Search mit alternierenden Matching- und Nicht-Matching-Kanten.

- (b) Man betrachte eine Iterationsfolge des Algorithmus mit konstant bleibender Länge des augmentierenden Weges. Man zeige, dass die für die gesamte Folge benötigte Zeit höchstens $O(n+m)$ ist.

Hinweis: Man wende zunächst (a) an und bestimme dann die Wege schrittweise mittels DFS. Dazu markiere man die bereits besuchten Knoten.

- (c) Man verbinde (b) mit Aufgabe 11(e) um einen $O(\sqrt{n}(m+n))$ -Algorithmus für das KARDINALITÄTS-MATCHING-PROBLEM in bipartiten Graphen zu erhalten.

Bemerkung: Dies kann als Spezialfall von Aufgabe 21(b), Kapitel 8, betrachtet werden.

(Hopcroft und Karp [1973], Karzanov [1973])

13. Sei G ein bipartiter Graph mit Bipartition $V(G) = A \dot{\cup} B$, $A = \{a_1, \dots, a_k\}$, $B = \{b_1, \dots, b_k\}$. Für jeden Vektor $x = (x_e)_{e \in E(G)}$ sei $M_G(x) = (m_{ij}^x)_{1 \leq i, j \leq k}$ die Matrix mit

$$m_{ij}^x := \begin{cases} x_e & \text{für } e = \{a_i, b_j\} \in E(G) \\ 0 & \text{sonst} \end{cases}.$$

Die Determinante $\det M_G(x)$ ist ein Polynom in $x = (x_e)_{e \in E(G)}$. Man beweise, dass G genau dann ein perfektes Matching besitzt, wenn $\det M_G(x)$ nicht identisch Null ist.

14. Die **Permanente** einer quadratischen Matrix $M = (m_{ij})_{1 \leq i, j \leq n}$ ist die Zahl

$$\text{per}(M) := \sum_{\pi \in S_n} \prod_{i=1}^n m_{i,\pi(i)},$$

wobei S_n die Menge der Permutationen von $\{1, \dots, n\}$ ist. Man beweise, dass ein einfacher bipartiter Graph G genau per($M_G(\mathbb{I})$) perfekte Matchings besitzt, wobei $M_G(x)$ die in Aufgabe 13 definierte Matrix ist.

15. Eine **doppelt-stochastische Matrix** ist eine nichtnegative quadratische Matrix, deren Zeilen- und Spaltensummen alle gleich 1 sind. Ganzzahlige doppelt-stochastische Matrizen heißen **Permutationsmatrizen**.

Falikman [1981] und Egoryčev [1980] haben bewiesen, dass für eine doppelt-stochastische $n \times n$ -Matrix M

$$\text{per}(M) \geq \frac{n!}{n^n}$$

gilt, mit Gleichheit genau dann, wenn jedes Element von M gleich $\frac{1}{n}$ ist. (Dies war eine berühmte Vermutung von van der Waerden; siehe auch Schrijver [1998].)

Brègman [1973] hat bewiesen, dass für eine 0-1-Matrix M mit Reihensummen r_1, \dots, r_n ,

$$\text{per}(M) \leq (r_1!)^{\frac{1}{r_1}} \cdot \dots \cdot (r_n!)^{\frac{1}{r_n}}$$

gilt.

Man verwende diese Resultate und Aufgabe 14, um Folgendes zu beweisen. Sei G ein einfacher k -regulärer bipartiter Graph mit $2n$ Knoten und $\Phi(G)$ die Anzahl der perfekten Matchings in G . Dann gilt

$$n! \left(\frac{k}{n} \right)^n \leq \Phi(G) \leq (k!)^{\frac{n}{k}}.$$

16. Man beweise, dass jeder 3-reguläre Graph mit höchstens zwei Brücken ein perfektes Matching besitzt. Gibt es einen 3-regulären Graphen ohne ein perfektes Matching?

Hinweis: Man verwende den Satz von Tutte (Satz 10.13).

(Petersen [1891])

- * 17. Sei G ein Graph, $n := |V(G)|$ gerade, und für jede Menge $X \subseteq V(G)$ mit $|X| \leq \frac{3}{4}n$ gelte

$$\left| \bigcup_{x \in X} \Gamma(x) \right| \geq \frac{4}{3}|X|.$$

Man beweise, dass G ein perfektes Matching besitzt.

Hinweis: Sei S eine die Tutte-Bedingung verletzende Menge. Man beweise, dass die Anzahl der einelementigen Zusammenhangskomponenten von $G - S$ höchstens gleich $\max \left\{ 0, \frac{4}{3}|S| - \frac{1}{3}n \right\}$ ist. Dazu betrachte man die beiden Fälle $|S| \geq \frac{n}{4}$ und $|S| < \frac{n}{4}$ separat.
(Anderson [1971])

18. Man beweise, dass ein ungerichteter Graph G genau dann faktorkritisch ist, wenn G zusammenhängend ist und $v(G) = v(G - v)$ für alle $v \in V(G)$ gilt.

19. Man beweise, dass alle ungeraden Ohrenzerlegungen eines faktorkritischen Graphen G dieselbe Anzahl von Ohren haben.
- * 20. Sei $\varphi(G)$ für einen 2-fach kantenzusammenhängenden Graphen G die minimale Anzahl gerader Ohren in einer Ohrenzerlegung von G (siehe Aufgabe 22(a), Kapitel 2). Man zeige, dass für jede Kante $e \in E(G)$ entweder $\varphi(G/e) = \varphi(G) + 1$ oder $\varphi(G/e) = \varphi(G) - 1$ gilt.
Bemerkung: Die Funktion $\varphi(G)$ ist von Frank [1993], von Szigeti [1996] und von Szegedy und Szegedy [2006] studiert worden.
21. Man beweise, dass ein minimaler faktorkritischer Graph G (d. h. nach dem Entfernen irgendeiner Kante ist G nicht mehr faktorkritisch) höchstens $\frac{3}{2}(|V(G)| - 1)$ Kanten hat. Man zeige, dass diese Schranke kleinstmöglich ist.
22. Man zeige, wie EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS ein kardinalitätsmaximales Matching für den in Abb. 10.1(b) dargestellten Graphen bestimmt.
23. Kann man für einen ungerichteten Graphen eine Kantenüberdeckung minimaler Kardinalität in polynomieller Zeit bestimmen?
- * 24. In einem ungerichteten Graphen G heißt eine Kante unmatchbar, wenn sie in keinem perfekten Matching liegt. Wie kann man die Menge der unmatchbaren Kanten in $O(n^3)$ -Zeit bestimmen?
Hinweis: Zunächst bestimme man ein perfektes Matching in G . Danach bestimme man für jeden Knoten v die Menge der mit v inzidenten unmatchbaren Kanten.
25. Sei G ein Graph und M ein kardinalitätsmaximales Matching in G . Seien F_1 und F_2 zwei spezielle Blütenwälder bezüglich M , beide mit jeweils der größtmöglichen Anzahl von Kanten. Man zeige, dass die Menge der inneren Kanten in F_1 gleich derjenigen für F_2 ist.
26. Sei G ein k -fach zusammenhängender Graph mit $2v(G) < |V(G)| - 1$. Man beweise:
 (a) $v(G) \geq k$;
 (b) $\tau(G) \leq 2v(G) - k$.
Hinweis: Man verwende den Gallai-Edmonds-Satz (Satz 10.32).
 (Erdős und Gallai [1961])

Literatur

Allgemeine Literatur:

- Gerards, A.M.H. [1995]: Matching. In: Handbooks in Operations Research and Management Science; Volume 7: Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, Hrsg.), Elsevier, Amsterdam 1995, pp. 135–224
- Lawler, E.L. [1976]: Combinatorial Optimization; Networks and Matroids. Holt, Rinehart and Winston, New York 1976, Kapitel 5 und 6
- Lovász, L., und Plummer, M.D. [1986]: Matching Theory. Akadémiai Kiadó, Budapest 1986, und North-Holland, Amsterdam 1986

- Papadimitriou, C.H., und Steiglitz, K. [1982]: Combinatorial Optimization; Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, Kapitel 10
- Pulleyblank, W.R. [1995]: Matchings and extensions. In: Handbook of Combinatorics; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, Hrsg.), Elsevier, Amsterdam 1995
- Schrijver, A. [2003]: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin 2003, Kapitel 16 und 24
- Tarjan, R.E. [1983]: Data Structures and Network Algorithms. SIAM, Philadelphia 1983, Kapitel 9

Zitierte Literatur:

- Alt, H., Blum, N., Mehlhorn, K., und Paul, M. [1991]: Computing a maximum cardinality matching in a bipartite graph in time $O\left(n^{1,5}\sqrt{m/\log n}\right)$. Information Processing Letters 37 (1991), 237–240
- Anderson, I. [1971]: Perfect matchings of a graph. Journal of Combinatorial Theory B 10 (1971), 183–186
- Berge, C. [1957]: Two theorems in graph theory. Proceedings of the National Academy of Science of the U.S. 43 (1957), 842–844
- Berge, C. [1958]: Sur le couplage maximum d'un graphe. Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris) Sér. I Math. 247 (1958), 258–259
- Brègman, L.M. [1973]: Certain properties of nonnegative matrices and their permanents. Doklady Akademii Nauk SSSR 211 (1973), 27–30 [auf Russisch]. English translation: Soviet Mathematics Doklady 14 (1973), 945–949
- Dilworth, R.P. [1950]: A decomposition theorem for partially ordered sets. Annals of Mathematics 51 (1950), 161–166
- Edmonds, J. [1965]: Paths, trees, and flowers. Canadian Journal of Mathematics 17 (1965), 449–467
- Egoryčev, G.P. [1980]: Solution of the van der Waerden problem for permanents. Soviet Mathematics Doklady 23 (1982), 619–622
- Erdős, P., und Gallai, T. [1961]: On the minimal number of vertices representing the edges of a graph. Magyar Tudományos Akadémia; Matematikai Kutató Intézetének Közleményei 6 (1961), 181–203
- Falikman, D.I. [1981]: A proof of the van der Waerden conjecture on the permanent of a doubly stochastic matrix. Matematicheskie Zametki 29 (1981), 931–938 [auf Russisch]. Englische Übersetzung: Math. Notes of the Acad. Sci. USSR 29 (1981), 475–479
- Feder, T., und Motwani, R. [1995]: Clique partitions, graph compression and speeding-up algorithms. Journal of Computer and System Sciences 51 (1995), 261–272
- Frank, A. [1993]: Conservative weightings and ear-decompositions of graphs. Combinatorica 13 (1993), 65–81
- Fremuth-Paeger, C., und Jungnickel, D. [2003]: Balanced network flows VIII: a revised theory of phase-ordered algorithms and the $O(\sqrt{nm} \log(n^2/m)/\log n)$ bound for the nonbipartite cardinality matching problem. Networks 41 (2003), 137–142
- Frobenius, G. [1917]: Über zerlegbare Determinanten. Sitzungsbericht der Königlich Preussischen Akademie der Wissenschaften XVIII (1917), 274–277
- Fulkerson, D.R. [1956]: Note on Dilworth's decomposition theorem for partially ordered sets. Proceedings of the AMS 7 (1956), 701–702
- Gallai, T. [1959]: Über extreme Punkt- und Kantenmengen. Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae; Sectio Mathematica 2 (1959), 133–138

- Gallai, T. [1964]: Maximale Systeme unabhängiger Kanten. Magyar Tudományos Akadémia; Matematikai Kutató Intézetének Közleményei 9 (1964), 401–413
- Geelen, J.F. [2000]: An algebraic matching algorithm. Combinatorica 20 (2000), 61–70
- Geelen, J. und Iwata, S. [2005]: Matroid matching via mixed skew-symmetric matrices. Combinatorica 25 (2005), 187–215
- Goldberg, A.V., und Karzanov, A.V. [2004]: Maximum skew-symmetric flows and matchings. Mathematical Programming A 100 (2004), 537–568
- Hall, P. [1935]: On representatives of subsets. Journal of the London Mathematical Society 10 (1935), 26–30
- Halmos, P.R., und Vaughan, H.E. [1950]: The marriage problem. American Journal of Mathematics 72 (1950), 214–215
- Hopcroft, J.E., und Karp, R.M. [1973]: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing 2 (1973), 225–231
- Karzanov, A.V. [1973]: Tochnaya otsenka algoritma nakhodeniya maksimal'nogo potoka, primenennogo k zadache "o predstaviteleyakh". In: Voprosy Kibernetiki, Trudy Seminara po Kombinatornoj Matematike, Sovetskoe Radio, Moskau, 1973, pp. 66–70 [auf Russisch]
- König, D. [1916]: Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. Mathematische Annalen 77 (1916), 453–465
- König, D. [1931]: Graphs and matrices. Matematikaiés Fizikai Lapok 38 (1931), 116–119 [auf Ungarisch]
- König, D. [1933]: Über trennende Knotenpunkte in Graphen (nebst Anwendungen auf Determinanten und Matrizen). Acta Litteratum ac Scientiarum Regiae Universitatis Hungaricae Francisco-Josephinae (Szeged). Sectio Scientiarum Mathematicarum 6 (1933), 155–179
- Kuhn, H.W. [1955]: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2 (1955), 83–97
- Lovász, L. [1972]: A note on factor-critical graphs. Studia Scientiarum Mathematicarum Hungarica 7 (1972), 279–280
- Lovász, L. [1979]: On determinants, matchings and random algorithms. In: Fundamentals of Computation Theory (L. Budach, Hrsg.), Akademie-Verlag, Berlin 1979, pp. 565–574
- Mądry, A. [2013]: Navigating central path with electrical flows: from flows to matchings, and back. Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (2013), 253–262
- Mendelsohn, N.S., und Dulmage, A.L. [1958]: Some generalizations of the problem of distinct representatives. Canadian Journal of Mathematics 10 (1958), 230–241
- Micali, S., und Vazirani, V.V. [1980]: An $O(V^{1/2}E)$ algorithm for finding maximum matching in general graphs. Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (1980), 17–27
- Mucha, M., und Sankowski, P. [2004]: Maximum matchings via Gaussian elimination. Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (2004), 248–255
- Mulmuley, K., Vazirani, U.V., und Vazirani, V.V. [1987]: Matching is as easy as matrix inversion. Combinatorica 7 (1987), 105–113
- Petersen, J. [1891]: Die Theorie der regulären Graphs. Acta Mathematica 15 (1891), 193–220
- Rabin, M.O., und Vazirani, V.V. [1989]: Maximum matchings in general graphs through randomization. Journal of Algorithms 10 (1989), 557–567
- Rizzi, R. [1998]: König's edge coloring theorem without augmenting paths. Journal of Graph Theory 29 (1998), 87

- Schrijver, A. [1998]: Counting 1-factors in regular bipartite graphs. *Journal of Combinatorial Theory B* 72 (1998), 122–135
- Sperner, E. [1928]: Ein Satz über Untermengen einer endlichen Menge. *Mathematische Zeitschrift* 27 (1928), 544–548
- Szegedy, B., und Szegedy, C. [2006]: Symplectic spaces and ear-decomposition of matroids. *Combinatorica* 26 (2006), 353–377
- Szigeti, Z. [1996]: On a matroid defined by ear-decompositions. *Combinatorica* 16 (1996), 233–241
- Tutte, W.T. [1947]: The factorization of linear graphs. *Journal of the London Mathematical Society* 22 (1947), 107–111
- Vazirani, V.V. [1994]: A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph maximum matching algorithm. *Combinatorica* 14 (1994), 71–109



11 Gewichtete Matchings

Das nicht-bipartite gewichtete Matching-Problem scheint eines der „schwersten“ in polynomieller Zeit lösbarer kombinatorischen Optimierungsprobleme zu sein. Wir werden EDMONDS’ KARDINALITÄTS-MATCHING-ALGORITHMUS für den gewichteten Fall erweitern und eine $O(n^3)$ -Implementierung angeben. Dieser Algorithmus hat diverse Anwendungen, von denen einige in den Aufgaben und in Abschnitt 12.2 vorgestellt werden. Es gibt zwei grundlegende Formulierungen des gewichteten Matching-Problems:

MAXIMUM-WEIGHT-MATCHING-PROBLEM

Instanz: Ein ungerichteter Graph G und Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Bestimme ein Matching maximalen Gewichtes in G .

MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM

Instanz: Ein ungerichteter Graph G und Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Aufgabe: Bestimme ein perfektes Matching minimalen Gewichtes in G oder entscheide, dass G kein perfektes Matching besitzt.

Proposition 11.1. *Das MAXIMUM-WEIGHT-MATCHING-PROBLEM und das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM sind äquivalent.*

Beweis: Ist (G, c) eine Instanz des MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEMS, so setzen wir $c'(e) := K - c(e)$ für alle $e \in E(G)$, wobei $K := 1 + \sum_{e \in E(G)} |c(e)|$. Dann ist jedes Matching maximalen Gewichtes in (G, c') ein kardinalitätsmaximales Matching und liefert somit eine Lösung des MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEMS (G, c) .

Sei umgekehrt (G, c) eine Instanz des MAXIMUM-WEIGHT-MATCHING-PROBLEMS. Sei H der Graph mit $V(H) := \{(v, i) : v \in V(G), i \in \{1, 2\}\}$ und $E(H) := \{\{(v, i), (w, i)\} : \{v, w\} \in E(G), i \in \{1, 2\}\} \cup \{\{(v, 1), (v, 2)\} : v \in V(G)\}$. So mit besteht H aus zwei Kopien von G und für jeden Knoten von G gibt es eine Kante in H , die die beiden Kopien dieses Knotens verbindet. Es hat H ein perfektes Matching. Sei $c'(\{(v, 1), (w, 1)\}) := -c(e)$ für alle $e = \{v, w\} \in E(G)$ und $c'(e) := 0$ für alle anderen Kanten $e \in E(H)$. Dann liefert ein perfektes Matching minimalen Gewichtes M in (H, c') ein Matching maximalen Gewichtes in (G, c) , indem man einfach $\{(v, w) \in E(G) : \{(v, 1), (w, 1)\} \in M\}$ nimmt. \square

Also werden wir jetzt nur noch das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM betrachten. Wie in dem vorangegangenen Kapitel, werden wir im Abschnitt 11.1 zunächst bipartite Graphen betrachten. Nach einem Abriss des GEWICHTETEN MATCHING-ALGORITHMUS in Abschnitt 11.2 werden wir im Abschnitt 11.3 einige Mühe auf Details der Implementierung aufwenden, um eine $O(n^3)$ -Laufzeit zu erreichen. Man ist öfters daran interessiert, viele nur in einigen Kanten differierende Matching-Probleme zu lösen; in solchen Fällen ist es nicht notwendig, die Probleme jeweils ganz von vorne anzufangen, wie wir in Abschnitt 11.4 zeigen werden. In Abschnitt 11.5 werden wir schließlich das Matching-Polytop besprechen. Dies ist die konvexe Hülle der Inzidenzvektoren von Matchings. Wir werden bereits bei der Gestaltung des GEWICHTETEN MATCHING-ALGORITHMUS eine Beschreibung des verwandten Perfekten-Matching-Polytops verwenden; andererseits folgt aus diesem Algorithmus direkt, dass diese Beschreibung vollständig ist.

11.1 Das Zuordnungsproblem

Das ZUORDNUNGSPROBLEM ist nichts als eine andere Bezeichnung für das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM in bipartiten Graphen. Es ist eines der klassischen kombinatorischen Optimierungsprobleme und lässt sich bis auf die Arbeiten von Monge [1784] zurückverfolgen.

Wie in dem Beweis von Satz 10.5 können wir das Zuordnungsproblem auf ein Netzwerkproblem zurückführen:

Satz 11.2. *Das ZUORDNUNGSPROBLEM kann in $O(nm + n^2 \log n)$ -Zeit gelöst werden.*

Beweis: Sei G ein bipartiter Graph mit Bipartition $V(G) = A \dot{\cup} B$. Wir nehmen an, dass $|A| = |B| = n$. Nun fügen wir einen Knoten s hinzu und verbinden ihn mit allen Knoten aus A , dann fügen wir einen weiteren Knoten t hinzu und verbinden ihn mit allen Knoten aus B . Die Kanten orientieren wir von s nach A , von A nach B und von B nach t . Es seien alle Kapazitäten gleich 1, ferner seien die Kosten der neuen Kanten alle 0.

Dann entspricht jeder ganzzahlige s - t -Fluss mit Wert n einem perfekten Matching mit denselben Kosten und umgekehrt. Also müssen wir ein MINIMUM-COST-FLOW-PROBLEM lösen. Dies erreichen wir durch Anwendung des SUKZESSIVE-KÜRZESTE-WEGE-ALGORITHMUS (siehe Abschnitt 9.4). Die Gesamtnachfrage ist n . Mit Satz 9.14 bekommen wir also eine $O(nm + n^2 \log n)$ -Laufzeit. \square

Dies ist der schnellste bekannte Algorithmus. Er ist im Wesentlichen äquivalent mit dem ältesten polynomiellen Algorithmus für das ZUORDNUNGSPROBLEM, der „Ungarischen Methode“ von Kuhn [1955] und Munkres [1957] (siehe Aufgabe 9).

Es ist lohnenswert, die LP-Formulierung des ZUORDNUNGSPROBLEMS zu betrachten. Es stellt sich heraus, dass man die Ganzzahligkeitsbedingungen in der Formulierung

$$\min \left\{ \sum_{e \in E(G)} c(e)x_e : x_e \in \{0, 1\} \ (e \in E(G)), \sum_{e \in \delta(v)} x_e = 1 \ (v \in V(G)) \right\}$$

weglassen kann (ersetze $x_e \in \{0, 1\}$ durch $x_e \geq 0$). Das duale LP ist

$$\max \left\{ \sum_{v \in V(G)} z_v : z_v + z_w \leq c(e) \text{ für alle } e = \{v, w\} \in E(G) \right\}.$$

Damit bekommen wir die folgende Umformulierung der Ungarischen Methode.

Proposition 11.3. *Sei G ein Graph und $c : E(G) \rightarrow \mathbb{R}$. Sei $z \in \mathbb{R}^{V(G)}$ mit $z_v + z_w \leq c(e)$ für alle $e = \{v, w\} \in E(G)$. Sei $G_z := (V(G), \{e = \{v, w\} \in E(G) : z_v + z_w = c(e)\})$. Sei M ein Matching in G_z , und sei F ein maximaler alternierender Wald in G_z bezüglich M . Mit X bzw. Y bezeichnen wir die Menge der inneren bzw. äußeren Knoten.*

- (a) Ist M ein perfektes Matching, so ist es ein perfektes Matching minimalen Gewichtes in G .
- (b) Ist $\Gamma_G(y) \subseteq X$ für alle $y \in Y$, so ist M ein kardinalitätsmaximales Matching in G .
- (c) Andernfalls sei

$$\begin{aligned} \varepsilon &:= \min \{ \min \{ (c(e) - z_v - z_w)/2 : e = \{v, w\} \in E(G[Y]) \}, \\ &\quad \min \{ c(e) - z_v - z_w : e = \{v, w\} \in \delta_G(Y) \cap \delta_G(V(F)) \} \} \end{aligned}$$

und setze $z'_v := z_v - \varepsilon$ für $v \in X$, $z'_v := z_v + \varepsilon$ für $v \in Y$ und $z'_v := z_v$ für $v \in V(G) \setminus V(F)$. Dann gilt $z'_v + z'_{w'} \leq c(e)$ für alle $e = \{v, w\} \in E(G)$, $M \cup E(F) \subseteq E(G_{z'})$, und $\Gamma_{G_{z'}}(y) \setminus X \neq \emptyset$ für einige $y \in Y$.

Beweis: (a) folgt mittels komplementären Schlupfes: für jedes andere perfekte Matching M' haben wir $\sum_{e \in M'} c(e) = \sum_{v \in V(G)} z_v + \sum_{e=\{v,w\} \in M'} (c(e) - z_v - z_w) \geq \sum_{v \in V(G)} z_v = \sum_{v \in V(G)} z_v + \sum_{e=\{v,w\} \in M} (c(e) - z_v - z_w) = \sum_{e \in M} c(e)$.

(b) folgt mit der Berge-Tutte-Formel (Theorem 10.14): Jeder äußere Knoten ist eine ungerade Zusammenhangskomponente von $G - X$, und genau $|Y| - |X|$ Knoten werden von M nicht überdeckt.

(c) folgt aus der Wahl von ε . □

Beachte, dass dieses Resultat auch für nicht-bipartite Graphen gilt. Ist G bipartit, so können wir nach jeder dualen Modifizierung im Fall (c) den alternierenden Wald anwachsen lassen oder das Matching augmentieren. Irgendwann werden wir mit Fall (a) enden, d. h. mit einem perfekten Matching minimalen Gewichtes, welches dann auch eine optimale LP-Lösung ist, da der Inzidenzvektor von M und z die Bedingungen des komplementären Schlupfes erfüllen (cf. Corollary 3.21). Somit ist Q in dem folgenden Satz ganzzahlig. Wir geben nun einen anderen Beweis an.

Satz 11.4. Sei G ein Graph und

$$\begin{aligned} P &:= \left\{ x \in \mathbb{R}_+^{E(G)} : \sum_{e \in \delta(v)} x_e \leq 1 \text{ für alle } v \in V(G) \right\} \text{ bzw.} \\ Q &:= \left\{ x \in \mathbb{R}_+^{E(G)} : \sum_{e \in \delta(v)} x_e = 1 \text{ für alle } v \in V(G) \right\} \end{aligned}$$

das **gebrochene Matching-Polytop** bzw. das **gebrochene Perfekte-Matching-Polytop** von G . Ist G bipartit, so sind P und Q beide ganzzahlig.

Beweis: Ist G bipartit, so ist die Inzidenzmatrix von G nach Satz 5.26 vollständig unimodular. Somit ist P nach dem Satz von Hoffman und Kruskal (Satz 5.21) ganzzahlig. Es ist Q eine Seitenfläche von P und demnach auch ganzzahlig. \square

Zu obigem Satz gibt es ein schönes Korollar über doppelt-stochastische Matrizen. Eine **doppelt-stochastische Matrix** ist eine nichtnegative quadratische Matrix, deren Zeilen- und Spaltensummen alle gleich 1 sind. Ganzzahlige doppelt-stochastische Matrizen heißen **Permutationsmatrizen**.

Korollar 11.5. (Birkhoff [1946], von Neumann [1953]) Jede doppelt-stochastische Matrix M kann als Konvexitätskombination von Permutationsmatrizen P_1, \dots, P_k (d. h. $M = c_1 P_1 + \dots + c_k P_k$ für nichtnegative c_1, \dots, c_k mit $c_1 + \dots + c_k = 1$) geschrieben werden.

Beweis: Sei $M = (m_{ij})_{i,j \in \{1, \dots, n\}}$ eine doppelt-stochastische $n \times n$ -Matrix und $K_{n,n}$ der vollständige bipartite Graph mit Bipartition $\{a_1, \dots, a_n\} \cup \{b_1, \dots, b_n\}$. Für $e = \{a_i, b_j\} \in E(K_{n,n})$ sei $x_e = m_{ij}$. Da M doppelt-stochastisch ist, liegt x in dem gebrochenen Perfekten-Matching-Polytop Q von $K_{n,n}$. Nach Satz 11.4 und Korollar 3.32 kann x als Konvexitätskombination der ganzzahligen Knoten von Q geschrieben werden. Diese entsprechen offensichtlich Permutationsmatrizen. \square

Dieses Korollar, das auch als der Birkhoff-von-Neumann-Satz bekannt ist, kann auch direkt bewiesen werden (Aufgabe 3).

Mehrere allgemeinere Zuordnungsprobleme sind bearbeitet worden; siehe Burkard, Dell'Amico und Martello [2009].

11.2 Abriss des gewichteten Matching-Algorithmus

In diesem und dem nächsten Abschnitt werden wir einen polynomiellen Algorithmus für das allgemeine MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM beschreiben. Dieser Algorithmus ist von Edmonds [1965] entwickelt worden und verwendet die Begriffe seines Algorithmus für das KARDINALITÄTS-MATCHING-PROBLEM (Abschnitt 10.5).

Wir werden zunächst die Hauptideen beschreiben, ohne auf die Implementierung einzugehen. Für einen gegebenen Graphen G mit Gewichten $c : E(G) \rightarrow \mathbb{R}$ kann das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM folgendermaßen als ganzzahliges LP formuliert werden:

$$\min \left\{ \sum_{e \in E(G)} c(e)x_e : x_e \in \{0, 1\} \ (e \in E(G)), \sum_{e \in \delta(v)} x_e = 1 \ (v \in V(G)) \right\}.$$

Ist A eine Teilmenge von $V(G)$ mit ungerader Kardinalität, so enthält jedes perfekte Matching eine ungerade Anzahl von Kanten aus $\delta(A)$, insbesondere mindestens eine. Somit ändert das Hinzufügen der Nebenbedingung

$$\sum_{e \in \delta(A)} x_e \geq 1$$

gar nichts. Im weiteren Verlauf dieses Kapitels werden wir die Notation $\mathcal{A} := \{A \subseteq V(G) : |A| \text{ ungerade}\}$ benutzen. Wir betrachten nun die folgende LP-Relaxierung:

$$\begin{aligned} \min \quad & \sum_{e \in E(G)} c(e)x_e \\ \text{bzgl.} \quad & \begin{aligned} x_e &\geq 0 && (e \in E(G)) \\ \sum_{e \in \delta(v)} x_e &= 1 && (v \in V(G)) \\ \sum_{e \in \delta(A)} x_e &\geq 1 && (A \in \mathcal{A}, |A| > 1). \end{aligned} \end{aligned} \tag{11.1}$$

Wir werden später noch beweisen, dass das durch (11.1) definierte Polytop ganzzahlig ist; somit beschreibt dieses LP das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM (dies bildet den Inhalt von Satz 11.15, eines der Hauptresultate dieses Kapitels). Für die folgenden Überlegungen benötigen wir diese Tatsache nicht, sondern werden die LP-Formulierung als Motivation heranziehen.

Um das Dual von (11.1) zu formulieren, fügen wir für jede Nebenbedingung des primalen LP, d. h. für jedes $A \in \mathcal{A}$, eine neue Variable z_A hinzu. Dann lautet das duale LP:

$$\begin{aligned} \max \quad & \sum_{A \in \mathcal{A}} z_A \\ \text{bzgl.} \quad & \begin{aligned} z_A &\geq 0 && (A \in \mathcal{A}, |A| > 1) \\ \sum_{A \in \mathcal{A}: e \in \delta(A)} z_A &\leq c(e) && (e \in E(G)). \end{aligned} \end{aligned} \tag{11.2}$$

Beachte, dass die dualen Variablen $z_{\{v\}}$ für $v \in V(G)$ nicht der Beschränkung unterliegen, nichtnegativ zu sein. Edmonds' Algorithmus ist ein primal-dualer Algorithmus. Er beginnt mit dem leeren Matching ($x_e = 0$ für alle $e \in E(G)$) und der zulässigen dualen Lösung

$$z_A := \begin{cases} \frac{1}{2} \min\{c(e) : e \in \delta(A)\} & \text{für } |A| = 1 \\ 0 & \text{sonst} \end{cases}.$$

Zu jedem Zeitpunkt des Algorithmus ist z eine zulässige duale Lösung und wir haben

$$\begin{aligned} x_e > 0 &\Rightarrow \sum_{A \in \mathcal{A}: e \in \delta(A)} z_A = c(e); \\ z_A > 0 &\Rightarrow \sum_{e \in \delta(A)} x_e \leq 1. \end{aligned} \quad (11.3)$$

Der Algorithmus terminiert, wenn x der Inzidenzvektor eines perfekten Matchings ist (d. h. es liegt primale Zulässigkeit vor). Wegen der Bedingungen des komplementären Schlupfes (11.3) (Korollar 3.23) haben wir dann die Optimalität der primalen und dualen Lösungen. Da x für (11.1) optimal und auch ganzzahlig ist, ist x der Inzidenzvektor eines perfekten Matchings minimalen Gewichtes.

Für eine gegebene zulässige duale Lösung z heißt eine Kante e **straff**, falls die entsprechende duale Nebenbedingung mit Gleichheit erfüllt ist, d. h., falls

$$\sum_{A \in \mathcal{A}: e \in \delta(A)} z_A = c(e).$$

Es ist klar, dass zu jedem Zeitpunkt das aktuelle Matching nur aus straffen Kanten besteht. Der Algorithmus läuft wie in Proposition 11.3 (gilt auch für nicht-bipartite Graphen) ab. Falls jedoch G nicht-bipartit ist, so können Blüten auftreten. Kontrahieren wir diese, so können wir fortfahren, werden aber duale Variablen $z_B > 0$ für $B \in \mathcal{A}$ bekommen.

Sei von hier an G_z derjenige Graph, der aus G durch das Entfernen aller nicht straffen Kanten und das Kontrahieren einer jeden Menge B mit $z_B > 0$ zu einem einzigen Knoten hervorgeht. Die Familie $\mathcal{B} := \{B \in \mathcal{A} : |B| = 1 \text{ oder } z_B > 0\}$ ist stets laminar, und für jedes Element $B \in \mathcal{B}$ gibt es einen faktorkritischen, nur aus straffen Kanten bestehenden Teilgraphen mit Knotenmenge B . Anfänglich besteht \mathcal{B} aus den einelementigen Knotenmengen.

Eine Iteration des Algorithmus verläuft in groben Zügen folgendermaßen. Zunächst bestimmen wir mittels EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS ein kardinalitätsmaximales Matching M in G_z . Ist M ein perfektes Matching, so sind wir fertig: Wir können M zu einem perfekten Matching in G allein durch das Hinzufügen straffer Kanten erweitern. Da die Bedingungen (11.3) erfüllt sind, ist das Matching optimal.

Ist M andererseits kein perfektes Matching, so betrachten wir die Gallai-Edmonds-Dekomposition W, X, Y von G_z (siehe Satz 10.32). Für jeden Knoten v von G_z sei $B(v) \in \mathcal{B}$ diejenige Knotenmenge, deren Kontraktion v ergab. Nun modifizieren wir die duale Lösung wie folgt (Abb. 11.1 zeigt ein Beispiel). Für jedes $v \in X$ verringern wir $z_{B(v)}$ um eine gewisse positive Konstante ε . Für jede Zusammensetzungskomponente C von $G_z[Y]$ erhöhen wir z_A um ε , wobei $A = \bigcup_{v \in C} B(v)$.

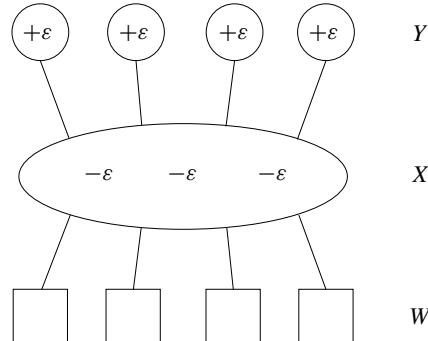


Abbildung 11.1.

Beachte, dass straffe Matching-Kanten straff bleiben, da alle Matching-Kanten mit einem Endknoten in X nach Satz 10.32 den anderen in Y haben. (In der Tat bleiben alle Kanten des alternierenden Waldes, mit dem wir arbeiten, straff.)

Wir wählen ε so groß wie möglich unter Erhaltung der dualen Zulässigkeit. Da der aktuelle Graph kein perfektes Matching enthält, ist die Anzahl der Zusammenhangskomponenten von $G_z[Y]$ größer als $|X|$. Somit erhöht die obige Modifizierung der dualen Lösung den dualen Zielfunktionswert $\sum_{A \in \mathcal{A}} z_A$ um mindestens ε . Kann man ε beliebig groß wählen, so ist das duale LP (11.2) unbeschränkt, also ist das primale LP (11.1) unzulässig (Satz 3.27) und damit besitzt G kein perfektes Matching.

Die obige Modifizierung der dualen Lösung ändert auch den Graphen G_z : Neue Kanten können straff werden, neue Knotenmengen können kontrahiert werden (entsprechend denjenigen Komponenten von Y , die nicht einelementig sind), und einige kontrahierte Mengen können „ausgepackt“ werden (die nicht einelementigen Knotenmengen, deren duale Variablen Null werden; diese entsprechen Knoten in X).

Der oben beschriebene Schritt wird so oft iteriert, bis ein perfektes Matching gefunden worden ist. Wir werden später noch zeigen, dass dieser Prozess endlich ist. Dies wird aus der Tatsache folgen, dass jeder Schritt (Anwachsen, Schrumpfen, Entkontrahieren) zwischen zwei Augmentierungen die Anzahl der äußeren Knoten erhöht.

11.3 Implementierung des gewichteten Matching-Algorithmus

Nach diesem Überblick des GEWICHTETEN MATCHING-ALGORITHMUS wenden wir uns den Details seiner Implementierung zu. Wie bei EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS, schrumpfen wir Blüten nicht explizit, sondern speichern ihre Ohrenzerlegungen. Es ergeben sich jedoch einige Schwierigkeiten.

Der „Schrumpfungs“-Schritt in EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS produziert eine äußere Blüte. Bei dem „Augmentierungs“-Schritt

wandern zwei Zusammenhangskomponenten des Blütenwaldes außerhalb des Waldes. Da die duale Lösung unverändert bleibt, müssen wir die Blüten beibehalten: Wir erhalten sogenannte Blüten **außerhalb des Waldes**. Der „Anwachsen“-Schritt kann Blüten außerhalb des Waldes betreffen, die dann innere oder äußere Blüten werden. Also müssen wir mit allgemeinen Blütenwäldern arbeiten.

Ein weiteres Problem ist, dass wir laminare Blüten schrittweise wiederherstellen können müssen. Genauer: Wird z_A Null für eine innere Blüte A , so könnte es Teilmengen $A' \subseteq A$ mit $|A'| > 1$ und $z_{A'} > 0$ geben. Dann müssen wir die Blüte A „auspacken“, aber nicht die kleineren Blüten in A (außer wenn sie innere bleiben und ihre dualen Variablen auch gleich Null sind).

Während des gesamten Algorithmus haben wir eine laminare Familie $\mathcal{B} \subseteq \mathcal{A}$, die wenigstens alle eelementigen Knotenmengen enthält. Alle Elemente von \mathcal{B} sind Blüten. Es gilt $z_A = 0$ für alle $A \notin \mathcal{B}$. Die Menge \mathcal{B} ist laminar und wird mittels einer Baumdarstellung gespeichert (siehe Proposition 2.14). Als Referenzmarkierung wird jeder Blüte in \mathcal{B} , die nicht eine eelementige Knotenmenge ist, eine Zahl zugeordnet. Ferner werden wir es öfters mit inklusionsmaximalen Blüten und Unterblüten zu tun haben; diese werden wir, um die Darstellung des Algorithmus nicht zu überfrachten, **maximale** (Unter-)Blüten nennen.

Die Ohrenzerlegungen aller Blüten in \mathcal{B} werden stets gespeichert. Die Variablen $\mu(x)$ für $x \in V(G)$ speichern hier wieder das aktuelle Matching M . Mit $b^1(x), \dots, b^{k_x}(x)$ bezeichnen wir die x enthaltenden Blüten in \mathcal{B} , aber ohne die eelementige Blüte x . Hierbei ist $b^{k_x}(x)$ die äußerste Blüte. Ferner haben wir die Variablen $\rho^i(x)$ und $\varphi^i(x)$ für jedes $x \in V(G)$ und $i = 1, \dots, k_x$. Es ist $\rho^i(x)$ die Basis der Blüte $b^i(x)$. Die Variablen $\mu(x)$ und $\varphi^j(x)$ für alle x und j mit $b^j(x) = i$ gehören zu einer M -alternierenden Ohrenzerlegung der Blüte i .

Natürlich müssen wir die Blütenstrukturen (φ und ρ) nach jeder Augmentierung aktualisieren. Die Aktualisierung von ρ ist einfach. Nach Lemma 10.23 kann die Aktualisierung von φ auch in linearer Zeit bewerkstelligt werden.

Für innere Blüten benötigen wir, zusätzlich zur Blütenbasis, erstens den der Wurzel des Baumes am nächsten gelegenen Knoten im allgemeinen Blütenwald und zweitens den Nachbarn in der nächsten äußeren Blüte. Diese beiden Knoten bezeichnen wir mit $\sigma(x)$ und $\chi(\sigma(x))$ für jede Basis x einer inneren Blüte. Abbildung 11.2 zeigt ein Beispiel hierzu.

Mittels dieser Variablen können wir die alternierenden Wege zu der Wurzel des Baumes bestimmen. Da die Blüten nach einer Augmentierung noch vorhanden sind, müssen wir den augmentierenden Weg so wählen, dass jede Blüte danach immer noch ein fast perfektes Matching enthält.

Abbildung 11.2 demonstriert, dass wir vorsichtig sein müssen: Hier liegen die zwei von $\{x_3, x_4, x_5\}$ und $\{x_1, x_2, x_3, x_4, x_5\}$ induzierten laminaren inneren Blüten vor. Betrachten wir bloß die Ohrenzerlegung der äußersten Blüte, um einen alternierenden Weg von x_0 zur Wurzel x_6 zu bestimmen, so bekommen wir $(x_0, x_1, x_4, x_5 = \sigma(x_1), x_6 = \chi(x_5))$. Nachdem wir entlang $(y_6, y_5, y_4, y_3, y_2, y_1, y_0, x_0, x_1, x_4, x_5, x_6)$ augmentiert haben, enthält der durch $\{x_3, x_4, x_5\}$ induzierte faktorkritische Teilgraph kein fast perfektes Matching mehr.

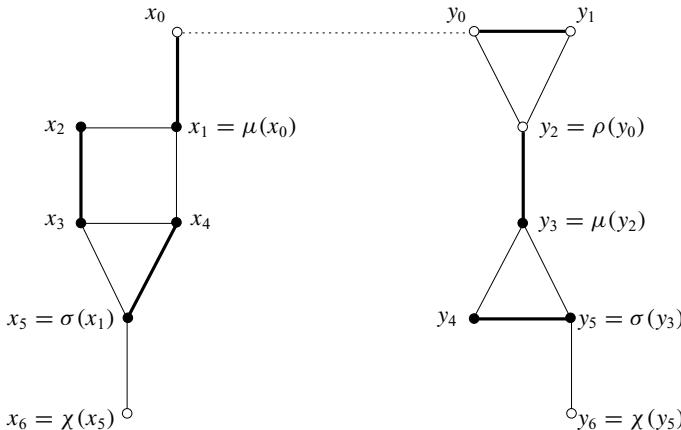


Abbildung 11.2.

Also müssen wir einen alternierenden Weg in jeder Blüte finden, der eine gerade Anzahl von Kanten in jeder Unterblüte enthält. Dies erreichen wir mit dem folgenden Verfahren:

BLÜTENWEG

Input: Ein Knoten x_0 .

Output: Ein M -alternierender Weg $Q(x_0)$ von x_0 nach $\rho^{k_{x_0}}(x_0)$.

- ① Setze $h := 0$ und $B := \{b^j(x_0) : j = 1, \dots, k_{x_0}\}$.
- ② **While** $x_{2h} \neq \rho^{k_{x_0}}(x_0)$ **do:**
 - Setze $x_{2h+1} := \mu(x_{2h})$ und $x_{2h+2} := \varphi^i(x_{2h+1})$, wobei $i = \min\{j \in \{1, \dots, k_{x_{2h+1}}\} : b^j(x_{2h+1}) \in B\}$.
 - Füge alle x_{2h+2} aber nicht x_{2h+1} enthaltenden Blüten in \mathcal{B} zu B hinzu.
 - Entferne alle Blüten mit Basis x_{2h+2} aus B .
 - Setze $h := h + 1$.
- ③ Sei $Q(x_0)$ der Weg mit den Knoten x_0, x_1, \dots, x_{2h} .

Proposition 11.6. Das Verfahren BLÜTENWEG kann in $O(n)$ -Zeit implementiert werden. $M \triangle E(Q(x_0))$ enthält innerhalb jeder Blüte ein fast perfektes Matching.

Beweis: Zunächst prüfen wir, ob das Verfahren überhaupt einen Weg berechnet. Wird eine Blüte in \mathcal{B} verlassen, so wird sie nie wieder besucht. Dies folgt aus der Tatsache, dass die Kontraktion der maximalen Unterblüten einer Blüte in \mathcal{B} einen Kreis bildet (eine Eigenschaft, die erhalten bleibt).

Am Anfang einer Iteration ist B die Liste aller Blüten, die entweder x_0 enthalten oder über eine Nicht-Matching-Kante besucht worden sind und noch nicht wieder verlassen wurden. Der konstruierte Weg verlässt eine Blüte in \mathcal{B} immer über eine

Matching-Kante. Also ist die Anzahl der Kanten in jeder Blüte gerade, womit die zweite Aussage bewiesen ist.

Bei der Implementierung des Verfahrens in $O(n)$ -Zeit ist die einzige nicht triviale Arbeit die Aktualisierung von B . Es wird B als sortierte Liste gespeichert. Unter Verwendung der Baumdarstellung von \mathcal{B} und der Tatsache, dass jede Blüte höchstens einmal besucht und verlassen wird, erreichen wir eine $O(n + |\mathcal{B}|)$ Laufzeit. Beachte, dass $|\mathcal{B}| = O(n)$, da \mathcal{B} laminar ist. \square

Die Bestimmung eines augmentierenden Weges besteht aus der Anwendung des Verfahrens BLÜTENWEG innerhalb von Blüten und der Verwendung von μ und χ zwischen Blüten. Treffen wir benachbarte äußere Knoten x, y in verschiedenen Bäumen des allgemeinen Blütenwaldes an, so wenden wir das folgende Verfahren sowohl auf x als auch auf y an. Die Vereinigung der beiden Wege und der Kante $\{x, y\}$ ergibt dann den augmentierenden Weg.

BAUMWEG

Input: Ein äußerer Knoten v .

Output: Ein alternierender Weg $P(v)$ von v zur Wurzel des Baumes in dem Blütenwald.

- ① Es bestehe $P(v)$ am Anfang nur aus v . Sei $x := v$.
- ② Sei $y := \rho^{k_x}(x)$. Sei $Q(x) := \text{BLÜTENWEG}(x)$. Hänge $Q(x)$ an $P(v)$ an.
If $\mu(y) = y$ **then stop**.
- ③ Setze $P(v) := P(v) + \{y, \mu(y)\}$.
Sei $Q(\sigma(\mu(y))) := \text{BLÜTENWEG}(\sigma(\mu(y)))$.
Hänge den gegenläufigen Weg des Weges $Q(\sigma(\mu(y)))$ an $P(v)$ an.
Sei $P(v) := P(v) + \{\sigma(\mu(y)), \chi(\sigma(\mu(y)))\}$.
Setze $x := \chi(\sigma(\mu(y)))$ und **go to** ②.

Die zweite größere Schwierigkeit ist die effiziente Bestimmung von ε . Nachdem alle Anwachs-, Schrumpf- und Augmentierungsschritte ausgeführt worden sind, liefert der allgemeine Blütenwald die Gallai-Edmonds-Dekomposition W, X, Y von G_z . Es enthält W die Blüten außerhalb des Waldes, X die inneren Blüten und Y die äußeren Blüten.

Um die Notation zu vereinfachen, setzen wir $c(\{v, w\}) := \infty$ für $\{v, w\} \notin E(G)$. Ferner benutzen wir die Abkürzung

$$\text{slack}(v, w) := c(\{v, w\}) - \sum_{A \in \mathcal{A}, \{v, w\} \in \delta(A)} z_A.$$

Somit ist $\{v, w\}$ genau dann eine straffe Kante, wenn $\text{slack}(v, w) = 0$. Seien ferner

$$\begin{aligned}
\varepsilon_1 &:= \min \{z_A : A \text{ ist eine maximale innere Blüte mit } |A| > 1\}; \\
\varepsilon_2 &:= \min \{\text{slack}(x, y) : x \text{ ist ein äußerer Knoten, } y \text{ ist ein Knoten außerhalb des Waldes}\}; \\
\varepsilon_3 &:= \frac{1}{2} \min \{\text{slack}(x, y) : x, y \text{ sind zu verschiedenen Blüten gehörende äußere Knoten}\}; \\
\varepsilon &:= \min \{\varepsilon_1, \varepsilon_2, \varepsilon_3\}.
\end{aligned}$$

Es ist ε die maximale Zahl mit der Eigenschaft, dass die duale Modifizierung um ε die duale Zulässigkeit erhält. Ist $\varepsilon = \infty$, so ist (11.2) unbeschränkt und folglich ist (11.1) unzulässig. In diesem Fall hat G kein perfektes Matching.

Offensichtlich kann ε in endlicher Zeit berechnet werden. Um jedoch eine $O(n^3)$ Gesamlaufzeit zu erreichen, müssen wir ε in $O(n)$ -Zeit berechnen können. Für ε_1 ist dies einfach, für ε_2 und ε_3 benötigt man aber weitere Datenstrukturen.

Sei für $A \in \mathcal{B}$

$$\zeta_A := \sum_{B \in \mathcal{B}: A \subseteq B} z_B.$$

Bei jeder Modifizierung der dualen Lösung werden wir diese Größen aktualisieren; dies kann leicht in linearer Zeit erledigt werden (mittels der Baumdarstellung von \mathcal{B}). Damit sind

$$\begin{aligned}
\varepsilon_2 &= \min \{c(\{x, y\}) - \zeta_{\{x\}} - \zeta_{\{y\}} : x \text{ ist ein äußerer Knoten, } y \text{ ist ein Knoten außerhalb des Waldes}\}, \\
\varepsilon_3 &= \frac{1}{2} \min \{c(\{x, y\}) - \zeta_{\{x\}} - \zeta_{\{y\}} : x, y \text{ sind äußere Knoten mit } \{x, y\} \not\subseteq B \\
&\quad \text{für } B \in \mathcal{B}\}.
\end{aligned}$$

Wir führen nun die Variablen t_v^A und τ_v^A für jeden äußeren Knoten v und jedes $A \in \mathcal{B}$ ein, sofern es kein $B \in \mathcal{B}$ mit $A \cup \{v\} \subseteq B$ gibt. Es ist τ_v^A ein Knoten in A , der $\text{slack}(v, \tau_v^A)$ minimiert, und $t_v^A := \text{slack}(v, \tau_v^A) + \Delta + \zeta_A$, wobei Δ die Summe der ε -Werte in allen dualen Modifizierungen bezeichnet. Beachte, dass sich t_v^A nicht verändert, solange v äußerer Knoten bleibt und $A \in \mathcal{B}$ ist. Schließlich setzen wir noch $t^A := \min\{t_v^A : v \notin A, v \text{ ist ein äußerer Knoten}\}$. Damit haben wir

$$\begin{aligned}
\varepsilon_2 &= \min \{\text{slack}(v, \tau_v^A) : v \text{ ist ein äußerer Knoten, } A \in \mathcal{B} \text{ ist eine maximale Blüte außerhalb des Waldes}\}, \\
&= \min \{t^A - \Delta - \zeta_A : A \in \mathcal{B} \text{ ist eine maximale Blüte außerhalb des Waldes}\},
\end{aligned}$$

und ebenso folgt

$$\varepsilon_3 = \frac{1}{2} \min \{t^A - \Delta - \zeta_A : A \in \mathcal{B} \text{ ist eine maximale äußere Blüte}\}.$$

Obwohl wir bei der Berechnung von ε_2 und ε_3 nur an den Werten t_v^A für maximale Blüten außerhalb des Waldes und maximale äußere Blüten in \mathcal{B} interessiert sind, aktualisieren wir diese Variablen auch für innere Blüten und für nicht-maximale, da diese später relevant werden könnten. Nicht-maximale äußere Blüten können erst nach einer weiteren Augmentierung maximale äußere Blüten werden. All diese Variablen werden jedoch nach jeder Augmentierung neu berechnet.

Am Anfang, aber auch nach jeder Augmentierung und auch wenn ein nicht bereits äußerer Knoten v zum äußeren Knoten wird, müssen wir für alle $A \in \mathcal{B}$ (außer den nicht-maximalen äußeren Blüten), τ_v^A und t_v^A neu berechnen und eventuell auch t^A aktualisieren. Dies kann wie folgt bewerkstelligt werden:

UPDATE

Input: Ein äußerer Knoten v .

Output: Aktualisierte Werte der Variablen τ_v^A , t_v^A und t^A für alle $A \in \mathcal{B}$ und der Variablen τ_w für alle Knoten w außerhalb des Waldes.

- ① **For** jedes $x \in V(G)$ **do:** Setze $\tau_v^{\{x\}} := x$ und $t_v^{\{x\}} := c(\{v, x\}) - \zeta_{\{v\}} + \Delta$.
- ② **For** $A \in \mathcal{B}$ mit $|A| > 1$ (geordnet nach nicht abnehmender Kardinalität) **do:** Setze $\tau_v^A := \tau_v^{A'}$ und $t_v^A := t_v^{A'} - \zeta_{A'} + \zeta_A$, wobei A' eine inklusionsmaximale echte Teilmenge von A in \mathcal{B} ist, für die $t_v^{A'} - \zeta_{A'}$ minimal ist.
- ③ **For** $A \in \mathcal{B}$ mit $v \notin A$, außer denjenigen, die nicht-maximale äußere sind, **do:** Setze $t^A := \min\{t^A, t_v^A\}$.

Offensichtlich stimmt diese Berechnung mit den obigen Definitionen von τ_v^A und t_v^A überein. Es ist ausschlaggebend, dass dieses Verfahren lineare Laufzeit hat:

Lemma 11.7. *Ist \mathcal{B} laminar, so kann das Verfahren UPDATE in $O(n)$ -Zeit implementiert werden.*

Beweis: Nach Proposition 2.15 hat eine laminare Familie von Teilmengen von $V(G)$ höchstens Kardinalität $2|V(G)| = O(n)$. Wird \mathcal{B} mittels seiner Baumdarstellung gespeichert, so ist eine Implementierung mit linearer Laufzeit einfach. \square

Wir können uns nun der formalen Beschreibung des Algorithmus zuwenden. Anstatt die inneren und äußeren Knoten mittels ihrer μ -, ϕ - und ρ -Werte zu identifizieren, markieren wir jeden Knoten direkt mit seinem Status (innerer, äußerer, oder außerhalb des Waldes).

GEWICHTETER MATCHING-ALGORITHMUS

Input: Ein Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$.

Output: Ein perfektes mittels der Kanten $\{x, \mu(x)\}$ gegebenes Matching minimalen Gewichtes in G , oder die Antwort, dass G kein perfektes Matching besitzt.

- ① Setze $\mathcal{B} := \{\{v\} : v \in V(G)\}$ und $K := 0$. Setze $\Delta := 0$.
 Setze $z_{\{v\}} := \frac{1}{2} \min\{c(e) : e \in \delta(v)\}$ und $\zeta_{\{v\}} := z_{\{v\}}$ für alle $v \in V(G)$.
 Setze $k_v := 0$, $\mu(v) := v$, $\rho^0(v) := v$ und $\varphi^0(v) := v$ für alle $v \in V(G)$.
 Markiere alle Knoten als äußere Knoten.
- ② Setze $t^A := \infty$ für alle $A \in \mathcal{B}$.
For alle äußeren Knoten v **do:** UPDATE(v).
- ③ („duale Modifizierung“)
 Setze $\varepsilon_1 := \min\{z_A : A \text{ maximales inneres Element von } \mathcal{B} \text{ mit } |A| > 1\}$.
 Setze $\varepsilon_2 := \min\{t^A - \Delta - \zeta_A : A \text{ maximales Element von } \mathcal{B} \text{ außerhalb des Waldes}\}$.
 Setze $\varepsilon_3 := \min\{\frac{1}{2}(t^A - \Delta - \zeta_A) : A \text{ maximales äußeres Element von } \mathcal{B}\}$.
 Setze $\varepsilon := \min\{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$. **If** $\varepsilon = \infty$ **then stop** (G besitzt kein perfektes Matching).
For jedes maximale äußere Element A von \mathcal{B} **do:**
 Setze $z_A := z_A + \varepsilon$ und $\zeta_{A'} := \zeta_{A'} + \varepsilon$ für alle $A' \in \mathcal{B}$ mit $A' \subseteq A$.
For jedes maximale innere Element A von \mathcal{B} **do:**
 Setze $z_A := z_A - \varepsilon$ und $\zeta_{A'} := \zeta_{A'} - \varepsilon$ für alle $A' \in \mathcal{B}$ mit $A' \subseteq A$.
 Setze $\Delta := \Delta + \varepsilon$.
- ④ **If** $\varepsilon = \varepsilon_1$ **then go to** ⑧.
If $\varepsilon = \varepsilon_2$ und $t_x^A - \Delta - \zeta_A = \text{slack}(x, y) = 0$, x äußerer Knoten, $y \in A$ außerhalb des Waldes **then go to** ⑤.
If $\varepsilon = \varepsilon_3$ und $t_x^A - \Delta - \zeta_A = \text{slack}(x, y) = 0$, x, y äußere Knoten, A maximales äußeres Element von \mathcal{B} , $x \notin A$, $y \in A$ **then:**
 Sei $P(x) := \text{TREEPATH}(x)$ gegeben durch $(x = x_0, x_1, x_2, \dots, x_{2h})$.
 Sei $P(y) := \text{TREEPATH}(y)$ gegeben durch $(y = y_0, y_1, y_2, \dots, y_{2j})$.
 If $P(x)$ und $P(y)$ sind knotendisjunkt **then go to** ⑥, **else go to** ⑦.
- ⑤ („anwachsen“)
 Setze $\sigma(\rho^{k_y}(y)) := y$ und $\chi(y) := x$.
 Markiere alle Knoten v mit $\rho^{k_y}(v) = \rho^{k_y}(y)$ als innere Knoten.
 Markiere alle Knoten v mit $\mu(\rho^{k_y}(v)) = \rho^{k_y}(y)$ als äußere Knoten.
For jeden neuen äußeren Knoten v **do:** UPDATE(v).
Go to ③.
- ⑥ („augmentieren“)
For $i := 0$ **to** $h - 1$ **do:** Setze $\mu(x_{2i+1}) := x_{2i+2}$ und $\mu(x_{2i+2}) := x_{2i+1}$.
For $i := 0$ **to** $j - 1$ **do:** Setze $\mu(y_{2i+1}) := y_{2i+2}$ und $\mu(y_{2i+2}) := y_{2i+1}$.
 Setze $\mu(x) := y$ und $\mu(y) := x$.
 Markiere diejenigen Knoten v als außerhalb des Waldes, für die der Endknoten von $\text{BAUMWEG}(v)$ entweder x_{2h} oder y_{2j} ist. Aktualisiere alle Werte von $\varphi^i(v)$ und $\rho^i(v)$ (unter Verwendung von Lemma 10.23).
If $\mu(v) \neq v$ für alle v **then stop**, **else go to** ②.

(7) („schrumpfen“)

Sei $r = x_{2h'} = y_{2j'}$ der erste äußere Knoten von $V(P(x)) \cap V(P(y))$ mit $\rho^{k_r}(r) = r$.

Sei $A := \{v \in V(G) : \rho^{k_v}(v) \in V(P(x)_{[x,r]}) \cup V(P(y)_{[y,r]})\}$.

Setze $K := K + 1$, $\mathcal{B} := \mathcal{B} \cup \{A\}$, $z_A := 0$ und $\zeta_A := 0$.

For alle $v \in A$ **do**:

 Setze $k_v := k_v + 1$, $b^{k_v}(v) := K$, $\rho^{k_v}(v) := r$, $\phi^{k_v}(v) := \phi^{k_v-1}(v)$.

For $i := 1$ **to** h' **do**:

If $\rho^{k_{x_{2i}}-1}(x_{2i}) \neq r$ **then** setze $\phi^{k_{x_{2i}}}(x_{2i}) := x_{2i-1}$.

If $\rho^{k_{x_{2i-1}}-1}(x_{2i-1}) \neq r$ **then** setze $\phi^{k_{x_{2i-1}}}(x_{2i-1}) := x_{2i}$.

For $i := 1$ **to** j' **do**:

If $\rho^{k_{y_{2i}}-1}(y_{2i}) \neq r$ **then** setze $\phi^{k_{y_{2i}}}(y_{2i}) := y_{2i-1}$.

If $\rho^{k_{y_{2i-1}}-1}(y_{2i-1}) \neq r$ **then** setze $\phi^{k_{y_{2i-1}}}(y_{2i-1}) := y_{2i}$.

If $\rho^{k_x-1}(x) \neq r$ **then** setze $\phi^{k_x}(x) := y$.

If $\rho^{k_y-1}(y) \neq r$ **then** setze $\phi^{k_y}(y) := x$.

For jeden äußeren Knoten $v \notin A$ **do**:

 Setze $t_v^A := t_v^{A'} - \zeta_{A'}$ und $\tau_v^A := \tau_v^{A'}$, wobei A' eine inklusionsmaximale echte Teilmenge von A in \mathcal{B} ist, die $t_v^{A'} - \zeta_{A'}$ minimiert.

Setze $t^A := \min\{t_v^A : v \text{ ist äußerer Knoten und es gibt kein } \bar{A} \in \mathcal{B} \text{ mit } A \cup \{v\} \subseteq \bar{A}\}$.

Markiere alle $v \in A$ als äußere Knoten.

For jeden neuen äußeren Knoten v **do**: UPDATE(v).

Go to ③.

(8) („auspacken“)

Sei $A \in \mathcal{B}$ eine maximale innere Blüte mit $z_A = 0$ und $|A| > 1$.

Setze $\mathcal{B} := \mathcal{B} \setminus \{A\}$.

Sei $y := \sigma(\rho^{k_v}(v))$ für ein $v \in A$.

Sei $Q(y) := \text{BLÜTENWEG}(y)$ gegeben durch

$(y = r_0, r_1, r_2, \dots, r_{2l-1}, r_{2l} = \rho^{k_y}(y))$.

Markiere alle $v \in A$ mit $\rho^{k_v-1}(v) \notin V(Q(y))$ als außerhalb des Waldes.

Markiere alle $v \in A$ mit $\rho^{k_v-1}(v) = r_{2i-1}$ für ein i als äußere Knoten.

For alle $v \in A$ mit $\rho^{k_v-1}(v) = r_{2i}$ für ein i (v bleibt innerer Knoten) **do**:

 Setze $\sigma(\rho^{k_v}(v)) := r_j$ und $\chi(r_j) := r_{j-1}$, wobei

$j := \min\{j' \in \{0, \dots, 2l\} : \rho^{k_{r_{j'}}-1}(r_{j'}) = \rho^{k_v-1}(v)\}$.

For alle $v \in A$ **do**: Setze $k_v := k_v - 1$.

For jeden neuen äußeren Knoten v **do**: UPDATE(v).

Go to ③.

Beachte, dass $\varepsilon = 0$ möglich ist, im Gegensatz zu unserer früheren Diskussion. Die Variablen τ_v^A werden nicht explizit gebraucht. Zu dem „auspacken“-Schritt ⑧ gibt Abb. 11.3 ein Beispiel, wo eine Blüte mit 19 Knoten „ausgepackt“ wird. Zwei der fünf Unterblüten werden zu Blüten außerhalb des Waldes, zwei werden innere Blüten und eine wird äußere Blüte.

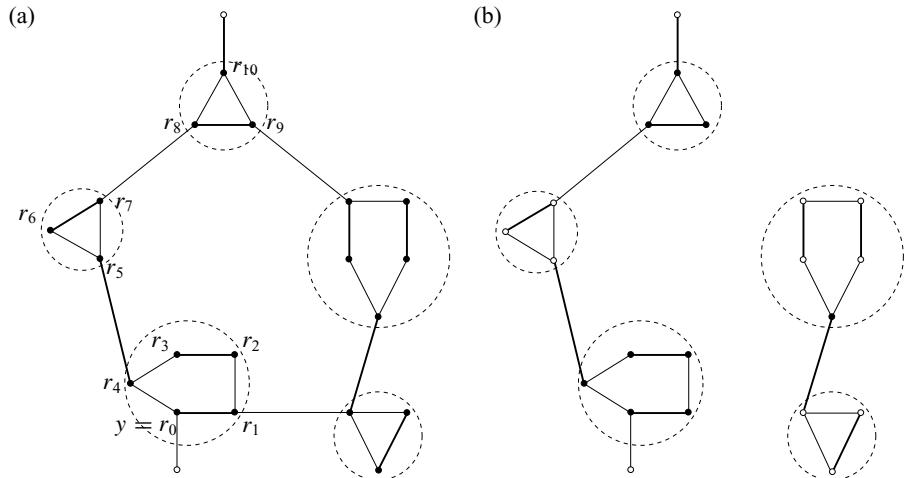


Abbildung 11.3.

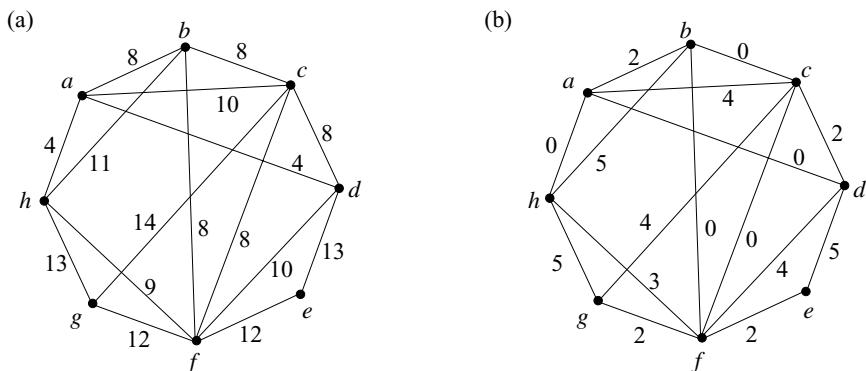


Abbildung 11.4.

Bevor wir den Algorithmus analysieren, werden wir seine Hauptschritte an einem Beispiel erläutern. Betrachte den in Abb. 11.4(a) angegebenen Graphen. Am Anfang setzt der Algorithmus $z_{\{a\}} = z_{\{d\}} = z_{\{h\}} = 2$, $z_{\{b\}} = z_{\{c\}} = z_{\{f\}} = 4$ und $z_{\{e\}} = z_{\{g\}} = 6$. In Abb. 11.4(b) sind die Schlüpfen sichtbar. Somit sind die Kanten $\{a, d\}$, $\{a, h\}$, $\{b, c\}$, $\{b, f\}$, $\{c, f\}$ am Anfang straff. Also wird während der ersten Iterationen $\epsilon = 0$ sein.

Wir gehen davon aus, dass der Algorithmus die Knoten in alphabetischer Reihenfolge scannt. Somit sind die ersten Schritte

$$\text{augmentiere}(a, d), \quad \text{augmentiere}(b, c), \quad \text{anwachse}(f, b).$$

Abbildung 11.5(a) gibt den aktuellen allgemeinen Blütenwald an.

Die nächsten Schritte sind

$$\text{schrumpfe}(f, c), \quad \text{anwachse}(h, a),$$

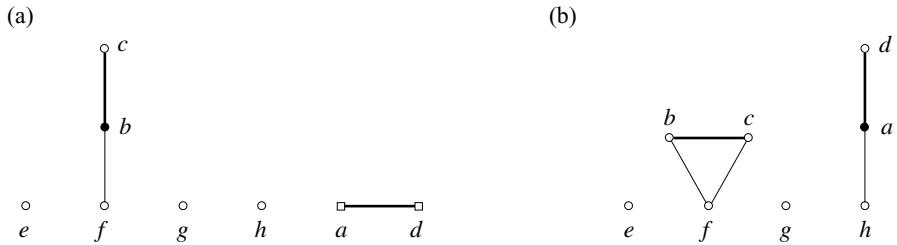


Abbildung 11.5.

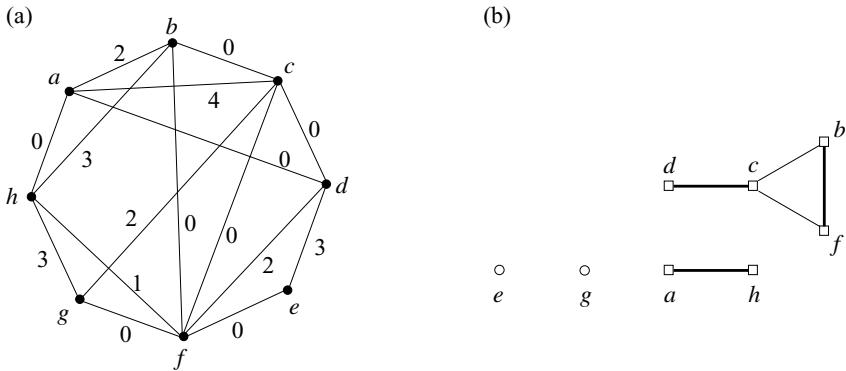


Abbildung 11.6.

die zu dem in Abb. 11.5(b) gezeigten allgemeinen Blütenwald führen. Nun sind alle straffen Kanten aufgebraucht, also müssen die dualen Variablen modifiziert werden. In ③ erhalten wir $\varepsilon = \varepsilon_3 = 1$, $A = \{b, c, f\}$ etwa und $\tau_d^A = c$. Die neuen dualen Variablen sind $z_{\{b, c, f\}} = 1$, $z_{\{a\}} = 1$, $z_{\{d\}} = z_{\{h\}} = 3$, $z_{\{b\}} = z_{\{c\}} = z_{\{f\}} = 4$, $z_{\{e\}} = z_{\{g\}} = 7$. Abbildung 11.6(a) zeigt die aktuellen Schlüpfen. Der nächste Schritt ist

augmentiere(d, c).

Die Blüte $\{b, c, f\}$ wird zu einer Blüte außerhalb des Waldes (Abb. 11.6(b)). Wiederum gilt $\varepsilon = \varepsilon_3 = 0$ in ③, da $\{e, f\}$ straff ist. Die nächsten Schritte sind somit

anwachse(e, f), anwachse(d, a).

Wir sind nun bei Abb. 11.7(a).

Es gibt keine straffen mit äußeren Knoten inzidenten Kanten mehr. Damit haben wir $\varepsilon = \varepsilon_1 = 1$ in ③ und bekommen die neue duale Lösung $z_{\{b, c, f\}} = 0$, $z_{\{a\}} = 0$, $z_{\{d\}} = z_{\{h\}} = z_{\{b\}} = z_{\{c\}} = z_{\{f\}} = 4$, $z_{\{e\}} = z_{\{g\}} = 8$. Abbildung 11.7(b) zeigt die neuen Schlüpfen. Da die duale Variable für die innere Blüte $\{B, C, F\}$ Null geworden ist, ist der nächste Schritt

auspacke($\{b, c, f\}$).

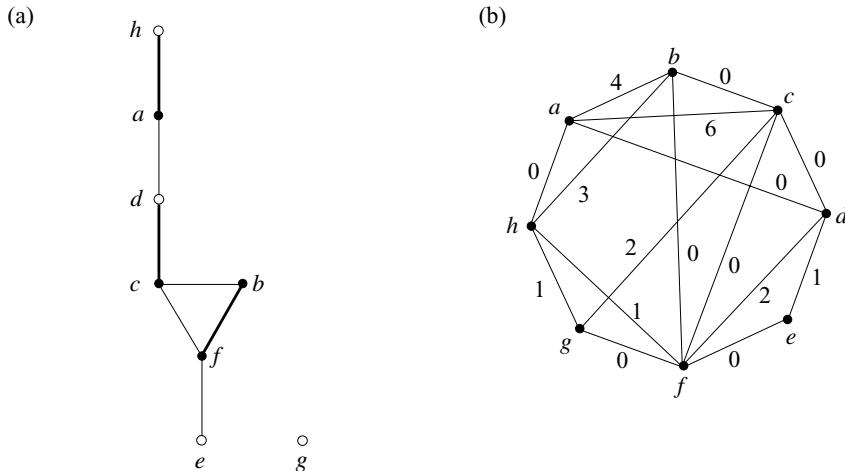


Abbildung 11.7.

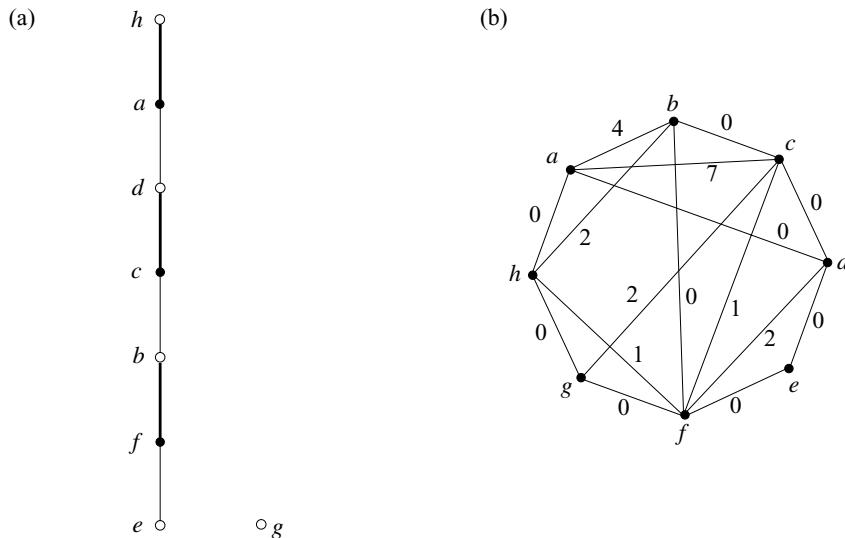


Abbildung 11.8.

Den resultierenden allgemeinen Blütenwald sehen wir in Abb. 11.8(a). Nach einer weiteren Modifizierung der dualen Variablen mit $\varepsilon = \varepsilon_3 = \frac{1}{2}$ haben wir $z_{\{a\}} = -0,5$, $z_{\{c\}} = z_{\{f\}} = 3,5$, $z_{\{b\}} = z_{\{d\}} = z_{\{h\}} = 4,5$, $z_{\{e\}} = z_{\{g\}} = 8,5$ (Abb. 11.8(b) zeigt die neuen Schlüsse). Die abschließenden Schritte sind

$$\text{schrumpfe}(d, e), \quad \text{augmentiere}(g, h),$$

womit der Algorithmus terminiert. Das letzte Matching ist $M = \{\{e, ff\}, \{b, c\}, \{a, d\}, \{g, h\}\}$. Wir prüfen leicht, dass M Gesamtgewicht 37 hat, gleich der Summe der dualen Variablen.

Wir werden nun zeigen, dass der Algorithmus korrekt arbeitet.

Proposition 11.8. *Die folgenden sieben Aussagen gelten zu jedem Zeitpunkt des GEWICHTETEN MATCHING-ALGORITHMUS:*

- (a) *Es sei $X(j) := \{v \in V(G) : j \in \{b^1(v), \dots, b^{k_v}(v)\}\}$ für $j \in \{1, \dots, K\}$. Dann ist $\mathcal{B} = \{X(j) : j = 1, \dots, K\} \cup \{\{v\} : v \in V(G)\}$ eine laminare Familie. Die Mengen $V_r := \{v : \rho^{k_v}(v) = r\}$ für $r \in V(G)$ mit $\rho^{k_r}(r) = r$ sind genau die maximalen Elemente von \mathcal{B} . Die Knoten in jedem V_r sind entweder alle als innere oder alle als äußere Knoten, oder alle als Knoten außerhalb des Waldes markiert. Jedes $(V_r, \{\{v, \varphi^{k_v}(v)\} : v \in V_r \setminus \{r\}\} \cup \{\{v, \mu(v)\} : v \in V_r \setminus \{r\}\})$ ist eine Blüte mit Basis r .*
- (b) *Die Kanten $\{x, \mu(x)\}$ bilden ein Matching M und M enthält ein fast perfektes Matching innerhalb jedes Elementes von \mathcal{B} .*
- (c) *Für $b = 1, \dots, K$ gehören die Variablen $\mu(v)$ und $\varphi^i(v)$ für diejenigen v und i mit $b^i(v) = b$ zu einer M -alternierenden Ohrenzerlegung in $G[X(b)]$.*
- (d) *Die Kanten $\{x, \mu(x)\}$ und $\{x, \varphi^i(x)\}$ für alle x und i und die Kanten $\{\sigma(x)$ und $\chi(\sigma(x))\}$ für alle Basen x maximaler innerer Blüten sind straff.*
- (e) *Die Kanten $\{x, \mu(x)\}, \{x, \varphi^{k_x}(x)\}$ für alle inneren oder äußeren x , zusammen mit den Kanten $\{\sigma(x), \chi(\sigma(x))\}$ für alle Basen x maximaler innerer Blüten, bilden einen allgemeinen Blütenwald F bezüglich M . Die Knotenmarkierungen (innerer, äußerer, außerhalb des Waldes) sind mit F vereinbar.*
- (f) *Die Kontraktion der maximalen Unterblüten einer jeglichen Blüte in $B \in \mathcal{B}$ mit $|B| > 1$ ergibt einen Kreis.*
- (g) *Für jeden äußeren Knoten v ergibt das Verfahren BAUMWEG einen M -alternierenden v - r -Weg, wobei r die Wurzel des v enthaltenden Baumes in F ist.*

Beweis: Es ist klar, dass diese Aussagen am Anfang gelten (nachdem ② erstmalig ausgeführt worden ist). Wir werden nun zeigen, dass sie im weiteren Verlauf des Algorithmus erhalten bleiben. Für (a) sieht man dies leicht mittels ⑦ und ⑧. Für (b) folgt dies nach Proposition 11.6 und der Annahme, dass (f) und (g) vor einer Augmentierung gelten.

Der Beweis, dass (c) nach einer Schrumpfung erhalten bleibt, ist wie im nichtgewichteten Fall (siehe Lemma 10.30 (c)). Die φ -Werte werden nach einer Augmentierung neu berechnet und verändern sich sonst nicht. Aussage (d) wird durch ③ gewährleistet.

Es ist einfach zu sehen, dass (e) in ⑤ erhalten bleibt: Die y enthaltende Blüte war außerhalb des Waldes, und sie wird zu einer inneren Blüte, indem wir $\chi(y) := x$ und $\sigma(v) := y$ für ihre Basis v setzen. Die $\mu(\rho^{k_y}(y))$ enthaltende Blüte war auch außerhalb des Waldes und wird zu einer äußeren Blüte.

In ⑥ werden zwei Zusammenhangskomponenten des allgemeinen Blütenwaldes zu Blüten außerhalb des Waldes, somit bleibt (e) erhalten. In ⑦ werden die Knoten in der neuen Blüte äußere Knoten, da r ein äußerer Knoten war. In ⑧ gilt für die Knoten $v \in A$ mit $\rho^{k_v-1}(v) \notin V(Q(y))$ auch $\mu(\rho^{k_v}(v)) \notin V(Q(y))$, also werden diese zu Knoten außerhalb des Waldes. Für jedes andere $v \in A$ haben wir

$\rho^{k_v-1}(v) = r_k$ für irgendein k . Da $\{r_i, r_{i+1}\} \in M$ genau dann, wenn i gerade ist, folgt, dass v genau dann äußerer Knoten wird, wenn k ungerade ist.

Es gilt (f), da jede neue Blüte aus einem ungeraden Kreis in (7) hervorgeht. Um zu sehen, dass (g) erhalten bleibt, genügt es zu beachten, dass $\sigma(x)$ und $\chi(\sigma(x))$ für alle Basen x maximaler innerer Blüten korrekt gesetzt sind. Dies ist sowohl für (5) als auch für (8) schnell erledigt. \square

Nach Proposition 11.8(a) ist es gerechtfertigt, in (3) und (8) des Algorithmus die maximalen Elemente von \mathcal{B} innere, äußere, oder außerhalb des Waldes zu nennen.

Als Nächstes zeigen wir, dass der Algorithmus jederzeit eine zulässige duale Lösung bereit hält.

Lemma 11.9. *Zu jedem Zeitpunkt des Algorithmus ist z eine zulässige duale Lösung. Ist $\varepsilon = \infty$, so besitzt G kein perfektes Matching.*

Beweis: Es ist immer $z_A = 0$ für alle $A \in \mathcal{A} \setminus \mathcal{B}$. Es wird z_A nur für diejenigen $A \in \mathcal{B}$ verringert, die maximal in \mathcal{B} und innere Elemente sind. Somit gewährleistet die Wahl von ε_1 , dass z_A weiterhin nichtnegativ für alle A mit $|A| > 1$ bleibt.

Wie können die Nebenbedingungen $\sum_{A \in \mathcal{A}: e \in \delta(A)} z_A \leq c(e)$ verletzt werden? Wird $\sum_{A \in \mathcal{A}: e \in \delta(A)} z_A$ in (3) vergrößert, so verbindet e entweder einen äußeren Knoten mit einem Knoten außerhalb des Waldes oder zwei verschiedene äußere Blüten. Somit ist das maximale ε mit der Eigenschaft, dass das neue z weiterhin $\sum_{A \in \mathcal{A}: e \in \delta(A)} z_A \leq c(e)$ erfüllt, gleich $slack(e)$ im ersten Fall und gleich $\frac{1}{2}slack(e)$ im zweiten Fall.

Wir müssen also zeigen, dass ε_2 und ε_3 korrekt berechnet werden:

$$\varepsilon_2 = \min\{slack(v, w) : v \text{ ist äußerer Knoten und } w \text{ ist außerhalb des Waldes}\}$$

und

$$\varepsilon_3 = \frac{1}{2} \min \left\{ slack(v, w) : v \text{ und } w \text{ sind äußere Knoten mit } \rho^{k_v}(v) \neq \rho^{k_w}(w) \right\}.$$

Wir behaupten, dass zu jedem Zeitpunkt des Algorithmus die folgenden fünf Aussagen für jeden äußeren Knoten v und jedes $A \in \mathcal{B}$ mit der Eigenschaft: Es gibt kein $\bar{A} \in \mathcal{B}$ mit $A \cup \{v\} \subseteq \bar{A}$, gelten:

- (a) $\tau_v^A \in A$.
- (b) $slack(v, \tau_v^A) = \min\{slack(v, u) : u \in A\}$.
- (c) $\zeta_A = \sum_{B \in \mathcal{B}: A \subseteq B} z_B$. Es ist Δ die Summe der ε -Werte in allen bereits vollzogenen dualen Modifizierungen.
- (d) $slack(v, \tau_v^A) = t_v^A - \Delta - \zeta_A$.
- (e) $t^A = \min\{t_v^A : v \text{ ist äußerer Knoten und es gibt kein } \bar{A} \in \mathcal{B} \text{ mit } A \cup \{v\} \subseteq \bar{A}\}$.

Man sieht leicht, dass (a), (c) und (e) gelten. Zum Zeitpunkt der Definition von τ_v^A (in (7) oder in $\text{UPDATE}(v)$) gelten (b) und (d), und danach wird $slack(v, u)$ um genau den Betrag verringert, um den sich $\Delta + \zeta_A$ erhöht (wegen (c)). Aus (a), (b), (d) und (e) folgt nun, dass ε_3 korrekt berechnet wird.

Nun nehmen wir an, dass $\varepsilon = \infty$, d. h. ε kann beliebig groß gewählt werden, ohne die duale Zulässigkeit zu zerstören. Da die duale Zielfunktion $\mathbb{1}z$ in ③ um mindestens ε erhöht wird, folgt, dass das duale LP (11.2) unbeschränkt ist. Damit ist das primale LP (11.1) nach Satz 3.27 unzulässig. \square

Hiermit folgt nun, dass der Algorithmus korrekt arbeitet:

Satz 11.10. *Terminiert der Algorithmus in ⑥, so bilden die Kanten $\{x, \mu(x)\}$ ein perfektes Matching minimalen Gewichtes in G .*

Beweis: Sei x der Inzidenzvektor von M (das aus den Kanten $\{x, \mu(x)\}$ bestehende Matching). Die Bedingungen des komplementären Schlupfes

$$\begin{aligned} x_e > 0 &\Rightarrow \sum_{A \in \mathcal{A}: e \in \delta(A)} z_A = c(e) \\ z_A > 0 &\Rightarrow \sum_{e \in \delta(A)} x_e = 1 \end{aligned}$$

sind erfüllt: Die erste Bedingung gilt, weil alle Matching-Kanten straff sind (Proposition 11.8(d)), und die zweite folgt aus Proposition 11.8(b).

Da zulässige primale und duale Lösungen vorliegen (Lemma 11.9), sind beide nach Korollar 3.23 optimal. Somit ist x optimal für das LP (11.1) und auch ganz-zahlig, womit bewiesen ist, dass M ein perfektes Matching minimalen Gewichtes ist. \square

Wir haben noch nicht gezeigt, dass der Algorithmus terminiert.

Satz 11.11. *Die Laufzeit des GEWICHTETEN MATCHING-ALGORITHMUS zwischen zwei aufeinander folgenden Augmentierungen ist $O(n^2)$. Die Gesamlaufzeit ist $O(n^3)$.*

Beweis: Nach Lemma 11.7 und Proposition 11.8(a) läuft das UPDATE Verfahren in linearer Zeit. Sowohl ② als auch ⑥ benötigen einmal pro Augmentierung $O(n^2)$ -Zeit. Der Zeitbedarf für ③ und ④ ist $O(n)$. Außerdem kann jeder der Schritte ⑤, ⑦ und ⑧ in $O(nk)$ -Zeit erledigt werden, wobei k die Anzahl der neuen äußeren Knoten ist. (In ⑦ ist die Anzahl der zu betrachtenden inklusionsmaximalen echten Teilmengen A' von A höchstens $2k + 1$: Jede zweite Unterblüte einer neuen Blüte muss eine innere gewesen sein.)

Da ein äußerer Knoten bis zu der nächsten Augmentierung ein solcher bleibt, ist die von ⑤, ⑦ und ⑧ zwischen zwei Augmentierungen benötigte Gesamtzeit $O(n^2)$. Ferner erzeugt jeder Aufruf von ⑤, ⑦ und ⑧ mindestens einen neuen äußeren Knoten. Da mindestens einer der Schritte ⑤, ⑥, ⑦, ⑧ bei jeder Iteration aufgerufen wird, ist die Anzahl der Iterationen zwischen zwei Augmentierungen $O(n)$. Damit haben wir bewiesen, dass die Laufzeit zwischen zwei Augmentierungen $O(n^2)$ ist. Da es nur $\frac{n}{2}$ Augmentierungen gibt, ist die Gesamlaufzeit $O(n^3)$. \square

Korollar 11.12. Das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM kann in $O(n^3)$ -Zeit gelöst werden.

Beweis: Diese Aussage folgt aus den Sätzen 11.10 und 11.11. \square

Die erste $O(n^3)$ -Implementierung von Edmonds' Algorithmus für das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM stammt von Gabow [1973] (siehe auch Gabow [1976] und Lawler [1976]). Die theoretisch beste Laufzeit, nämlich $O(mn + n^2 \log n)$, wurde auch von Gabow [1990] erreicht. Für planare Graphen kann ein perfektes Matching minimalen Gewichtes in $O\left(n^{\frac{3}{2}} \log n\right)$ -Zeit gefunden werden, wie von Lipton und Tarjan [1979, 1980] mittels eines „divide and conquer“ Ansatzes unter Verwendung der Eigenschaft, dass planare Graphen kleine „Separatoren“ haben, gezeigt worden ist. Für euklidische Instanzen (ein in der Ebene durch eine Punktmenge definierter vollständiger Graph mit durch euklidische Abstände gegebenen Kantengewichten) hat Varadarajan [1998] einen $O\left(n^{\frac{3}{2}} \log^5 n\right)$ Algorithmus gefunden.

Effiziente Implementierungen wurden von Mehlhorn und Schäfer [2000], Cook und Rohe [1999], und Kolmogorov [2009] beschrieben. Sie lösen Matching Probleme mit Millionen von Knoten optimal. Ein „primaler“ GEWICHTETER MATCHING-ALGORITHMUS, bei dem laufend ein perfektes Matching gespeichert ist und erst bei Terminierung eine zulässige duale Lösung vorliegt, ist von Cunningham und Marsh [1978] beschrieben worden. Ein Matching, dessen Gewicht mindestens $(1 - \epsilon)$ mal das maximale Gewicht eines Matchings ist, kann für jedes feste $\epsilon > 0$ in linearer Zeit bestimmt werden (Duan und Pettie [2014]).

11.4 Postoptimierung

In diesem Abschnitt werden wir ein Postoptimierungsresultat beweisen, das wir später in Abschnitt 12.2 benötigen. Zunächst fügen wir einer bereits gelösten Instanz zwei weitere Knoten hinzu:

Lemma 11.13. Sei (G, c) eine Instanz des MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEMS und seien $s, t \in V(G)$. Wir nehmen an, dass der GEWICHTETE MATCHING-ALGORITHMUS auf die Instanz $(G - \{s, t\}, c)$ angewendet worden ist. Dann kann ein perfektes Matching minimalen Gewichtes bezüglich (G, c) in $O(n^2)$ -Zeit bestimmt werden.

Beweis: Das Hinzufügen zweier Knoten erfordert die Initialisierung der Datenstrukturen. Insbesondere führen wir für jedes $v \in \{s, t\}$ Folgendes aus: Wir markieren v als äußerer Knoten, setzen $\mu(v) := v$, fügen $\{v\}$ zu \mathcal{B} hinzu, setzen $k_v := 0$, $\rho^0(v) := v$, $\varphi^0(v) := v$ und $\zeta_{\{v\}} := z_v := \min\left\{\frac{1}{2}c(\{s, t\}), \min\{c(\{v, w\}) - \zeta_{\{w\}} : w \in V(G) \setminus \{s, t\}\}\right\}$, wobei wir $c(e) := \infty$ für $e \notin E(G)$ gesetzt haben. Dann starten wir den GEWICHTETEN MATCHING-ALGORITHMUS mit ②. Nach Satz

[11.11](#) terminiert der Algorithmus nach $O(n^2)$ Schritten mit einer Augmentierung und liefert ein perfektes Matching minimalen Gewichtes in (G, c) . \square

Wir erhalten auch ein zweites Postoptimierungsresultat:

Lemma 11.14. (Weber [1981], Ball und Derigs [1983]) *Wir nehmen an, dass der GEWICHTETE MATCHING-ALGORITHMUS auf eine Instanz (G, c) angewendet worden ist. Sei $s \in V(G)$ und $c' : E(G) \rightarrow \mathbb{R}$ mit $c'(e) = c(e)$ für alle $e \notin \delta(s)$. Dann kann ein perfektes Matching minimalen Gewichtes bezüglich (G, c') in $O(n^2)$ -Zeit bestimmt werden.*

Beweis: Es gehe G' aus G durch das Hinzufügen zweier Knoten x, y , einer Kante $\{s, x\}$ und der Kanten $\{v, y\}$ für jede Kante $\{v, s\} \in E(G)$ hervor. Es sei $c(\{v, y\}) := c'(\{v, s\})$ für diese neuen Kanten. Das Gewicht der Kante $\{s, x\}$ kann beliebig gewählt werden. Mittels Lemma [11.13](#) bestimmen wir nun ein perfektes Matching minimalen Gewichtes in (G', c) . Entfernen wir die Kante $\{s, x\}$ und ersetzen die Matching-Kante $\{v, y\}$ durch $\{v, s\}$, so liegt ein perfektes Matching minimalen Gewichtes bezüglich (G, c') vor. \square

Dasselbe Resultat für einen „primalen“ GEWICHTETEN MATCHING-ALGORITHMUS kann man bei Cunningham und Marsh [1978] nachlesen.

11.5 Das Matching-Polytop

Die Korrektheit des GEWICHTETEN MATCHING-ALGORITHMUS liefert als Nebenresultat auch Edmonds' Charakterisierung des Perfekten-Matching-Polytops. Wiederum benutzen wir die Bezeichnung $\mathcal{A} := \{A \subseteq V(G) : |A| \text{ ungerade}\}$.

Satz 11.15. (Edmonds [1965]) *Sei G ein ungerichteter Graph. Das **Perfekte-Matching-Polytop** von G , d.h. die konvexe Hülle der Inzidenzvektoren aller perfekten Matchings in G , ist die Menge der Vektoren x , die das folgende lineare System erfüllen:*

$$\begin{aligned} x_e &\geq 0 & (e \in E(G)) \\ \sum_{e \in \delta(v)} x_e &= 1 & (v \in V(G)) \\ \sum_{e \in \delta(A)} x_e &\geq 1 & (A \in \mathcal{A}). \end{aligned}$$

Beweis: Nach Korollar [3.32](#) genügt es zu zeigen, dass alle Ecken des durch das obige System beschriebenen Polytops ganzzahlig sind. Nach Satz [5.14](#) gilt dies, falls das Minimierungsproblem eine ganzzahlige optimale Lösung für jede Gewichtsfunktion hat. Unser GEWICHTETER MATCHING-ALGORITHMUS findet aber solch eine Lösung für jede Gewichtsfunktion (siehe den Beweis von Satz [11.10](#)). \square

Einen zweiten Beweis werden wir in Abschnitt 12.3 antreffen (siehe die Bemerkung nach Satz 12.18).

Wir können auch eine Charakterisierung des **Matching-Polytops**, d. h. der konvexen Hülle der Inzidenzvektoren aller Matchings in einem ungerichteten Graphen G , angeben:

Satz 11.16. (Edmonds [1965]) *Sei G ein Graph. Das Matching-Polytop von G ist die Menge der Vektoren $x \in \mathbb{R}_+^{E(G)}$, die das folgende lineare Ungleichungssystem erfüllen:*

$$\sum_{e \in \delta(v)} x_e \leq 1 \quad \text{für alle } v \in V(G) \quad \text{und} \quad \sum_{e \in E(G[A])} x_e \leq \frac{|A| - 1}{2} \quad \text{für alle } A \in \mathcal{A}.$$

Beweis: Da der Inzidenzvektor eines Matchings offensichtlich diese Ungleichungen erfüllt, müssen wir nur die umgekehrte Richtung zeigen. Sei $x \in \mathbb{R}_+^{E(G)}$ ein Vektor mit $\sum_{e \in \delta(v)} x_e \leq 1$ für $v \in V(G)$ und $\sum_{e \in E(G[A])} x_e \leq \frac{|A|-1}{2}$ für $A \in \mathcal{A}$. Wir werden beweisen, dass x eine Konvexitätskombination von Inzidenzvektoren von Matchings ist.

Wir werden dieselbe Konstruktion wie im Beweis von Proposition 11.1 benutzen: Sei H der Graph mit $V(H) := \{(v, i) : v \in V(G), i \in \{1, 2\}\}$ und $E(H) := \{(v, i), (w, i) : \{v, w\} \in E(G), i \in \{1, 2\}\} \cup \{(v, 1), (v, 2) : v \in V(G)\}$. Somit besteht H aus zwei Kopien von G und es gibt eine Kante zwischen den beiden Kopien eines jeden Knotens. Sei $y_{\{(v,i),(w,i)\}} := x_e$ für jedes $e = \{v, w\} \in E(G)$ und $i \in \{1, 2\}$, und sei $y_{\{(v,1),(v,2)\}} := 1 - \sum_{e \in \delta_G(v)} x_e$ für jedes $v \in V(G)$. Wir behaupten, dass y im Perfekten-Matching-Polytop von H liegt. Betrachten wir den von $\{(v, 1) : v \in V(G)\}$ induzierten Teilgraphen, der zu G isomorph ist, so sehen wir, dass x eine Konvexitätskombination von Inzidenzvektoren von Matchings in G ist.

Offensichtlich gilt $y \in \mathbb{R}_+^{E(H)}$ und $\sum_{e \in \delta_H(v)} y_e = 1$ für alle $v \in V(H)$. Um zu zeigen, dass y im Perfekten-Matching-Polytop von H liegt, benutzen wir Satz 11.15. Sei also $X \subseteq V(H)$ mit $|X|$ ungerade. Wir beweisen, dass $\sum_{e \in \delta_H(X)} y_e \geq 1$. Seien $A := \{v \in V(G) : (v, 1) \in X, (v, 2) \notin X\}$, $B := \{v \in V(G) : (v, 1) \in X, (v, 2) \in X\}$ und $C := \{v \in V(G) : (v, 1) \notin X, (v, 2) \in X\}$. Da $|X|$ ungerade ist, hat entweder A oder C ungerade Kardinalität. O. B. d. A. können wir annehmen, dass $|A|$ ungerade ist. Setzen wir $A_i := \{(a, i) : a \in A\}$ und $B_i := \{(b, i) : b \in B\}$ für $i = 1, 2$ (siehe Abb. 11.9), so folgt:

$$\begin{aligned} \sum_{e \in \delta_H(X)} y_e &\geq \sum_{v \in A_1} \sum_{e \in \delta_H(v)} y_e - 2 \sum_{e \in E(H[A_1])} y_e - \sum_{e \in E_H(A_1, B_1)} y_e + \sum_{e \in E_H(B_2, A_2)} y_e \\ &= \sum_{v \in A_1} \sum_{e \in \delta_H(v)} y_e - 2 \sum_{e \in E(G[A])} x_e \\ &\geq |A_1| - (|A| - 1) = 1. \end{aligned}$$

□

In der Tat können wir das folgende stärkere Ergebnis beweisen:

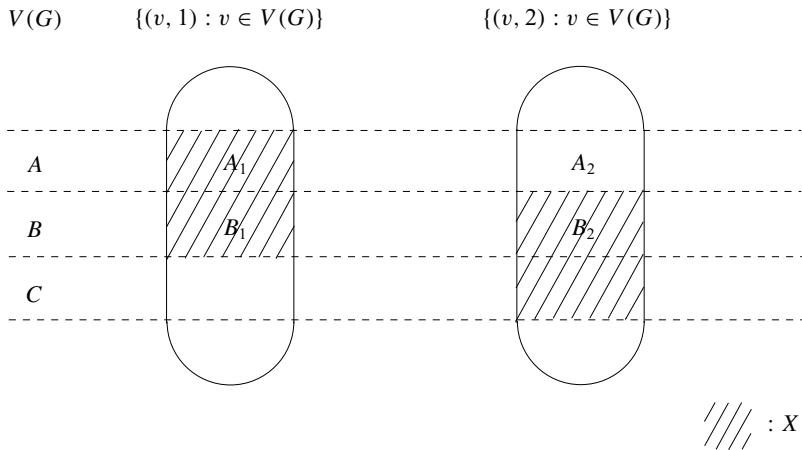


Abbildung 11.9.

Satz 11.17. (Cunningham und Marsh [1978]) Für jeden ungerichteten Graphen G ist das folgende lineare Ungleichungssystem TDI:

$$\begin{aligned} x_e &\geq 0 & (e \in E(G)) \\ \sum_{e \in \delta(v)} x_e &\leq 1 & (v \in V(G)) \\ \sum_{e \subseteq A} x_e &\leq \frac{|A|-1}{2} & (A \in \mathcal{A}, |A| > 1). \end{aligned}$$

Beweis: Für $c : E(G) \rightarrow \mathbb{Z}$ betrachten wir das LP $\max \sum_{e \in E(G)} c(e)x_e$ mit den obigen Nebenbedingungen. Das duale LP ist:

$$\begin{aligned} \min \quad & \sum_{v \in V(G)} y_v + \sum_{A \in \mathcal{A}, |A| > 1} \frac{|A|-1}{2} z_A \\ \text{bzgl.} \quad & \sum_{v \in e} y_v + \sum_{A \in \mathcal{A}, e \subseteq A} z_A \geq c(e) \quad (e \in E(G)) \\ & y_v \geq 0 \quad (v \in V(G)) \\ & z_A \geq 0 \quad (A \in \mathcal{A}, |A| > 1). \end{aligned}$$

Sei (G, c) das kleinste Gegenbeispiel, d.h. es gibt keine ganzzahlige optimale duale Lösung und $|V(G)| + |E(G)| + \sum_{e \in E(G)} |c(e)|$ ist minimal. Dann gilt $c(e) \geq 1$ für alle e (sonst könnten wir jede Kante mit nicht-positivem Gewicht entfernen), und G hat keine isolierten Knoten (sonst könnten wir sie entfernen).

Ferner behaupten wir, für jede optimale Lösung y, z gilt $y = 0$. Angenommen, es gäbe ein $v \in V(G)$ mit $y_v > 0$. Mittels des komplementären Schlupfes (Korollar 3.23) gilt dann $\sum_{e \in \delta(v)} x_e = 1$ für jede optimale primale Lösung x . Verringern wir nun $c(e)$ um eins für jedes $e \in \delta(v)$, so bekommen wir eine kleinere Instanz

(G, c') , für die der optimale Zielfunktionswert des LP um eins geringer ist (hier benutzen wir die primale Ganzzahligkeit, d. h. Satz 11.16). Da (G, c) das kleinste Gegenbeispiel ist, gibt es eine ganzzahlige optimale duale Lösung y', z' für (G, c') . Erhöhen wir y'_v um eins, so bekommen wir eine ganzzahlige optimale duale Lösung für (G, c) , ein Widerspruch.

Nun sei $y = 0$ und z eine optimale duale Lösung, für die

$$\sum_{A \in \mathcal{A}, |A| > 1} |A|^2 z_A \quad (11.4)$$

so groß wie möglich ist. Wir behaupten, dass $\mathcal{F} := \{A : z_A > 0\}$ laminar ist. Angenommen, das Gegenteil sei der Fall, d. h. es gäbe zwei Mengen $X, Y \in \mathcal{F}$ mit $X \setminus Y \neq \emptyset$, $Y \setminus X \neq \emptyset$ und $X \cap Y \neq \emptyset$. Sei $\epsilon := \min\{z_X, z_Y\} > 0$.

Ist $|X \cap Y|$ ungerade, so ist $|X \cup Y|$ auch ungerade. Setze $z'_X := z_X - \epsilon$, $z'_Y := z_Y - \epsilon$, $z'_{X \cap Y} := z_{X \cap Y} + \epsilon$ (außer wenn $|X \cap Y| = 1$), $z'_{X \cup Y} := z_{X \cup Y} + \epsilon$, und setze $z'_A := z_A$ für alle anderen Mengen A . Dann ist y, z' auch eine zulässige duale Lösung; ferner ist sie auch optimal. Dies ist aber ein Widerspruch, da (11.4) größer ist.

Ist $|X \cap Y|$ gerade, so sind $|X \setminus Y|$ und $|Y \setminus X|$ ungerade. Setze $z'_X := z_X - \epsilon$, $z'_Y := z_Y - \epsilon$, $z'_{X \setminus Y} := z_{X \setminus Y} + \epsilon$ (außer wenn $|X \setminus Y| = 1$), $z'_{Y \setminus X} := z_{Y \setminus X} + \epsilon$ (außer wenn $|Y \setminus X| = 1$), und setze $z'_A := z_A$ für alle anderen Mengen A . Setze $y'_v := \epsilon$ für $v \in X \cap Y$ und $y'_v := 0$ für $v \notin X \cap Y$. Dann ist (y', z') eine zulässige duale Lösung für die Folgendes gilt:

$$\begin{aligned} \sum_{v \in V(G)} y'_v + \sum_{A \in \mathcal{A}, |A| > 1} \frac{|A| - 1}{2} z'_A &= \epsilon |X \cap Y| + \sum_{A \in \mathcal{A}, |A| > 1} \frac{|A| - 1}{2} z_A \\ &\quad + \epsilon \left(\frac{|X \setminus Y| - 1}{2} + \frac{|Y \setminus X| - 1}{2} - \frac{|X| - 1}{2} - \frac{|Y| - 1}{2} \right) \\ &= \sum_{A \in \mathcal{A}, |A| > 1} \frac{|A| - 1}{2} z_A, \end{aligned}$$

also ist sie auch eine optimale Lösung. Dies widerspricht jedoch der Tatsache, dass für jede optimale duale Lösung (y', z') , $y' = 0$ gilt.

Nun sei $A \in \mathcal{F}$ mit $z_A \notin \mathbb{Z}$ und A inklusionsmaximal. Setze $\epsilon := z_A - \lfloor z_A \rfloor > 0$. Seien A_1, \dots, A_k die inklusionsmaximalen echten Teilmengen von A in \mathcal{F} ; diese sind paarweise disjunkt, da \mathcal{F} laminar ist. Setzen wir nun $z'_A := z_A - \epsilon$ und $z'_{A_i} := z_{A_i} + \epsilon$ für $i = 1, \dots, k$ (und $z'_D := z_D$ für alle anderen $D \in \mathcal{A}$), so erhalten wir eine weitere zulässige duale Lösung $y = 0, z'$ (da c ganzzahlig ist).

Damit folgt

$$\sum_{B \in \mathcal{A}, |B| > 1} \frac{|B| - 1}{2} z'_B < \sum_{B \in \mathcal{A}, |B| > 1} \frac{|B| - 1}{2} z_B,$$

im Widerspruch zur Optimalität der ursprünglichen dualen Lösung $y = 0, z$. \square

Dieser Beweis stammt von Schrijver [1983a]. Zu weiteren Beweisen siehe Lovász [1979] und Schrijver [1983b]. Letzterer benutzt nicht Satz 11.16. Ferner ergibt das Ersetzen von $\sum_{e \in \delta(v)} x_e \leq 1$ durch $\sum_{e \in \delta(v)} x_e = 1$ für $v \in V(G)$ in Satz 11.17 eine weitere Charakterisierung des Perfekten-Matching-Polytops, die auch TDI ist (nach Satz 5.19). Satz 11.15 kann leicht aus Letzterer abgeleitet werden; das lineare Ungleichungssystem von Satz 11.15 ist aber im Allgemeinen nicht TDI (K_4 ist ein Gegenbeispiel). Aus Satz 11.17 folgt auch die Berge-Tutte-Formel (Satz 10.14; siehe Aufgabe 16). Verallgemeinerungen werden wir in Abschnitt 12.1 besprechen.

Aufgaben

1. Man verwende Satz 11.4 um eine gewichtete Version des Satzes von König (Satz 10.2) zu beweisen.
(Egerváry [1931])
2. Man beschreibe die konvexe Hülle der Inzidenzvektoren aller
 - (a) Knotenüberdeckungen,
 - (b) stabilen Mengen,
 - (c) Kantenüberdeckungen,
 in einem bipartiten Graphen G . Man zeige, wir man Satz 10.2 und die Aussage von Aufgabe 2(c), Kapitel 10, daraus ableiten kann.
Hinweis: Man benutze Satz 5.26 und Korollar 5.22.
3. Man gebe einen direkten Beweis des Birkhoff-von-Neumann-Satzes (Satz 11.5) an.
4. Sei G ein Graph und P das gebrochene Perfekte-Matching-Polytop von G . Man beweise, dass die Knoten von P genau diejenigen Vektoren x mit

$$x_e = \begin{cases} \frac{1}{2} & \text{für } e \in E(C_1) \cup \dots \cup E(C_k) \\ 1 & \text{für } e \in M \\ 0 & \text{sonst} \end{cases}$$

sind, wobei C_1, \dots, C_k paarweise knotendisjunkte ungerade Kreise sind und M ein perfektes Matching in $G - (V(C_1) \cup \dots \cup V(C_k))$ ist.
(Balinski [1972]; siehe Lovász [1979])

5. Sei G ein bipartiter Graph mit Bipartition $V = A \cup B$ und $A = \{a_1, \dots, a_p\}$, $B = \{b_1, \dots, b_q\}$. Seien $c : E(G) \rightarrow \mathbb{R}$ die Kantengewichte. Gesucht wird das die Reihenfolge erhaltende Matching M maximalen Gewichtes, d. h. für je zwei Kanten $\{a_i, b_j\}, \{a_{i'}, b_{j'}\} \in M$ mit $i < i'$ gelte auch $j < j'$. Man löse dieses Problem mit einem $O(n^3)$ -Algorithmus.
Hinweis: Man benutze dynamische Optimierung.
6. Man beweise, dass zu jedem Zeitpunkt des GEWICHTETEN MATCHING-ALGORITHMUS $|\mathcal{B}| \leq \frac{3}{2}n$ gilt.

7. Sei G ein Graph mit nichtnegativen Gewichten $c : E(G) \rightarrow \mathbb{R}_+$. Sei M das Matching auf einer beliebigen Zwischenstufe des GEWICHTETEN MATCHING-ALGORITHMUS. Sei X die von M überdeckte Knotenmenge. Man zeige, dass jedes X überdeckende Matching mindestens so teuer wie M ist.
(Ball und Derigs [1983])
8. Ein Graph mit ganzzahligen Gewichten auf den Kanten hat die „gerade-Kreiseigenschaft“, falls das Gesamtgewicht eines jeden Kreises gerade ist. Man zeige, dass bei der Anwendung des GEWICHTETEN MATCHING-ALGORITHMUS auf einen Graphen mit der „gerade-Kreiseigenschaft“ diese Eigenschaft (bezüglich der Schläpfe) erhalten bleibt, und auch laufend eine gespeicherte duale Lösung, die ganzzahlig ist, bereit steht. Man folgere hieraus, dass es für jeden Graphen eine optimale duale Lösung z gibt, die halbganzzahlig ist (d. h. $2z$ ist ganzzahlig).
9. Für bipartite Graphen wird der GEWICHTETE MATCHING-ALGORITHMUS viel einfacher. Man zeige, welche Teile weiterhin notwendig sind und welche nicht.

Bemerkung: Man gelangt zu der sogenannten Ungarischen Methode für das ZUORDNUNGSPROBLEM (Kuhn [1955]). Dieses Verfahren kann auch als eine äquivalente Beschreibung des im Beweis von Satz 11.2 vorgeschlagenen Verfahrens betrachtet werden.

10. Angenommen G ist ein vollständiger Graph und $c : E(G) \rightarrow \mathbb{R}_+$ erfüllt die Dreiecksungleichung, d. h. $c(\{x, z\}) \leq c(\{x, y\}) + c(\{y, z\})$ für alle $x, y, z \in V(G)$. Man zeige, dass es dann eine optimale Lösung z für (11.2) mit $z \geq 0$ gibt.
- * 11. Man zeige, dass nach jedem Augmentierungsschritt des GEWICHTETEN MATCHING-ALGORITHMUS das aktuell betrachtete Matching M unter allen Matchings der Kardinalität $|M|$ minimales Gewicht hat.

Hinweis: Man erweitere die duale Lösung auf eine Instanz mit $|V(G)| - 2|M|$ zusätzlichen Hilfsknoten.

Bemerkung: Dies liefert eine alternative Lösung für das MAXIMUM-WEIGHT-MATCHING-PROBLEM, die effizienter sein könnte als die im Beweis von Proposition 11.1.

(Lawler [1976], Walter und Kaibel [2017])

12. Wie kann das Bottleneck-Matching-Problem (man bestimme ein perfektes Matching M , welches $\max\{c(e) : e \in M\}$ minimiert) in $O(n^3)$ -Zeit gelöst werden?
13. Man gebe ein polynomielles Verfahren zur Lösung des MINIMUM-WEIGHT-EDGE-COVER-PROBLEMS an: Man bestimme eine Kantenüberdeckung minimalen Gewichtes für einen ungerichteten Graphen G mit Gewichten $c : E(G) \rightarrow \mathbb{R}$.
14. Gegeben sei ein ungerichteter Graph G mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ und zwei Knoten s und t . Gesucht wird ein kürzester $s-t$ -Weg mit einer geraden (oder mit einer ungeraden) Anzahl von Kanten. Man führe dies zurück auf ein MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM.

Hinweis: Man nehme zwei Kopien von G , verbinde jeden Knoten mit seiner

Kopie mittels einer Kante mit Gewicht Null und entferne dann s und t (oder s und die Kopie von t).

(Grötschel und Pulleyblank [1981])

15. Sei G ein k -regulärer und $(k-1)$ -fach kantenzusammenhängender Graph mit einer geraden Anzahl von Knoten und den Gewichten $c : E(G) \rightarrow \mathbb{R}_+$. Man beweise, dass es ein perfektes Matching M in G mit $c(M) \geq \frac{1}{k}c(E(G))$ gibt.

Hinweis: Man zeige, dass $\frac{1}{k}\mathbf{1}$ im Perfekten-Matching-Polytop liegt.
(Naddef und Pulleyblank [1981])

- * 16. Man folgere aus Satz 11.17:

(a) die Berge-Tutte-Formel (Satz 10.14);

(b) Satz 11.15;

(c) die Existenz einer optimalen halbganzzahligen dualen Lösung des dualen LP (11.2) (siehe Aufgabe 8).

Hinweis: Benutze Satz 5.19.

17. Das gebrochene Perfekte-Matching-Polytop Q von G ist gleich dem Perfekten-Matching-Polytop wenn G bipartit ist (Satz 11.4). Man betrachte zunächst die Gomory-Chvátal-Stützung Q' von Q (Definition 5.30). Man beweise, dass Q' immer gleich dem Perfekten-Matching-Polytop ist.

18. Man zeige, wie man zu einem gegebenen Graphen G , der Kantengraph eines Graphen H ist, und Gewichten $c : V(G) \rightarrow \mathbb{R}_+$ eine maximal gewichtete stabile Menge in $O(n^3)$ -Zeit bestimmen kann, wobei $n = |V(G)|$.

Bemerkung: Faenza, Oriolo und Stauffer [2014] geben einen Algorithmus mit $O(mn + n^2 \log n)$ -Zeit für *klauenfreie* Graphen an, also Graphen, die $K_{1,3}$ nicht als induzierten Teilgraphen enthalten. Beachte, dass Kantengraphen klauenfrei sind, also ist dies eine Verallgemeinerung des gewichteten Matchings.

Literatur

Allgemeine Literatur:

Gerards, A.M.H. [1995]: Matching. In: Handbooks in Operations Research and Management Science; Volume 7: Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, Hrsg.), Elsevier, Amsterdam 1995, pp. 135–224

Lawler, E.L. [1976]: Combinatorial Optimization; Networks and Matroids. Holt, Rinehart and Winston, New York 1976, Kapitel 5 und 6

Papadimitriou, C.H., und Steiglitz, K. [1982]: Combinatorial Optimization; Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, Kapitel 11

Pulleyblank, W.R. [1995]: Matchings and extensions. In: Handbook of Combinatorics; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, Hrsg.), Elsevier, Amsterdam 1995

Zitierte Literatur:

Balinski, M.L. [1972]: Establishing the matching polytope. Journal of Combinatorial Theory 13 (1972), 1–13

Ball, M.O., und Derigs, U. [1983]: An analysis of alternative strategies for implementing matching algorithms. Networks 13 (1983), 517–549

- Birkhoff, G. [1946]: Tres observaciones sobre el algebra lineal. Revista Universidad Nacional de Tucumán, Series A 5 (1946), 147–151
- Burkard, R., Dell'Amico, M., und Martello, S. [2009]: Assignment Problems. SIAM, Philadelphia 2009
- Cook, W.J., und Rohe, A. [1999]: Computing minimum-weight perfect matchings. INFORMS Journal of Computing 11 (1999), 138–148
- Cunningham, W.H., und Marsh, A.B. [1978]: A primal algorithm for optimum matching. Mathematical Programming Study 8 (1978), 50–72
- Duan, R., und Pettie, S. [2014]: Linear-time approximation for maximum weight matching. Journal of the ACM 61 (2014), 1–23
- Edmonds, J. [1965]: Maximum matching and a polyhedron with (0,1) vertices. Journal of Research of the National Bureau of Standards B 69 (1965), 125–130
- Egerváry, E. [1931]: Matrixok kombinatorikus tulajdonságairol. Matematikai és Fizikai Lapok 38 (1931), 16–28 [auf Ungarisch]
- Faenza, Y., Oriolo, G., und Stauffer, G. [2014]: Solving the weighted stable set problem in claw-free graphs via decomposition. Journal of the ACM 61 (2014), Article 20
- Gabow, H.N. [1973]: Implementation of algorithms for maximum matching on non-bipartite graphs. Ph.D. Thesis, Stanford University, Dept. of Computer Science, 1973
- Gabow, H.N. [1976]: An efficient implementation of Edmonds' algorithm for maximum matching on graphs. Journal of the ACM 23 (1976), 221–234
- Gabow, H.N. [1990]: Data structures for weighted matching and nearest common ancestors with linking. Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms (1990), 434–443. Vollständige Version bei arXiv:1611.07541
- Grötschel, M., und Pulleyblank, W.R. [1981]: Weakly bipartite graphs and the max-cut problem. Operations Research Letters 1 (1981), 23–27
- Kolmogorov, V. [2009]: Blossom V: a new implementation of a minimum cost perfect matching algorithm. Mathematical Programming Computation 1 (2009), 43–67
- Kuhn, H.W. [1955]: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2 (1955), 83–97
- Lipton, R.J., und Tarjan, R.E. [1979]: A separator theorem for planar graphs. SIAM Journal on Applied Mathematics 36 (1979), 177–189
- Lipton, R.J., und Tarjan, R.E. [1980]: Applications of a planar separator theorem. SIAM Journal on Computing 9 (1980), 615–627
- Lovász, L. [1979]: Graph theory and integer programming. In: Discrete Optimization I; Annals of Discrete Mathematics 4 (P.L. Hammer, E.L. Johnson, B.H. Korte, Hrsg.), North-Holland, Amsterdam 1979, pp. 141–158
- Mehlhorn, K., und Schäfer, G. [2000]: Implementation of $O(nm \log n)$ weighted matchings in general graphs: the power of data structures. In: Algorithm Engineering; WAE-2000; LNCS 1982 (S. Näher, D. Wagner, Hrsg.), pp. 23–38; also electronically in The ACM Journal of Experimental Algorithms 7 (2002)
- Monge, G. [1784]: Mémoire sur la théorie des déblais et des remblais. Histoire de l'Académie Royale des Sciences 2 (1784), 666–704
- Munkres, J. [1957]: Algorithms for the assignment und transportation problems. Journal of the Society for Industrial and Applied Mathematics 5 (1957), 32–38
- Naddef, D., und Pulleyblank, W.R. [1981]: Matchings in regular graphs. Discrete Mathematics 34 (1981), 283–291
- von Neumann, J. [1953]: A certain zero-sum two-person game equivalent to the optimal assignment problem. In: Contributions to the Theory of Games II; Ann. of Math. Stud. 28 (H.W. Kuhn, Hrsg.), Princeton University Press, Princeton 1953, pp. 5–12

- Schrijver, A. [1983a]: Short proofs on the matching polyhedron. *Journal of Combinatorial Theory B* 34 (1983), 104–108
- Schrijver, A. [1983b]: Min-max results in combinatorial optimization. In: *Mathematical Programming; The State of the Art – Bonn 1982* (A. Bachem, M. Grötschel, B. Korte, Hrsg.), Springer, Berlin 1983, pp. 439–500
- Varadarajan, K.R. [1998]: A divide-and-conquer algorithm for min-cost perfect matching in the plane. *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science* (1998), 320–329
- Walter, M., und Kaibel, V. [2017]: A note on matchings constructed during Edmonds’ weighted perfect matching algorithm. arXiv:1703.09505
- Weber, G.M. [1981]: Sensitivity analysis of optimal matchings. *Networks* 11 (1981), 41–56



12 b -Matchings und T -Joins

In diesem Kapitel werden wir zwei weitere kombinatorische Optimierungsprobleme einführen, nämlich das MAXIMUM-WEIGHT- b -MATCHING-PROBLEM (in Abschnitt 12.1) und das MINIMUM-WEIGHT- T -JOIN-PROBLEM (in Abschnitt 12.2). Beide können als Verallgemeinerungen des MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEMS aufgefasst werden und beide enthalten auch andere wichtige Probleme. Andererseits können beide auf das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM zurückgeführt werden. Ferner gibt es für beide polynomiale kombinatorische Algorithmen und auch polyedrische Beschreibungen. Da das SEPARATIONS-PROBLEM in beiden Fällen in polynomieller Zeit gelöst werden kann, erhalten wir einen weiteren polynomiellen Algorithmus für diese verallgemeinerten Matching-Probleme (mittels der ELLIPSOIDMETHODE, siehe Abschnitt 4.6). In der Tat kann das SEPARATIONS-PROBLEM in beiden Fällen auf die Bestimmung eines T -Schnittes minimaler Kapazität zurückgeführt werden; siehe Abschnitte 12.3 und 12.4. Das Problem der Bestimmung eines Schnittes $\delta(X)$ minimaler Kapazität mit $|X \cap T|$ ungerade für eine bestimmte Knotenmenge T kann mittels Netzwerkfluss-Verfahren gelöst werden.

12.1 b -Matchings

Definition 12.1. Sei G ein ungerichteter Graph mit ganzzahligen Kantenkapazitäten $u : E(G) \rightarrow \mathbb{N} \cup \{\infty\}$ und Zahlen $b : V(G) \rightarrow \mathbb{N}$. Dann ist ein **b -Matching** in (G, u) eine Funktion $f : E(G) \rightarrow \mathbb{Z}_+$ mit $f(e) \leq u(e)$ für alle $e \in E(G)$ und $\sum_{e \in \delta(v)} f(e) \leq b(v)$ für alle $v \in V(G)$. Ist $u \equiv 1$, so liegt ein **einfaches b -Matching** in G vor. Ein b -Matching f heißt **perfekt**, falls $\sum_{e \in \delta(v)} f(e) = b(v)$ für alle $v \in V(G)$.

Gilt $b \equiv 1$, so spielen die Kapazitäten keine Rolle und wir haben ein ganz normales Matching. Ein einfaches perfektes b -Matching wird auch ein b -Faktor genannt. Man kann es als eine Knotenmenge betrachten. In Kapitel 21 werden wir perfekte einfache 2-Matchings in G betrachten, d. h. Knotenmengen mit der Eigenschaft: Jeder Knoten von G ist mit genau zwei der Kanten (aus der betrachteten Knotenmenge) inzident.

MAXIMUM-WEIGHT-*b*-MATCHING-PROBLEM

Instanz: Ein Graph G , Kapazitäten $u : E(G) \rightarrow \mathbb{N} \cup \{\infty\}$, Gewichte $c : E(G) \rightarrow \mathbb{R}$ und Zahlen $b : V(G) \rightarrow \mathbb{N}$.

Aufgabe: Bestimme ein *b*-Matching f in (G, u) mit maximalem Gewicht $\sum_{e \in E(G)} c(e)f(e)$.

Man kann Edmonds' GEWICHTETEN MATCHING-ALGORITHMUS erweitern, so dass er dieses Problem löst (Marsh [1979]). Wir werden hierauf jedoch nicht näher eingehen, sondern stattdessen eine polyedrische Beschreibung angeben und zeigen, dass das SEPARATIONS-PROBLEM in polynomieller Zeit gelöst werden kann. Mittels der ELLIPSOIDMETHODE (siehe Korollar 3.33) erhalten wir damit einen polynomiellen Algorithmus.

Das ***b*-Matching-Polytop** von (G, u) ist die konvexe Hülle aller *b*-Matchings in (G, u) . Zunächst betrachten wir den Fall ohne Kapazitäten ($u \equiv \infty$):

Satz 12.2. (Edmonds [1965]) Sei G ein ungerichteter Graph und $b : V(G) \rightarrow \mathbb{N}$. Das *b*-Matching-Polytop von (G, ∞) ist die Menge derjenigen Vektoren $x \in \mathbb{R}_+^{E(G)}$, die das folgende Ungleichungssystem erfüllen:

$$\begin{aligned} \sum_{e \in \delta(v)} x_e &\leq b(v) & (v \in V(G)); \\ \sum_{e \in E(G[X])} x_e &\leq \left\lfloor \frac{1}{2} \sum_{v \in X} b(v) \right\rfloor & (X \subseteq V(G)). \end{aligned}$$

Beweis: Da offensichtlich jedes *b*-Matching diese Ungleichungen erfüllt, brauchen wir nur die umgekehrte Richtung zu zeigen. Sei also $x \in \mathbb{R}_+^{E(G)}$ mit $\sum_{e \in \delta(v)} x_e \leq b(v)$ für alle $v \in V(G)$ und $\sum_{e \in E(G[X])} x_e \leq \lfloor \frac{1}{2} \sum_{v \in X} b(v) \rfloor$ für alle $X \subseteq V(G)$. Wir werden zeigen, dass x eine Konvexitätskombination von *b*-Matchings ist.

Dazu definieren wir einen neuen Graphen H , indem wir $b(v)$ Kopien von jedem Knoten v nehmen: Wir definieren $X_v := \{(v, i) : i \in \{1, \dots, b(v)\}\}$ für $v \in V(G)$, $V(H) := \bigcup_{v \in V(G)} X_v$ und $E(H) := \{\{v', w'\} : \{v, w\} \in E(G), v' \in X_v, w' \in X_w\}$. Sei $y_e := \frac{1}{b(v)b(w)}x_{\{v,w\}}$ für jede Kante $e = \{v', w'\} \in E(H)$, $v' \in X_v, w' \in X_w$. Wir behaupten nun, dass y eine Konvexitätskombination von Inzidenzvektoren von Matchings in H ist. Durch anschließende Kontraktion der Mengen X_v ($v \in V(G)$) in H bekommen wir dann wieder G und x , womit folgt, dass x eine Konvexitätskombination von *b*-Matchings in G ist.

Zum Beweis der Behauptung: Wir verwenden Satz 11.16 um zu beweisen, dass y im Matching-Polytop von H liegt. Die Ungleichung $\sum_{e \in \delta(v)} y_e \leq 1$ gilt offensichtlich für jedes $v \in V(H)$. Sei $C \subseteq V(H)$ mit $|C|$ ungerade. Wir werden zeigen, dass $\sum_{e \in E(H[C])} y_e \leq \frac{1}{2}(|C| - 1)$ gilt.

Gilt $X_v \subseteq C$ oder $X_v \cap C = \emptyset$ für jedes $v \in V(G)$, so folgt dies direkt aus den von x erfüllten Ungleichungen. Andernfalls, seien $a, b \in X_v$, $a \in C, b \notin C$. Dann haben wir

$$\begin{aligned}
2 \sum_{e \in E(H[C])} y_e &= \sum_{c \in C \setminus \{a\}} \sum_{e \in E(\{c\}, C \setminus \{c\})} y_e + \sum_{e \in E(\{a\}, C \setminus \{a\})} y_e \\
&\leq \sum_{c \in C \setminus \{a\}} \sum_{e \in \delta(c) \setminus \{\{c,b\}\}} y_e + \sum_{e \in E(\{a\}, C \setminus \{a\})} y_e \\
&= \sum_{c \in C \setminus \{a\}} \sum_{e \in \delta(c)} y_e - \sum_{e \in E(\{b\}, C \setminus \{a\})} y_e + \sum_{e \in E(\{a\}, C \setminus \{a\})} y_e \\
&= \sum_{c \in C \setminus \{a\}} \sum_{e \in \delta(c)} y_e \\
&\leq |C| - 1.
\end{aligned}$$

□

Beachte, dass diese Konstruktion zwar einen Algorithmus liefert, dieser aber im Allgemeinen exponentielle Laufzeit hat. In dem Spezialfall $\sum_{v \in V(G)} b(v) = O(n)$ können wir jedoch das MAXIMUM-WEIGHT-*b*-MATCHING-PROBLEM ohne Kapazitäten in $O(n^3)$ -Zeit lösen (mittels des GEWICHTETEN MATCHING-ALGORITHMUS; siehe Korollar 11.12). Pulleyblank [1973, 1980] hat die Facetten dieses Polytops beschrieben und gezeigt, dass das in Satz 12.2 gegebene lineare Gleichungssystem TDI ist. Die folgende Verallgemeinerung erlaubt endliche Kapazitäten:

Satz 12.3. (Edmonds und Johnson [1970]) *Sei G ein ungerichteter Graph, $u : E(G) \rightarrow \mathbb{N} \cup \{\infty\}$ und $b : V(G) \rightarrow \mathbb{N}$. Das *b*-Matching-Polytop von (G, u) ist die Menge derjenigen Vektoren $x \in \mathbb{R}_+^{E(G)}$, die das folgende Ungleichungssystem erfüllen:*

$$\begin{aligned}
x_e &\leq u(e) && (e \in E(G)); \\
\sum_{e \in \delta(v)} x_e &\leq b(v) && (v \in V(G)); \\
\sum_{e \in E(G[X])} x_e + \sum_{e \in F} x_e &\leq \left\lfloor \frac{1}{2} \left(\sum_{v \in X} b(v) + \sum_{e \in F} u(e) \right) \right\rfloor && (X \subseteq V(G), \\
&&& F \subseteq \delta(X)).
\end{aligned}$$

Beweis: Beachte zunächst, dass jedes *b*-Matching x die Ungleichungen erfüllt; dies ist klar bis auf die letzte Ungleichung. Es erfüllt aber jeder Vektor $x \in \mathbb{R}_+^{E(G)}$ mit $x_e \leq u(e)$ ($e \in E(G)$) und $\sum_{e \in \delta(v)} x_e \leq b(v)$ ($v \in V(G)$) die folgende Ungleichung:

$$\begin{aligned}
\sum_{e \in E(G[X])} x_e + \sum_{e \in F} x_e &= \frac{1}{2} \left(\sum_{v \in X} \sum_{e \in \delta(v)} x_e + \sum_{e \in F} x_e - \sum_{e \in \delta(X) \setminus F} x_e \right) \\
&\leq \frac{1}{2} \left(\sum_{v \in X} b(v) + \sum_{e \in F} u(e) \right).
\end{aligned}$$

Ist x ganzzahlig, so ist die linke Seite eine ganze Zahl. Somit können wir die rechte Seite abrunden.

Nun sei $x \in \mathbb{R}_+^{E(G)}$ ein Vektor mit $x_e \leq u(e)$ für alle $e \in E(G)$, $\sum_{e \in \delta(v)} x_e \leq b(v)$ für alle $v \in V(G)$ und

$$\sum_{e \in E(G[X])} x_e + \sum_{e \in F} x_e \leq \left\lfloor \frac{1}{2} \left(\sum_{v \in X} b(v) + \sum_{e \in F} u(e) \right) \right\rfloor$$

für alle $X \subseteq V(G)$ und $F \subseteq \delta(X)$. Wir werden zeigen, dass x eine Konvexitätskombination von b -Matchings in (G, u) ist.

Sei H der aus G durch Unterteilung einer jeden Kante $e = \{v, w\}$ mit $u(e) \neq \infty$ mittels zweier neuer Knoten $(e, v), (e, w)$ hervorgehende Graph. (Anstatt e enthält H nun die Kanten $\{v, (e, v)\}, \{(e, v), (e, w)\}$ und $\{(e, w), w\}$.) Setze $b((e, v)) := b((e, w)) := u(e)$ für die neuen Knoten.

Für jede unterteilte Kante $e = \{v, w\}$ setzen wir $y_{\{v, (e, v)\}} := y_{\{(e, w), w\}} := x_e$ und $y_{\{(e, v), (e, w)\}} := u(e) - x_e$. Für jede ursprüngliche Kante e mit $u(e) = \infty$ setzen wir $y_e := x_e$. Wir behaupten nun, dass y im b -Matching-Polytop P von (H, ∞) liegt.

Dazu benutzen wir Satz 12.2. Offensichtlich gilt $y \in \mathbb{R}_+^{E(H)}$ und $\sum_{e \in \delta(v)} y_e \leq b(v)$ für alle $v \in V(H)$. Angenommen, es gäbe eine Menge $A \subseteq V(H)$ mit

$$\sum_{e \in E(H[A])} y_e > \left\lfloor \frac{1}{2} \sum_{a \in A} b(a) \right\rfloor. \quad (12.1)$$

Sei $B := A \cap V(G)$. Für jedes $e = \{v, w\} \in E(G[B])$ können wir $(e, v), (e, w) \in A$ annehmen, da sonst das Hinzufügen von (e, v) und (e, w) die Ungleichung (12.1) nicht verletzen würde. Andererseits können wir annehmen: Aus $(e, v) \in A$ folgt $v \in A$, denn ist $(e, v), (e, w) \in A$ aber $v \notin A$, so können wir (e, v) und (e, w) aus A entfernen ohne (12.1) zu verletzen. Ist $(e, v) \in A$ aber $v, (e, w) \notin A$, so können wir einfach (e, v) aus A entfernen. Abbildung 12.1 zeigt die weiteren möglichen Arten von Kanten.

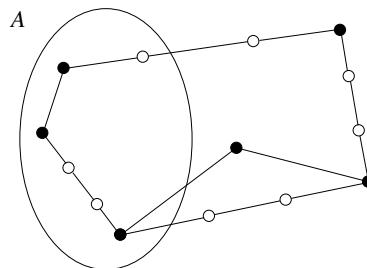


Abbildung 12.1.

Sei $F := \{e = \{v, w\} \in E(G) : |A \cap \{(e, v), (e, w)\}| = 1\}$. Dann folgt

$$\begin{aligned}
\sum_{e \in E(G[B])} x_e + \sum_{e \in F} x_e &= \sum_{e \in E(H[A])} y_e - \sum_{\substack{e \in E(G[B]), \\ u(e) < \infty}} u(e) \\
&> \left\lfloor \frac{1}{2} \sum_{a \in A} b(a) \right\rfloor - \sum_{\substack{e \in E(G[B]), \\ u(e) < \infty}} u(e) \\
&= \left\lfloor \frac{1}{2} \left(\sum_{v \in B} b(v) + \sum_{e \in F} u(e) \right) \right\rfloor,
\end{aligned}$$

im Widerspruch zu unserer Annahme. Somit ist $y \in P$, und y liegt in der Tat auf der Seitenfläche

$$\left\{ z \in P : \sum_{e \in \delta(v)} z_e = b(v) \text{ für alle } v \in V(H) \setminus V(G) \right\}$$

von P . Da die Knoten dieser Seitenfläche auch Knoten von P sind, ist y eine Konvexitätskombination von b -Matchings f_1, \dots, f_m in (H, ∞) , die alle die Gleichung $\sum_{e \in \delta(v)} f_i(e) = b(v)$ für jedes $v \in V(H) \setminus V(G)$ erfüllen. Daraus folgt $f_i(\{v, (e, v)\}) = f_i(\{(e, w), w\}) \leq u(e)$ für jede unterteilte Kante $e = \{v, w\} \in E(G)$. Kehren wir nun von H nach G zurück, so sehen wir, dass x eine Konvexitätskombination von b -Matchings in (G, u) ist. \square

Die Konstruktionen in den Beweisen von Satz 12.2 und Satz 12.3 stammen beide von Tutte [1954]. Sie können auch dazu dienen, eine Verallgemeinerung des Satzes von Tutte (Satz 10.13) zu beweisen (Aufgabe 4):

Satz 12.4. (Tutte [1952]) *Sei G ein Graph, $u : E(G) \rightarrow \mathbb{N} \cup \{\infty\}$ und $b : V(G) \rightarrow \mathbb{N}$. Dann hat (G, u) ein perfektes b -Matching genau dann, wenn für je zwei disjunkte Teilmengen $X, Y \subseteq V(G)$ die Anzahl derjenigen Zusammenhangskomponenten C in $G - X - Y$, für die $\sum_{c \in V(C)} b(c) + \sum_{e \in E_G(V(C), Y)} u(e)$ ungerade ist, die Zahl*

$$\sum_{v \in X} b(v) + \sum_{y \in Y} \left(\sum_{e \in \delta(y)} u(e) - b(y) \right) - \sum_{e \in E_G(X, Y)} u(e)$$

nicht überschreitet.

12.2 T-Joins mit minimalem Gewicht

Betrachte das folgende Problem: Ein Postbote muss in einem bestimmten Stadtteil Post austragen. Dazu verlässt er am Anfang das Postamt, durchläuft jede Straße mindestens einmal und kehrt schließlich zum Postamt zurück. Das Problem besteht

darin, eine Tour minimaler Länge zu bestimmen, und ist als das CHINESISCHE POSTBOTEN-PROBLEM bekannt (Guan [1962]).

Natürlich kann man den Straßenplan als Graphen darstellen, den wir als zusammenhängend annehmen werden (denn sonst müsste der Postbote Straßen benutzen, die nicht zu seinem Stadtteil gehören, was das Problem *NP*-schwer machen würde; siehe Aufgabe 17(d), Kapitel 15). Nach dem Satz von Euler (Satz 2.24) wissen wir, dass es genau dann eine Postbotentour gibt, die jede Kante genau einmal durchläuft (d.h. ein eulerscher Spaziergang ist), wenn der Grad eines jeden Knotens gerade ist.

Ist der Graph nicht eulersch, so müssen wir einige Kanten mehrfach durchlaufen. Mit Kenntnis des Satzes von Euler können wir das CHINESISCHE POSTBOTEN-PROBLEM folgendermaßen formulieren: Für einen gegebenen Graphen G mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ bestimme man eine Funktion $n : E(G) \rightarrow \mathbb{N}$, so dass der aus G durch das Ersetzen einer jeden Kante $e \in E(G)$ mit $n(e)$ Kopien von e hervorgehende Graph G' eulersch ist und $\sum_{e \in E(G)} n(e)c(e)$ minimal ist.

Dies trifft alles sowohl für den gerichteten als auch für den ungerichteten Fall zu. Der gerichtete Fall kann mittels Netzwerkfluss-Verfahren gelöst werden (Aufgabe 9, Kapitel 9). Somit werden wir fortan nur ungerichtete Graphen betrachten. Dazu verwenden wir den GEWICHTETEN MATCHING-ALGORITHMUS.

Natürlich ist es nicht sinnvoll, eine Kante e mehr als zweimal zu durchlaufen, denn dann könnten wir den Betrag 2 von einem geeigneten $n(e)$ abziehen und hätten somit eine Lösung, die nicht schlechter sein kann. Also besteht das Problem in der Bestimmung einer Kantenmenge $J \subseteq E(G)$ minimalen Gewichtes, so dass der Graph $(V(G), E(G) \cup J)$ (der durch Duplizierung der Kanten in J entsteht) eulersch ist. In diesem Abschnitt lösen wir eine Verallgemeinerung dieses Problems.

Definition 12.5. Gegeben sei ein ungerichteter Graph G und eine Knotenmenge $T \subseteq V(G)$. Eine Kantenmenge $J \subseteq E(G)$ heißt ein **T-Join**, falls $|J \cap \delta(x)|$ genau dann ungerade ist, wenn $x \in T$.

MINIMUM-WEIGHT-T-JOIN-PROBLEM

Instanz: Ein ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$ und eine Knotenmenge $T \subseteq V(G)$.

Aufgabe: Bestimme einen T -Join mit minimalem Gewicht in G oder entscheide, dass es keinen solchen gibt.

Das MINIMUM-WEIGHT-T-JOIN-PROBLEM verallgemeinert verschiedene kombinatorische Optimierungsprobleme:

- Sind die Gewichte c nichtnegativ und ist T die Menge der Knoten in G mit ungeradem Grad, so haben wir das UNGERICHTETE CHINESISCHE POSTBOTEN-PROBLEM (falls G zusammenhängend ist).
- Ist $T = \emptyset$, so sind die T -Joins genau die eulerschen Teilgraphen. Also ist die leere Menge genau dann ein \emptyset -Join mit minimalem Gewicht, wenn c eine konservative Gewichtsfunktion ist.

- Ist $|T| = 2$, etwa $T = \{s, t\}$, so ist jeder T -Join die Vereinigung eines $s-t$ -Weges mit eventuellen Kreisen. Ist c konservativ, so ist das MINIMUM-WEIGHT- T -JOIN-PROBLEM somit äquivalent zum KÜRZESTE-WEGE-PROBLEM in ungerichteten Graphen. (Beachte, dass wir in Kapitel 7 das KÜRZESTE-WEGE-PROBLEM in ungerichteten Graphen nicht lösen konnten, außer für den Fall nichtnegativer Gewichte.)
- Ist $T = V(G)$, so sind die T -Joins der Kardinalität $\frac{|V(G)|}{2}$ genau die perfekten Matchings. Somit kann das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM durch Addition einer großen Konstante zu jedem Kantengewicht auf das MINIMUM-WEIGHT- T -JOIN-PROBLEM zurückgeführt werden.

Zunächst betrachten wir das folgende einfache Resultat:

Proposition 12.6. *Sei G ein Graph und seien $T, T' \subseteq V(G)$. Sei J ein T -Join und J' ein T' -Join. Dann ist $J \triangle J'$ ein $(T \triangle T')$ -Join.*

Beweis: Für jedes $v \in V(G)$ gilt

$$\begin{aligned} |\delta_{J \triangle J'}(v)| &\equiv |\delta_J(v)| + |\delta_{J'}(v)| \pmod{2} \\ &\equiv |\{v\} \cap T| + |\{v\} \cap T'| \pmod{2} \\ &\equiv |\{v\} \cap (T \triangle T')| \pmod{2}. \end{aligned}$$

□

Hauptziel dieses Abschnitts ist es, einen polynomiellen Algorithmus für das MINIMUM-WEIGHT- T -JOIN-PROBLEM zu beschreiben. Die Frage, ob es überhaupt einen T -Join gibt, lässt sich leicht beantworten:

Proposition 12.7. *Sei G ein Graph und $T \subseteq V(G)$. Es gibt einen T -Join in G genau dann, wenn $|V(C) \cap T|$ für jede Zusammenhangskomponente C von G gerade ist.*

Beweis: Ist J ein T -Join, so gilt $\sum_{v \in V(C)} |J \cap \delta(v)| = 2|J \cap E(C)|$ für jede Zusammenhangskomponente C von G . Somit ist $|J \cap \delta(v)|$ ungerade für eine gerade Anzahl von Knoten $v \in V(C)$. Da J ein T -Join ist, folgt, dass $|V(C) \cap T|$ gerade ist.

Zur umgekehrten Richtung, sei $|V(C) \cap T|$ gerade für jede Zusammenhangskomponente C von G . Dann kann man die Knoten von T in Paare $\{v_1, w_1\}, \dots, \{v_k, w_k\}$ einteilen, mit $k = \frac{|T|}{2}$ und der Eigenschaft, dass v_i und w_i für $i = 1, \dots, k$ in derselben Zusammenhangskomponente liegen. Sei P_i ein v_i-w_i -Weg ($i = 1, \dots, k$) und $J := E(P_1) \triangle E(P_2) \triangle \dots \triangle E(P_k)$. Mit Proposition 12.6 folgt, dass J ein T -Join ist. □

Wir nennen ein T -Join optimal falls es minimales Gewicht hat. Wir haben das folgende einfache Optimalitätskriterium:

Proposition 12.8. Ein T -Join J in einem Graphen G mit Gewichten $c : E(G) \rightarrow \mathbb{R}$ ist genau dann optimal, wenn $c(J \cap E(C)) \leq c(E(C) \setminus J)$ für jeden Kreis C in G .

Beweis: Ist $c(J \cap E(C)) > c(E(C) \setminus J)$, so ist $J \triangle E(C)$ ein T -Join mit geringerem Gewicht als J . Ist andererseits J' ein T -Join mit $c(J') < c(J)$, so ist $J' \triangle J$ eulersch, d. h. eine Vereinigung von Kreisen, wobei für mindestens einen dieser Kreise, etwa C , die Ungleichung $c(J \cap E(C)) > c(J' \cap E(C)) = c(E(C) \setminus J)$ gilt. \square

Proposition 12.8 kann als Spezialfall von Satz 9.7 betrachtet werden. Wir werden nun das MINIMUM-WEIGHT-T-JOIN-PROBLEM mit nichtnegativen Gewichten lösen, indem wir es auf das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM zurückführen. Die Hauptidee ist in folgendem Lemma enthalten:

Lemma 12.9. Sei G ein Graph, $c : E(G) \rightarrow \mathbb{R}_+$ und $T \subseteq V(G)$. Jeder optimale T -Join in G ist die disjunkte Vereinigung der Kantenmengen von $\frac{|T|}{2}$ Wegen, dessen Endknoten verschieden sind und in T liegen, zuzüglich eventueller Kreise mit Gewicht null.

Beweis: Mittels Induktion über $|T|$. Der Fall $T = \emptyset$ ist trivial, da das minimale Gewicht eines \emptyset -Joins null ist.

Sei J ein optimaler T -Join in G ; o. B. d. A. können wir annehmen, dass J keinen Kreis mit Gewicht null enthält. Nach Proposition 12.8 enthält J keinen Kreis mit positivem Gewicht. Da c nichtnegativ ist, bildet J einen Wald. Seien x, y zwei Blätter derselben Zusammenhangskomponente dieses Waldes, d. h. $|J \cap \delta(x)| = |J \cap \delta(y)| = 1$, und sei P der x - y -Weg mit Kanten in J . Dann gilt $x, y \in T$, und $J \setminus E(P)$ ist ein $(T \setminus \{x, y\})$ -Join mit minimalem Gewicht (gäbe es einen leichteren $(T \setminus \{x, y\})$ -Join J' , so wäre $J' \triangle E(P)$ ein T -Join mit geringerem Gewicht als J). Die Aussage folgt nun mittels der Induktionsvoraussetzung. \square

Satz 12.10. (Edmonds und Johnson [1973]) Im Falle nichtnegativer Gewichte kann das MINIMUM-WEIGHT-T-JOIN-PROBLEM in $O(n^3)$ -Zeit gelöst werden.

Beweis: Sei (G, c, T) eine Instanz. Zunächst lösen wir ein KÜRZESTE-WEGE-PROBLEM FÜR ALLE PAARE in (G, c) , genauer: In dem Graphen, den wir durch das Ersetzen einer jeden Kante durch ein Paar entgegengesetzt orientierter Kanten mit demselben Gewicht erhalten. Nach Satz 7.8 benötigt dies $O(mn + n^2 \log n)$ -Zeit. Insbesondere erhalten wir den metrischen Abschluss (\bar{G}, \bar{c}) von (G, c) (siehe Korollar 7.10).

Nun bestimmen wir ein perfektes Matching M mit minimalem Gewicht in $(\bar{G}[T], \bar{c})$. Nach Korollar 11.12 benötigt dies $O(n^3)$ -Zeit. Nach Lemma 12.9 ist $\bar{c}(M)$ höchstens gleich dem minimalen Gewicht eines T -Joins.

Wir betrachten nun den kürzesten x - y -Weg in G für jedes $\{x, y\} \in M$ (diese haben wir bereits berechnet). Sei J die symmetrische Differenz der Kantenmengen aller dieser Wege. Offensichtlich ist J ein T -Join in G . Ferner gilt $c(J) \leq \bar{c}(M)$, also ist J optimal. \square

Diese Methode funktioniert jedoch nicht bei negativen Gewichten, da dann negative Kreise auftauchen würden. Wir können jedoch das MINIMUM-WEIGHT-T-JOIN-PROBLEM mit beliebigen Gewichten auf dasjenige mit nichtnegativen Gewichten zurückführen:

Satz 12.11. *Sei G ein Graph mit Gewichten $c : E(G) \rightarrow \mathbb{R}$ und $T \subseteq V(G)$. Sei E^- die Menge der Kanten mit negativem Gewicht, V^- die Menge der mit einer ungeraden Anzahl von negativen Kanten inzidenten Knoten und $d : E(G) \rightarrow \mathbb{R}_+$ mit $d(e) := |c(e)|$.*

Dann ist $J \triangle E^-$ ein T -Join mit minimalem c -Gewicht genau dann, wenn J ein $(T \triangle V^-)$ -Join mit minimalem d -Gewicht ist.

Beweis: Da E^- ein V^- -Join ist, folgt mit Proposition 12.6, dass $J \triangle E^-$ genau dann ein T -Join ist, wenn J ein $(T \triangle V^-)$ -Join ist. Ferner gilt für jede Teilmenge J von $E(G)$:

$$\begin{aligned} c(J \triangle E^-) &= c(J \setminus E^-) + c(E^- \setminus J) \\ &= d(J \setminus E^-) + c(E^- \setminus J) + c(J \cap E^-) + d(J \cap E^-) \\ &= d(J) + c(E^-). \end{aligned}$$

Der Satz folgt nun, da $c(E^-)$ konstant ist. \square

Korollar 12.12. *Das MINIMUM-WEIGHT-T-JOIN-PROBLEM kann in $O(n^3)$ -Zeit gelöst werden.*

Beweis: Dies folgt sofort mittels Satz 12.10 und Satz 12.11. \square

Wir sind nun endlich in der Lage, das KÜRZESTE-WEGE-PROBLEM in ungerichteten Graphen zu lösen:

Korollar 12.13. *Das Problem der Bestimmung eines kürzesten Weges zwischen zwei vorgegebenen Knoten in einem ungerichteten Graphen mit konservativen Gewichten kann in $O(n^3)$ -Zeit gelöst werden.*

Beweis: Seien s und t zwei vorgegebene Knoten. Wir setzen $T := \{s, t\}$ und wenden Korollar 12.12 an. Nach dem Entfernen von Kreisen mit Gewicht null ist der resultierende T -Join die Kantenmenge eines kürzesten s - t -Weges. \square

Natürlich ergibt dies auch einen $O(mn^3)$ -Algorithmus zur Bestimmung eines Kreises mit minimalem Gesamtgewicht in einem ungerichteten Graphen mit konservativen Gewichten (und insbesondere zur Bestimmung der Taille des Graphen). Interessiert uns das KÜRZESTE-WEGE-PROBLEM FÜR ALLE PAARE in ungerichteten Graphen, so brauchen wir nicht $\binom{n}{2}$ unabhängige Gewichtete-Matching-Berechnungen auszuführen (dies würde eine $O(n^5)$ -Laufzeit ergeben). Unter Verwendung der Postoptimierungsresultate von Abschnitt 11.4 können wir den folgenden Satz beweisen:

Satz 12.14. Das Problem der Bestimmung kürzester Wege für alle Knotenpaare in einem ungerichteten Graphen G mit konservativen Gewichten $c : E(G) \rightarrow \mathbb{R}$ kann in $O(n^4)$ -Zeit gelöst werden.

Beweis: Nach Satz 12.11 und dem Beweis von Korollar 12.13 müssen wir einen optimalen $(\{s, t\} \triangle V^-)$ -Join bezüglich der Gewichte $d(e) := |c(e)|$ für alle $s, t \in V(G)$ berechnen, wobei V^- die Menge der mit einer ungeraden Anzahl von negativen Kanten inzidenten Knoten ist. Sei $\bar{d}(\{x, y\}) := \text{dist}_{(G, d)}(x, y)$ für $x, y \in V(G)$ und H_X der vollständige Graph mit der Knotenmenge $X \triangle V^-$ ($X \subseteq V(G)$). Nach dem Beweis von Satz 12.10 genügt es, für alle s und t ein perfektes Matching mit minimalem Gewicht in $(H_{\{s, t\}}, \bar{d})$ zu berechnen.

Unser $O(n^4)$ -Algorithmus läuft wie folgt. Zunächst berechnen wir \bar{d} (siehe Korollar 7.10) und wenden den GEWICHTETEN MATCHING-ALGORITHMUS auf die Instanz (H_\emptyset, \bar{d}) an. Bis hierher haben wir $O(n^3)$ -Zeit benötigt.

Nun zeigen wir, dass wir für beliebige s und t ein perfektes Matching mit minimalem Gewicht in $(H_{\{s, t\}}, \bar{d})$ in $O(n^2)$ -Zeit berechnen können. Es gibt vier Fälle:

Fall 1: $s, t \notin V^-$. Dann fügen wir diese beiden Knoten hinzu und optimieren erneut mittels Lemma 11.13. Auf diese Weise erhalten wir in $O(n^2)$ -Zeit ein perfektes Matching mit minimalem Gewicht in $(H_{\{s, t\}}, \bar{d})$.

Fall 2: $s, t \in V^-$. Dann bilden wir H' , indem wir zwei Hilfsknoten s', t' und zwei Kanten $\{s, s'\}, \{t, t'\}$ beliebigen Gewichtes hinzufügen. Nun optimieren wir erneut mittels Lemma 11.13 und entfernen anschließend die zwei neuen Kanten aus dem resultierenden perfekten Matching mit minimalem Gewicht in H' .

Fall 3: $s \in V^-$ und $t \notin V^-$. Dann bilden wir H' , indem wir t , einen Hilfsknoten s' und eine Kante $\{s, s'\}$ beliebigen Gewichtes hinzufügen, zuzüglich der Kanten, mit denen t inzident ist. Nun optimieren wir erneut mittels Lemma 11.13 und entfernen anschließend die Kante $\{s, s'\}$ aus dem resultierenden perfekten Matching mit minimalem Gewicht in H' .

Fall 4: $s \notin V^-$ und $t \in V^-$. Symmetrisch zu Fall 3. □

Die Laufzeit wurde von Gabow [1983] auf $O(\min\{n^3, nm \log n\})$ verbessert.

12.3 T-Joins und T-Schnitte

In diesem Abschnitt werden wir eine polyedrische Beschreibung des MINIMUM-WEIGHT-T-JOIN-PROBLEMS entwickeln. Im Gegensatz zu der Beschreibung des Perfekten-Matching-Polytops (Satz 11.15), wo wir für jeden Schnitt $\delta(X)$ mit ungeradem $|X|$ eine Restriktion hatten, benötigen wir hier eine Restriktion pro T -Schnitt. Ein **T -Schnitt** ist ein Schnitt $\delta(X)$ mit $|X \cap T|$ ungerade. Die folgende einfache Tatsache wird sich als sehr nützlich erweisen:

Proposition 12.15. Sei G ein ungerichteter Graph und $T \subseteq V(G)$ mit $|T|$ gerade. Dann gilt $J \cap C \neq \emptyset$ für jeden T -Join J und T -Schnitt C .

Beweis: Es sei $C = \delta(X)$, dann ist $|X \cap T|$ ungerade. Also ist die Anzahl der Kanten in $J \cap C$ ungerade, insbesondere nicht gleich Null. \square

Eine stärkere Aussage befindet sich in Aufgabe 12.

Aus Proposition 12.15 folgt, dass die minimale Kardinalität eines T-Joins größer oder gleich der maximalen Anzahl der paarweise kantendisjunkten T-Schnitte ist. Im Allgemeinen gilt nicht Gleichheit: betrachte das Gegenbeispiel $G = K_4$ und $T = V(G)$. Für bipartite Graphen gilt jedoch Gleichheit:

Satz 12.16. (Seymour [1981])

Sei G ein bipartiter Graph und $T \subseteq V(G)$, so dass es einen T-Join in G gibt. Dann ist die minimale Kardinalität eines T-Joins gleich der maximalen Anzahl der paarweise kantendisjunkten T-Schnitte.

Beweis: (Sebő [1987]) Wir brauchen nur “ \leq ” zu zeigen. Der Beweis erfolgt mittels Induktion über $|V(G)|$. Die Aussage ist trivial, falls $T = \emptyset$ (insbesondere, falls $|V(G)| = 1$). Also nehmen wir an, dass $|V(G)| \geq |T| \geq 2$. Sei $\tau(G, T)$ die minimale Kardinalität eines T-Joins in G (und $\tau(G, T) = \infty$ falls kein T-Join existiert). Wir werden Proposition 12.8 (für Einheitsgewichte) und Proposition 12.6 mehrfach benutzen. Wir können hier annehmen, dass es einen T-Join in G gibt, da es sonst einen leeren T-Schnitt gibt, den wir unendlich oft auflisten können.

Wähle $a, b \in V(G)$ so, dass $\tau(G, T \triangle \{a\} \triangle \{b\})$ minimal ist. Sei $T' := T \triangle \{a\} \triangle \{b\}$. Da die Endknoten einer Kante in einem T-Join mit minimalem Gewicht (beachte, dass $T \neq \emptyset$) eine mögliche Wahl von a und b darstellen, folgt $\tau(G, T') < \tau(G, T)$ und somit ist $a \neq b$ und $T' = T \triangle \{a, b\}$.

Behauptung: Für jeden T-Join J mit minimalem Gewicht in G gilt $|J \cap \delta(a)| = |J \cap \delta(b)| = 1$.

Zum Beweis dieser Behauptung sei J ein T-Join mit minimalem Gewicht und J' ein T' -Join mit minimalem Gewicht. Es ist $J \triangle J'$ die paarweise kantendisjunkte Vereinigung eines a - b -Weges P mit einigen Kreisen C_1, \dots, C_k . Es gilt $|E(C_i) \cap J| = |E(C_i) \cap J'|$ für alle i , da J und J' beide kardinalitätsminimal sind. Somit ist $|J \triangle E(P)| = |J'|$, und $J'' := J \triangle E(P)$ ist auch ein T' -Join mit minimalem Gewicht. Nun gilt $J'' \cap \delta(a) = J'' \cap \delta(b) = \emptyset$: Wäre etwa $\{b, b'\} \in J''$, so ist $J'' \setminus \{\{b, b'\}\}$ ein $(T \triangle \{a\} \triangle \{b'\})$ -Join und es folgt $\tau(G, T \triangle \{a\} \triangle \{b'\}) < |J''| = |J'| = \tau(G, T')$, im Widerspruch zu der Wahl von a und b . Daraus folgern wir, dass $|J \cap \delta(a)| = |J \cap \delta(b)| = 1$, womit die Behauptung bewiesen ist.

Insbesondere gilt $a, b \in T$. Nun sei J ein T-Join in G mit minimalem Gewicht. Kontrahiere $B := \{b\} \cup \Gamma(b)$ zu einem einzigen Knoten v_B ; es sei G^* der resultierende Graph. Es ist G^* auch bipartit. Sei $T^* := T \setminus B$, falls $|T \cap B|$ gerade ist, sonst sei $T^* := (T \setminus B) \cup \{v_B\}$. Die aus J durch die Kontraktion von B resultierende Menge J^* ist offensichtlich ein T^* -Join in G^* . Da $\Gamma(b)$ eine stabile Menge in G ist (da G bipartit), folgt mit der Behauptung, dass $|J| = |J^*| + 1$.

Es genügt zu zeigen, dass J^* ein T^* -Join in G^* mit minimalem Gewicht ist, denn damit haben wir $\tau(G, T) = |J| = |J^*| + 1 = \tau(G^*, T^*) + 1$ und der Satz folgt dann mittels Induktion (beachte, dass $\delta(b)$ und $E(G^*)$ disjunkt sind und $\delta(b)$ ein T-Schnitt in G ist).

Also nehmen wir an, J^* sei kein T^* -Join mit minimalem Gewicht in G^* . Nach Proposition 12.8 gibt es dann einen Kreis C^* in G^* mit $|J^* \cap E(C^*)| > |E(C^*) \setminus J^*|$. Da G^* bipartit ist, gilt $|J^* \cap E(C^*)| \geq |E(C^*) \setminus J^*| + 2$. Die Kantenmenge $E(C^*)$ entspricht einer Kantenmenge Q in G . Aber Q kann kein Kreis sein, da $|J \cap Q| > |Q \setminus J|$ und J ein T -Join mit minimalem Gewicht ist. Somit ist Q ein x - y -Weg in G für irgendwelche $x, y \in \Gamma(b)$ mit $x \neq y$. Sei C der durch Q zusammen mit $\{x, b\}$ und $\{b, y\}$ gebildete Kreis in G . Da J ein T -Join in G mit minimalem Gewicht ist, gilt

$$|J \cap E(C)| \leq |E(C) \setminus J| \leq |E(C^*) \setminus J^*| + 2 \leq |J^* \cap E(C^*)| \leq |J \cap E(C)|.$$

Also muss überall Gleichheit gelten, insbesondere folgt $\{x, b\}, \{b, y\} \notin J$ und $|J \cap E(C)| = |E(C) \setminus J|$. Somit ist $\bar{J} := J \triangle E(C)$ auch ein T -Join mit minimalem Gewicht und es gilt $|\bar{J} \cap \delta(b)| = 3$. Dies widerspricht aber der Behauptung. \square

Korollar 12.17. *Sei G ein Graph, $c : E(G) \rightarrow \mathbb{Z}_+$ und $T \subseteq V(G)$ mit der Eigenschaft, dass es einen T -Join in G gibt. Seien k die minimalen Kosten eines T -Joins in G . Dann gibt es T -Schnitte C_1, \dots, C_{2k} , so dass jede Kante $e \in E(G)$ in höchstens $2c(e)$ von ihnen enthalten ist.*

Beweis: Sei E_0 die Menge der Kanten mit Gewicht Null. Wir bilden einen bipartiten Graphen G' , indem wir die Zusammenhangskomponenten von $(V(G), E_0)$ kontrahieren und jede Kante e durch einen Weg der Länge $2c(e)$ ersetzen. Sei T' die Menge der Knoten in G' , die den Zusammenhangskomponenten X von $(V(G), E_0)$ mit $|X \cap T|$ ungerade entsprechen.

Behauptung: Die minimale Kardinalität eines T' -Joins in G' ist $2k$.

Um die Behauptung zu beweisen, beachten wir zunächst, dass die minimale Kardinalität eines T' -Joins in G' nicht größer als $2k$ ist, da jeder T -Join J in G einem T' -Join in G' mit Kardinalität höchstens gleich $2c(J)$ entspricht. Zur umgekehrten Richtung: Sei J' ein T' -Join in G' . Dieser entspricht einer Kantenmenge J in G . Sei nun $\bar{T} := T \triangle \{v \in V(G) : |\delta(v) \cap J| \text{ ungerade}\}$. Dann enthält jede Zusammenhangskomponente X von $(V(G), E_0)$ eine gerade Anzahl von Knoten aus \bar{T} (da $|\delta(X) \cap J| \equiv |X \cap T| \pmod{2}$). Nach Proposition 12.7 hat $(V(G), E_0)$ einen \bar{T} -Join \bar{J} und $J \cup \bar{J}$ ist ein T -Join in G mit Gewicht $c(J) = \frac{|J'|}{2}$. Hiermit ist die Behauptung bewiesen.

Nach Satz 12.16 gibt es $2k$ paarweise kantendisjunkte T' -Schnitte in G' . Gehen wir zurück nach G , so liefern diese eine Liste von $2k$ T -Schnitten in G mit der Eigenschaft, dass jede Kante e in höchstens $2c(e)$ von ihnen enthalten ist. \square

Karzanov [1986] hat einen effizienten Algorithmus zur Bestimmung einer solchen T -Schnittpackung beschrieben.

T -Schnitte spielen auch eine entscheidende Rolle in der folgenden Beschreibung des T -Join-Polyeders:

Satz 12.18. (Edmonds und Johnson [1973]) *Sei G ein ungerichteter Graph, $c : E(G) \rightarrow \mathbb{R}_+$ und $T \subseteq V(G)$ mit der Eigenschaft, dass es einen T -Join in G*

gibt. Dann ist der Inzidenzvektor eines jeden T-Joins mit minimalem Gewicht eine optimale Lösung des LP

$$\min \left\{ cx : x \geq 0, \sum_{e \in C} x_e \geq 1 \text{ für alle } T\text{-Schnitte } C \right\}.$$

(Dieses Polyeder heißt das **T-Join-Polyeder** von G .)

Beweis: Nach Proposition 12.15 erfüllt der Inzidenzvektor eines T-Joins die Nebenbedingungen des obigen LP.

Ferner gilt mit 12.7, dass jeder 0-1-Vektor, der die Nebenbedingungen des obigen LP erfüllt, der Inzidenzvektor einer Kantenmenge ist, die einen T-Join enthält. Somit genügt es nach “(g) \Rightarrow (f)” des Satzes 5.14 zu zeigen, dass für jedes ganzzahlige c dessen Minimum endlich ist, d. h. für jedes $c : E(G) \rightarrow \mathbb{Z}_+$, der LP-Wert ganzzahlig ist.

Sei also $c : E(G) \rightarrow \mathbb{Z}_+$ und sei k das minimale Gewicht (bezüglich c) eines T-Joins in G . Nach Korollar 12.17 gibt es T -Schnitte C_1, \dots, C_{2k} in G , so dass jede Kante e in höchstens $2c(e)$ von ihnen enthalten ist.

Für jede zulässige Lösung x des obigen LP gilt somit

$$2cx \geq \sum_{i=1}^{2k} \sum_{e \in C_i} x_e \geq 2k,$$

womit bewiesen ist, dass der optimale Zielfunktionswert des LP gleich k ist. \square

Hieraus folgt nun Satz 11.15: Sei G ein Graph mit einem perfekten Matching und $T := V(G)$. Dann folgt mit Satz 12.18:

$$\min \left\{ cx : x \geq 0, \sum_{e \in C} x_e \geq 1 \text{ für alle } T\text{-Schnitte } C \right\}$$

ist ganzzahlig für jedes $c \in \mathbb{Z}^{E(G)}$ mit der Eigenschaft, dass obiges Minimum endlich ist. Nach Satz 5.14 ist das Polyeder ganzzahlig, somit auch die Seitenfläche

$$\left\{ x \in \mathbb{R}_+^{E(G)} : \sum_{e \in C} x_e \geq 1 \text{ für alle } T\text{-Schnitte } C, \sum_{e \in \delta(v)} x_e = 1 \text{ für alle } v \in V(G) \right\}.$$

Man kann ferner eine Beschreibung der konvexen Hülle der Inzidenzvektoren aller T-Joins ableiten (Aufgabe 16). Mit Satz 12.18 und Satz 4.21 (zusammen mit Korollar 3.33) ergibt sich ein weiterer polynomieller Algorithmus für das MINIMUM-WEIGHT-T-JOIN-PROBLEM, wenn wir das SEPARATIONS-PROBLEM für die obige Beschreibung lösen können. Dies ist offensichtlich äquivalent mit der Überprüfung der Existenz eines T -Schnittes mit Kapazität kleiner als eins (hier dient x als Kapazitätsvektor). Somit genügt es, das folgende Problem zu lösen:

MINIMUM-CAPACITY-*T*-CUT-PROBLEM

Instanz: Ein Graph G , Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und eine nichtleere Menge $T \subseteq V(G)$ mit gerader Kardinalität.

Aufgabe: Bestimme einen T -Schnitt minimaler Kapazität in G .

Beachte, dass das MINIMUM-CAPACITY-*T*-CUT-PROBLEM auch das SEPARATIONS-PROBLEM für das Perfekte-Matching-Polytop löst (Satz 11.15; $T := V(G)$). Der folgende Satz löst das MINIMUM-CAPACITY-*T*-CUT-PROBLEM: Es genügt, die Fundamentalschnitte eines Gomory-Hu-Baumes zu betrachten. Wie erinnern daran, dass man einen Gomory-Hu-Baum für einen ungerichteten Graphen mit Kapazitäten in $O(n^4)$ -Zeit bestimmen kann (Satz 8.38).

Satz 12.19. (Padberg und Rao [1982]) *Sei G ein ungerichteter Graph mit Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$. Sei ferner H ein Gomory-Hu-Baum für (G, u) und $T \subseteq V(G)$ mit $|T| \geq 2$ gerade. Dann gibt es unter den Fundamentalschnitten von H einen T -Schnitt minimaler Kapazität. Somit kann ein T -Schnitt minimaler Kapazität in $O(n^4)$ -Zeit bestimmt werden.*

Beweis: Sei $\delta_G(X)$ ein T -Schnitt minimaler Kapazität in (G, u) . Sei J die Menge derjenigen Kanten e von H , für die $|C_e \cap T|$ ungerade ist, wobei C_e eine Zusammenhangskomponente von $H - e$ ist. Da $|\delta_J(x)| \equiv \sum_{e \in \delta_H(x)} |C_e \cap T| \equiv |\{x\} \cap T| \pmod{2}$ für alle $x \in V(G)$, ist J ein T -Join in H . Nach Proposition 12.15 gibt es eine Kante $f \in \delta_H(X) \cap J$. Dann folgt

$$u(\delta_G(X)) \geq \min\{u(\delta_G(Y)) : |Y \cap f| = 1\} = u(\delta_G(C_f)),$$

also ist $\delta_G(C_f)$ ein T -Schnitt minimaler Kapazität. □

12.4 Der Satz von Padberg und Rao

Satz 12.19 wurde von Letchford, Reinelt und Theis [2008] folgendermaßen verallgemeinert:

Lemma 12.20. *Sei G ein ungerichteter Graph mit mindestens einer Kante, $T \subseteq V(G)$ mit $|T|$ gerade, und $c, c' : E(G) \rightarrow \mathbb{R}_+ \cup \{\infty\}$. Dann gibt es einen $O(n^4)$ -Algorithmus zur Bestimmung von Mengen $X \subseteq V(G)$ und $F \subseteq \delta(X)$ mit $|X \cap T| + |F|$ ungerade und $\sum_{e \in \delta(X) \setminus F} c(e) + \sum_{e \in F} c'(e)$ minimal.*

Beweis: Da die Hinzufügung von Kanten e mit $c(e) = 0$ und $c'(e) = \infty$ nichts verändert, können wir annehmen, dass G zusammenhängend ist. Sei $d(e) := \min\{c(e), c'(e)\}$ ($e \in E(G)$). Seien $E' := \{e \in E(G) : c'(e) < c(e)\}$ und $V' := \{v \in V(G) : |\delta_{E'}(v)|$ ungerade}. Sei $T' := T \Delta V'$. Beachte: Für $X \subseteq V(G)$ gilt $|X \cap T| + |\delta(X) \cap E'| \equiv |X \cap T| + |X \cap V'| \equiv |X \cap T'| \pmod{2}$.

Der Algorithmus berechnet zunächst einen Gomory-Hu-Baum H für (G, d) . Sei X_f für jedes $f \in E(H)$ die Knotenmenge einer Zusammenhangskomponente von

$H - f$. Sei $g_f \in \delta_G(X_f)$ mit $|c'(g_f) - c(g_f)|$ minimal. Sei ferner $F_f := \delta_G(X_f) \cap E'$ falls $|X_f \cap T'|$ ungerade, und $F_f := (\delta_G(X_f) \cap E') \triangle \{g_f\}$ sonst. Schließlich wählen wir noch ein $f \in E(H)$ mit $\sum_{e \in \delta_G(X_f) \setminus F_f} c(e) + \sum_{e \in F_f} c'(e)$ minimal und setzen $X := X_f$ und $F := F_f$.

Offensichtlich wird die Gesamlaufzeit durch die Berechnung des Gomory-Hu-Baumes dominiert.

Seien $X^* \subseteq V(G)$ und $F^* \subseteq \delta(X^*)$ optimale Mengen, d. h. $|X^* \cap T| + |F^*|$ ist ungerade und $\sum_{e \in \delta_G(X^*) \setminus F^*} c(e) + \sum_{e \in F^*} c'(e)$ ist minimal.

Fall 1: Es sei $|X^* \cap T'|$ ungerade. Dann ist die Menge derjenigen $f \in E(H)$ mit $|X_f \cap T'|$ ungerade ein T' -Join in H und hat somit einen nicht verschwindenden Durchschnitt mit dem T' -Schnitt $\delta_H(X^*)$. Sei $f \in \delta_H(X^*)$ mit $|X_f \cap T'|$ ungerade. Nach Definition des Gomory-Hu-Baumes folgt dann $d(\delta_G(X_f)) \leq d(\delta_G(X^*))$ und $\sum_{e \in \delta_G(X_f) \setminus F_f} c(e) + \sum_{e \in F_f} c'(e) = d(\delta_G(X_f))$.

Fall 2: Es sei $|X^* \cap T'|$ gerade. Sei $g^* \in \delta_G(X^*)$ mit $|c'(g^*) - c(g^*)|$ minimal. Der eindeutig definierte Kreis in $H + g^*$ enthält eine Kante $f \in \delta_H(X^*)$. Dann folgt $\sum_{e \in \delta_G(X^*) \setminus F^*} c(e) + \sum_{e \in F^*} c'(e) = d(\delta_G(X^*)) + |c'(g^*) - c(g^*)| \geq d(\delta_G(X_f)) + |c'(g^*) - c(g^*)| \geq \sum_{e \in \delta_G(X_f) \setminus F_f} c(e) + \sum_{e \in F_f} c'(e)$. Die erste Ungleichung folgt aus der Definition des Gomory-Hu-Baumes (beachte, dass $f \in \delta_H(X^*)$), und die zweite Ungleichung folgt, da $g^* \in \delta_G(X_f)$. \square

Hiermit können wir das SEPARATIONS-PROBLEM für das b -Matching-Polytop (Satz 12.3) in polynomieller Zeit lösen. Dieses Resultat ist als Padberg-Rao-Satz bekannt. Letchford, Reinelt und Theis [2008] haben den Beweis vereinfacht und die Laufzeit verbessert:

Satz 12.21. (Padberg und Rao [1982], Letchford, Reinelt und Theis [2008]) Für einen ungerichteten Graphen G mit $u : E(G) \rightarrow \mathbb{N} \cup \{\infty\}$ und $b : V(G) \rightarrow \mathbb{N}$ kann das SEPARATIONS-PROBLEM für das b -Matching-Polytop von (G, u) in $O(n^4)$ -Zeit gelöst werden.

Beweis: Gegeben sei ein Vektor $x \in \mathbb{R}_+^{E(G)}$ mit $x_e \leq u(e)$ für alle $e \in E(G)$ und $\sum_{e \in \delta_G(v)} x_e \leq b(v)$ für alle $v \in V(G)$ (diese trivialen Ungleichungen lassen sich in linearer Zeit prüfen). Wir müssen den letzten Block Ungleichungen in Satz 12.3 prüfen. Im Beweis dieses Satzes sahen wir, dass diese Ungleichungen immer dann automatisch erfüllt sind, wenn $b(X) + u(F)$ gerade ist. Sie werden genau dann verletzt, wenn

$$b(X) - 2 \sum_{e \in E(G[X])} x_e + \sum_{e \in F} (u(e) - 2x_e) < 1$$

für ein $X \subseteq V(G)$ und ein $F \subseteq \delta(X)$ mit $b(X) + u(F)$ ungerade.

Nun erweitern wir G zu einem Graphen \bar{G} durch das Hinzufügen eines neuen Knotens z und der Kanten $\{z, v\}$ für alle $v \in V(G)$. Sei $T := \{v \in V(\bar{G}) : b(v) \text{ ungerade}\}$, wobei $b(z) := \sum_{v \in V(G)} b(v)$.

Sei ferner $E' := \{e \in E(G) : u(e) \text{ endlich und ungerade}\}$. Setze $c(e) := x_e$ und $c'(e) := u(e) - x_e$ für $e \in E'$, $c(e) := \min\{x_e, u(e) - x_e\}$ und $c'(e) := \infty$

für $e \in E(G) \setminus E'$, und $c(\{z, v\}) := b(v) - \sum_{e \in \delta_G(v)} x_e$ und $c'(\{z, v\}) := \infty$ für $v \in V(G)$.

Für jedes $X \subseteq V(G)$ setzen wir $D_X := \{e \in \delta_G(X) \setminus E' : u(e) < 2x_e\}$.

Für jedes $X \subseteq V(G)$ und $F \subseteq \delta_G(X) \cap E'$ folgt dann

$$|X \cap T| + |F| \equiv b(X) + u(F \cup D_X) \pmod{2}$$

und

$$\begin{aligned} c(\delta_{\bar{G}}(X) \setminus F) + c'(F) &= \sum_{v \in X} \left(b(v) - \sum_{e \in \delta_G(v)} x_e \right) + \sum_{e \in (\delta_G(X) \cap E') \setminus F} x_e \\ &\quad + \sum_{e \in \delta_G(X) \setminus E'} \min\{x_e, u(e) - x_e\} + \sum_{e \in F} (u(e) - x_e) \\ &= b(X) - 2 \sum_{e \in E(G[X])} x_e + \sum_{e \in F \cup D_X} (u(e) - 2x_e). \end{aligned}$$

Daraus schließen wir: Gibt es Mengen $X \subseteq V(\bar{G})$ und $F \subseteq \delta_{\bar{G}}(X)$ mit $c(\delta_{\bar{G}}(X) \setminus F) + c'(F) < 1$, so gilt $F \subseteq E'$ und o. B. d. A. ist $z \notin X$ (sonst gehe man zum Komplement über). Somit haben wir $b(X) - 2 \sum_{e \in E(G[X])} x_e + \sum_{e \in F \cup D_X} (u(e) - 2x_e) < 1$.

Ist umgekehrt $b(X) - 2 \sum_{e \in E(G[X])} x_e + \sum_{e \in F} (u(e) - 2x_e) < 1$ für ein $X \subseteq V(G)$ und ein $F \subseteq \delta_G(X)$, so gilt o. B. d. A. $D_X \subseteq F \subseteq D_X \cup E'$ und somit $c(\delta_{\bar{G}}(X) \setminus (F \setminus D_X)) + c'(F \setminus D_X) < 1$.

Damit ist das Separations-Problem zurückgeführt worden auf die Bestimmung von Mengen $X \subseteq V(\bar{G})$ und $F \subseteq \delta_{\bar{G}}(X)$ mit $|X \cap T| + |F|$ ungerade und $c(\delta_{\bar{G}}(X) \setminus F) + c'(F)$ minimal. Dies kann mit Lemma 12.20 bewerkstelligt werden.

□

Eine Verallgemeinerung dieses Resultats ist von Caprara und Fischetti [1996] angegeben worden. Aus dem Satz von Padberg und Rao folgt:

Korollar 12.22. *Das MAXIMUM-WEIGHT-b-MATCHING-PROBLEM kann in polynomieller Zeit gelöst werden.*

Beweis: Nach Korollar 3.33 müssen wir das in Satz 12.3 gegebene LP lösen. Nach Satz 4.21 genügt es, einen polynomiellen Algorithmus für das SEPARATIONS-PROBLEM zu besitzen. Ein solcher Algorithmus wurde in Satz 12.21 bereitgestellt.

□

Marsh [1979] hat Edmonds' GEWICHTETEN MATCHING-ALGORITHMUS auf das MAXIMUM-WEIGHT-b-MATCHING-PROBLEM erweitert. Dieser kombinatorische Algorithmus ist natürlich in der Praxis nützlicher als die ELLIPSOIDMETHODE. Satz 12.21 ist aber auch in anderer Hinsicht interessant (siehe z. B. Abschnitt 21.4). Siehe auch Gerards [1995]. Kombinatorische Algorithmen mit streng polynomieller Laufzeit wurden von Anstee [1987] beschrieben, und für den ungewichteten Fall von Goldberg und Karzanov [2004].

Aufgaben

1. Man zeige, dass man ein perfektes einfaches 2-Matching mit minimalem Gewicht in einem ungerichteten Graphen G in $O(n^6)$ -Zeit bestimmen kann.
- * 2. Sei G ein ungerichteter Graph und $b_1, b_2 : V(G) \rightarrow \mathbb{N}$. Man gebe eine Beschreibung der konvexen Hülle der Funktionen $f : E(G) \rightarrow \mathbb{Z}_+$ mit $b_1(v) \leq \sum_{e \in \delta(v)} f(e) \leq b_2(v)$ an.

Hinweis: Für $X, Y \subseteq V(G)$ mit $X \cap Y = \emptyset$ betrachte man die Ungleichung

$$\sum_{e \in E(G[X])} f(e) - \sum_{e \in E(G[Y]) \cup E(Y, Z)} f(e) \leq \left\lfloor \frac{1}{2} \left(\sum_{x \in X} b_2(x) - \sum_{y \in Y} b_1(y) \right) \right\rfloor,$$

wobei $Z := V(G) \setminus (X \cup Y)$. Man verwende Satz 12.3.

(Schrijver [1983])

- * 3. Kann man das Resultat von Aufgabe 2 durch Einführung oberer und unterer Kapazitäten auf den Kanten weiter verallgemeinern?

Bemerkung: Dies kann als ungerichtete Version des Problems von Aufgabe 2, Kapitel 9, betrachtet werden. Für eine gemeinsame Verallgemeinerung beider Probleme sowie des MINIMUM-WEIGHT-T-JOIN-PROBLEMS wird auf die Arbeiten von Edmonds und Johnson [1973] und von Schrijver [1983] hingewiesen. Sogar hier ist eine Beschreibung des Polytops bekannt, die TDI ist.

- * 4. Man beweise Satz 12.4.

Hinweis: Für die hinreichende Richtung verwende man den Satz von Tutte (Satz 10.13) und die Konstruktionen in den Beweisen von Satz 12.2 und Satz 12.3.

5. Das Teilgraph-Grad-Polytop eines Graphen G ist die konvexe Hülle aller Vektoren $b \in \mathbb{Z}_+^{V(G)}$ mit der Eigenschaft, dass G ein perfektes einfaches b -Matching besitzt. Man beweise, dass die Dimension dieses Polytops gleich $|V(G)| - k$ ist, wobei k die Anzahl der bipartiten Zusammenhangskomponenten von G ist.

- * 6. Für einen gegebenen ungerichteten Graphen ist eine ungerade Kreisüberdeckung eine Kantenmenge, die mindestens eine Kante aus jedem ungeraden Kreis enthält. Man zeige, wie man in polynomieller Zeit eine ungerade Kreisüberdeckung minimalen Gewichtes in einem planaren Graphen mit nichtnegativen Kantengewichten finden kann. Kann man dieses Problem auch für allgemeine Gewichte lösen?

Hinweis: Man betrachte das UNGERICHTETE CHINESISCHE POSTBOTEN-PROBLEM im planaren Dual und verwende Satz 2.26 und Korollar 2.45.

7. Man betrachte das MAXIMUM-WEIGHT-CUT-PROBLEM in planaren Graphen: Man bestimme einen Schnitt maximalen Gewichtes in einem gegebenen ungerichteten planaren Graphen G mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$. Kann man dieses Problem in polynomieller Zeit lösen?

Hinweis: Man benutze Aufgabe 6.

Bemerkung: Für allgemeine Graphen ist dieses Problem *NP*-schwer; siehe Satz 16.6.

(Hadlock [1975])

8. Gegeben sei ein Graph G mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ und eine Menge $T \subseteq V(G)$ mit $|T|$ gerade. Ein neuer Graph G' werde folgendermaßen definiert:

$$\begin{aligned} V(G') &:= \{(v, e) : v \in e \in E(G)\} \cup \\ &\quad \{\bar{v} : v \in V(G), |\delta_G(v)| + |\{v\} \cap T| \text{ ungerade}\}, \\ E(G') &:= \{\{(v, e), (w, e)\} : e = \{v, w\} \in E(G)\} \cup \\ &\quad \{\{(v, e), (v, f)\} : v \in V(G), e, f \in \delta_G(v), e \neq f\} \cup \\ &\quad \{\{\bar{v}, (v, e)\} : v \in e \in E(G), \bar{v} \in V(G')\}. \end{aligned}$$

Ferner definiere man $c'(\{(v, e), (w, e)\}) := c(e)$ für $e = \{v, w\} \in E(G)$ und $c'(e') = 0$ für alle weiteren Kanten e' von G' .

Man zeige, dass einem perfekten Matching mit minimalem Gewicht in G' ein T -Join minimalen Gewichtes in G entspricht. Ist diese Reduktion derjenigen im Beweis von Satz 12.10 vorzuziehen?

- * 9. Das folgende Problem kombiniert einfache perfekte b -Matchings mit T -Joins. Gegeben sei ein ungerichteter Graph G mit Gewichten $c : E(G) \rightarrow \mathbb{R}$, eine Partition der Knotenmenge $V(G) = R \dot{\cup} S \dot{\cup} T$ und eine Funktion $b : R \rightarrow \mathbb{Z}_+$. Gesucht wird eine Kantenmenge $J \subseteq E(G)$ mit minimalem Gewicht und mit $|J \cap \delta(v)| = b(v)$ für $v \in R$, $|J \cap \delta(v)|$ gerade für $v \in S$ und $|J \cap \delta(v)|$ ungerade für $v \in T$. Man zeige, wie man dieses Problem auf ein MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM zurückführen kann.

Hinweis: Man betrachte die Konstruktionen in Abschnitt 12.1 und Aufgabe 8.

- 10. Wir betrachten das folgende UNGERICHTETE MINIMUM-MEAN-CYCLE-PROBLEM: Gegeben sei ein ungerichteter Graph G und Gewichte $c : E(G) \rightarrow \mathbb{R}$. Man bestimme einen Kreis C in G , dessen Durchschnittsgewicht $\frac{c(E(C))}{|E(C)|}$ minimal ist. Sei (G, c) eine Instanz und \mathcal{D} die Menge der \emptyset -Joins in G . Für $a \in \mathbb{R}$ und $X \in \mathcal{D}$ setzen wir $c^a(X) := c(X) - a|X|$ und $\bar{c}^a(X) := \frac{c(X)}{|X|} - a$. Man beweise:

- Für jedes $a \in \mathbb{R}$ kann man ein Element $X \in \mathcal{D}$ mit minimalem $c^a(X)$ in $O(n^3)$ -Zeit finden.
- Sei $a \in \mathbb{R}$ mit $c^a(Y) \leq 0$ für alle $Y \in \mathcal{D}$, und $X \in \mathcal{D}$ mit minimalem $c^a(X)$. Dann gibt es ein $Y \in \mathcal{D}$ mit $c^{a+b}(Y) \leq 0$, wobei $b = \bar{c}^a(X)$.
- Man betrachte den folgenden Ablauf: Beginne mit $a = \max\{c(e) : e \in E(G)\}$. Berechne ein Element $X \in \mathcal{D}$ mit minimalem $c^a(X)$. Terminiere, falls $c^a(X) = 0$. Falls nicht, ersetze a durch $a + \bar{c}^a(X)$ und iteriere. Man zeige, dass $|X|$ mit jeder Iteration abnimmt.
- Man zeige, dass der in (c) angegebene Algorithmus eine optimale Lösung in $O(n^5)$ -Zeit berechnet.

Bemerkung: Die Laufzeit kann auf $O(n^2m + n^3 \log n)$ verbessert werden. Im Grunde derselbe Algorithmus funktioniert auch für andere Minimum-Ratio-Probleme.

(Karzanov [1985], Barahona [1993], Babenko und Karzanov [2009])

11. Gegeben sei ein Graph G und eine Menge $T \subseteq V(G)$. Man beschreibe einen Algorithmus mit linearer Laufzeit, der einen T -Join in G findet oder entscheidet, dass es keinen solchen gibt.

Hinweis: Man betrachte einen maximalen Wald in G .

12. Sei G ein ungerichteter Graph, $T \subseteq V(G)$ mit $|T|$ gerade und $F \subseteq E(G)$. Man beweise: Die Kantenmenge F hat nichtleeren Durchschnitt mit jedem T -Join genau dann, wenn F einen T -Schnitt enthält. Es hat F nichtleeren Durchschnitt mit jedem T -Schnitt genau dann, wenn F einen T -Join enthält.
13. Sei G ein Graph und $F \subseteq E(G)$. Man beweise, dass es eine Menge A mit $F \subseteq A \subseteq E(G)$ gibt, so dass $(V(G), A)$ genau dann eulersch ist, wenn G keinen Schnitt B mit $B \subseteq F$ und $|B|$ ungerade enthält.
- * 14. Sei G ein planarer 2-fach zusammenhängender Graph mit einer festen Einbettung. Sei ferner C der das äußere Gebiet berandende Kreis und T eine Teilmenge von $V(C)$ mit gerader Kardinalität. Man beweise, dass die minimale Kardinalität eines T -Joins gleich der maximalen Anzahl der paarweise kantendisjunkten T -Schnitte ist.

Hinweis: Man färbe die Kanten von C rot und blau, so dass beim Durchlaufen von C die Farbe genau in den Knoten von T wechselt. Man betrachte das planare Dual, spalte den das äußere Gebiet darstellenden Knoten in einen roten und einen blauen Knoten und wende den Satz von Menger (Satz 8.9) an.

15. Man beweise Satz 12.18 mittels Satz 11.15 und der Konstruktion in Aufgabe 8.
(Edmonds und Johnson [1973])
16. Sei G ein ungerichteter Graph und $T \subseteq V(G)$ mit $|T|$ gerade. Man beweise, dass die konvexe Hülle der Inzidenzvektoren aller T -Joins in G die Menge derjenigen Vektoren $x \in [0, 1]^{E(G)}$ ist, welche die Ungleichung

$$\sum_{e \in \delta_G(X) \setminus F} x_e + \sum_{e \in F} (1 - x_e) \geq 1$$

für alle $X \subseteq V(G)$ und $F \subseteq \delta_G(X)$ mit $|X \cap T| + |F|$ ungerade erfüllen.

Hinweis: Man verwende Satz 12.18 und Satz 12.11.

17. Sei G ein ungerichteter Graph. Man zeige, dass der von den Inzidenzvektoren aller Kreise in G erzeugte Kegel gegeben wird durch

$$\left\{ x \in \mathbb{R}^{E(G)} : x \geq 0, \sum_{e' \in C \setminus \{e\}} x(e') \geq x(e) \text{ für alle Schnitte } C \text{ in } G \text{ und } e \in C \right\}.$$

Hinweis: Man benutze Aufgabe 16.

18. Man stelle das UNGERICHTETE MINIMUM-MEAN-CYCLE-PROBLEM als ein LP mit polynomieller Anzahl von Variablen und Nebenbedingungen dar.

Hinweis: Man benutze Aufgabe 17, füge die Nebenbedingung $x(E(G)) = 1$ hinzu und wende das Max-Flow-Min-Cut-Theorem an.

(Barahona [1993])

Bemerkung: Da wir zusätzliche Variablen benötigen, wird dies eine erweiterte

Formulierung genannt (siehe Yannakakis [1991]). Barahona [1993] hat ferner gezeigt, dass man das MINIMUM-WEIGHT-*T*-JOIN-PROBLEM lösen kann, indem man mit einem beliebigen *T*-Join beginnt und schrittweise Proposition 12.8 anwendet und dabei auf einen Kreis mit minimalem durchschnittlichem Kantengewicht achtet, wobei die Gewichte des aktuellen *T*-Joins mit -1 multipliziert werden. Die Anzahl der Iterationen kann durch $O(m^2 \log n)$ beschränkt werden. Somit kann man das MINIMUM-WEIGHT-*T*-JOIN-PROBLEM lösen (und damit auch das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM), indem man eine polynomielle Anzahl polynomiell großer LPs löst. Trotzdem hat jede LP-Darstellung für das gewichtete Matching exponentielle Größe (siehe Rothvoß [2017]).

19. Sei G ein ungerichteteter Graph und $T \subseteq V(G)$ mit $|T| = 2k$ gerade. Man beweise, dass die minimale Kardinalität eines *T*-Schnittes in G gleich dem maximalen Wert von $\min_{i=1}^k \lambda_{s_i, t_i}$ für alle Einteilungen $T = \{s_1, t_1, s_2, t_2, \dots, s_k, t_k\}$ von T in Paare ist. (Es bezeichnet $\lambda_{s,t}$ die maximale Anzahl der paarweise kantendisjunkten s - t -Wege.) Ist eine gewichtete Version dieser Min-Max Formel denkbar?

Hinweis: Man verwende Satz 12.19.

(Rizzi [2002])

20. Diese Aufgabe beinhaltet einen Algorithmus für das MINIMUM-CAPACITY-*T*-CUT-PROBLEM ohne Verwendung von Gomory-Hu-Bäumen. Der Algorithmus ist rekursiv und läuft für gegebene G, u und T wie folgt:

1. Zunächst bestimmt man eine Menge $X \subseteq V(G)$ mit $T \cap X \neq \emptyset$ und $T \setminus X \neq \emptyset$, so dass $u(X) := \sum_{e \in \delta_G(X)} u(e)$ minimal ist (siehe Aufgabe 31, Kapitel 8). Stellt sich $|T \cap X|$ als ungerade heraus, so ist man fertig (der Output ist X).

2. Andernfalls wende man den Algorithmus rekursiv an, zuerst auf G, u und $T \cap X$, dann auf G, u und $T \setminus X$. Man erhält eine Menge $Y \subseteq V(G)$ mit $|(T \cap X) \cap Y|$ ungerade und $u(Y)$ minimal und eine Menge $Z \subseteq V(G)$ mit $|(T \setminus X) \cap Z|$ ungerade und $u(Z)$ minimal. O. B. d. A. kann man annehmen, dass $T \setminus X \not\subseteq Y$ und $X \cap T \not\subseteq Z$ (sonst ersetze man Y durch $V(G) \setminus Y$ und/oder Z durch $V(G) \setminus Z$).

3. Ist $u(X \cap Y) < u(Z \setminus X)$, so ist der Output $X \cap Y$, sonst ist der Output $Z \setminus X$.

Man zeige, dass dieser Algorithmus korrekt arbeitet und $O(n^5)$ -Laufzeit hat, wobei $n = |V(G)|$.

21. Man zeige, wie man das MAXIMUM-WEIGHT-*b*-MATCHING-PROBLEM für den Fall, dass $b(v)$ für alle $v \in V(G)$ gerade ist, in streng polynomieller Zeit lösen kann.

Hinweis: Man führe das Problem wie in Aufgabe 10, Kapitel 9, auf ein MINIMUM-COST-FLOW-PROBLEM zurück.

Literatur

Allgemeine Literatur:

- Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., und Schrijver, A. [1998]: Combinatorial Optimization. Wiley, New York 1998, Abschnitte 5.4 und 5.5
- Frank, A. [1996]: A survey on T -joins, T -cuts, and conservative weightings. In: Combinatorics, Paul Erdős is Eighty; Volume 2 (D. Miklós, V.T. Sós, T. Szőnyi, Hrsg.), Bolyai Society, Budapest 1996, pp. 213–252
- Gerards, A.M.H. [1995]: Matching. In: Handbooks in Operations Research and Management Science; Volume 7: Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, Hrsg.), Elsevier, Amsterdam 1995, pp. 135–224
- Lovász, L., und Plummer, M.D. [1986]: Matching Theory. Akadémiai Kiadó, Budapest 1986, und North-Holland, Amsterdam 1986
- Schrijver, A. [1983]: Min-max results in combinatorial optimization; Abschnitt 6. In: Mathematical Programming; The State of the Art – Bonn 1982 (A. Bachem, M. Grötschel, B. Korte, Hrsg.), Springer, Berlin 1983, pp. 439–500
- Schrijver, A. [2003]: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin 2003, Kapitel 29–33

Zitierte Literatur:

- Anstee, R.P. [1987]: A polynomial algorithm for b -matchings: an alternative approach. Information Processing Letters 24 (1987), 153–157
- Babenko, M.A. und Karzanov, A.V. [2009]: Minimum mean cycle problem in bidirected and skew-symmetric graphs. Discrete Optimization 6 (2009), 92–97
- Barahona, F. [1993]: Reducing matching to polynomial size linear programming. SIAM Journal on Optimization 3 (1993), 688–695
- Caprara, A., und Fischetti, M. [1996]: $\{0, \frac{1}{2}\}$ -Chvátal-Gomory cuts. Mathematical Programming 74 (1996), 221–235
- Edmonds, J. [1965]: Maximum matching and a polyhedron with $(0,1)$ vertices. Journal of Research of the National Bureau of Standards B 69 (1965), 125–130
- Edmonds, J., und Johnson, E.L. [1970]: Matching: A well-solved class of integer linear programs. In: Combinatorial Structures and Their Applications; Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications 1969 (R. Guy, H. Hanani, N. Sauer, J. Schönheim, Hrsg.), Gordon and Breach, New York 1970, pp. 69–87
- Edmonds, J., und Johnson, E.L. [1973]: Matching, Euler tours and the Chinese postman. Mathematical Programming 5 (1973), 88–124
- Gabow, H.N. [1983]: An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. Proceedings of the 15th Annual ACM Symposium on Theory of Computing (1983), 448–456
- Goldberg, A.V., und Karzanov, A.V. [2004]: Maximum skew-symmetric flows and matchings. Mathematical Programming A 100 (2004), 537–568
- Guan, M. [1962]: Graphic programming using odd and even points. Chinese Mathematics 1 (1962), 273–277
- Hadlock, F. [1975]: Finding a maximum cut of a planar graph in polynomial time. SIAM Journal on Computing 4 (1975), 221–225

- Karzanov, A.V. [1985]: Minimum mean weight cuts and cycles in directed graphs. In: Kachestvennye i Priblizhennye Metody Issledovaniya Operatornykh Uravnenii (V.S. Klimov, Hrsg.), Yaroslavl State University Press, Yaroslavl 1985, pp. 72–83 [auf Russisch]. Englische Übersetzung: American Mathematical Society Translations Ser. 2, Vol. 158 (1994), 47–55
- Karzanov, A.V. [1986]: An algorithm for determining a maximum packing of odd-terminus cuts and its applications. In: Issledovaniya po Prikladnoi Teorii Grafov (A.S. Alekseev, Hrsg.), Nauka Siberian Dept., Novosibirsk, 1986, pp. 126–140 [auf Russisch]. Englische Übersetzung: American Mathematical Society Translations Ser. 2, Vol. 158 (1994), 57–70
- Letchford, A.N., Reinelt, G., und Theis, D.O. [2008]: Odd minimum cut sets and *b*-matchings revisited. SIAM Journal on Discrete Mathematics 22 (2008), 1480–1487
- Marsh, A.B. [1979]: Matching algorithms. Ph.D. thesis, Johns Hopkins University, Baltimore 1979
- Padberg, M.W., und Rao, M.R. [1982]: Odd minimum cut-sets and *b*-matchings. Mathematics of Operations Research 7 (1982), 67–80
- Pulleyblank, W.R. [1973]: Faces of matching polyhedra. Ph.D. thesis, University of Waterloo, 1973
- Pulleyblank, W.R. [1980]: Dual integrality in *b*-matching problems. Mathematical Programming Study 12 (1980), 176–196
- Rizzi, R. [2002]: Minimum *T*-cuts and optimal *T*-pairings. Discrete Mathematics 257 (2002), 177–181
- Rothvoß, T. [2017]: The matching polytope has exponential extension complexity. Journal of the ACM 64 (2017), Article 41
- Sebő, A. [1987]: A quick proof of Seymour's theorem on *T*-joins. Discrete Mathematics 64 (1987), 101–103
- Seymour, P.D. [1981]: On odd cuts and multicommodity flows. Proceedings of the London Mathematical Society (3) 42 (1981), 178–192
- Tutte, W.T. [1952]: The factors of graphs. Canadian Journal of Mathematics 4 (1952), 314–328
- Tutte, W.T. [1954]: A short proof of the factor theorem for finite graphs. Canadian Journal of Mathematics 6 (1954), 347–352
- Yannakakis, M. [1991]: Expressing combinatorial optimization problems by linear programs. Journal of Computer and System Sciences 43 (1991), 441–466



13 Matroide

Viele kombinatorische Optimierungsprobleme können folgendermaßen formuliert werden. Für ein gegebenes Mengensystem (E, \mathcal{F}) (d. h. eine endliche Menge E und eine Familie $\mathcal{F} \subseteq 2^E$) mit einer Kostenfunktion $c : \mathcal{F} \rightarrow \mathbb{R}$, bestimme man ein Element aus \mathcal{F} mit minimalen oder maximalen Kosten. Hier betrachten wir modulare Funktionen c , d. h. wir nehmen an, dass $c(X) = c(\emptyset) + \sum_{x \in X} (c(\{x\}) - c(\emptyset))$ für alle $X \subseteq E$; äquivalent formuliert, haben wir eine Funktion $c : E \rightarrow \mathbb{R}$ und schreiben $c(X) = \sum_{e \in X} c(e)$.

In diesem Kapitel beschränken wir uns auf diejenigen kombinatorischen Optimierungsprobleme, wo \mathcal{F} ein Unabhängigkeitssystem (d. h. \mathcal{F} ist abgeschlossen bezüglich Teilmengenbildung) oder sogar ein Matroid ist. Die Resultate dieses Kapitels verallgemeinern einige der früheren Resultate.

In Abschnitt 13.1 führen wir Unabhängigkeitssysteme und Matroide ein und zeigen, dass viele kombinatorische Optimierungsprobleme in diesem Kontext beschrieben werden können. Es gibt mehrere äquivalente Axiomensysteme für Matroide (Abschnitt 13.2) und eine interessante Dualitätsrelation (Abschnitt 13.3). Der Hauptgrund, weswegen Matroide wichtig sind, ist, dass ein einfacher Greedy-Algorithmus zur Optimierung in Matroiden benutzt werden kann. In Abschnitt 13.4 werden wir Greedy-Algorithmen analysieren und wenden uns dann der Optimierung in dem Schnitt zweier Matroide zu. Wie wir in den Abschnitten 13.5 und 13.7 zeigen werden, ist dieses Problem polynomiell lösbar. Damit wird auch das Problem der Überdeckung eines Matroids mit unabhängigen Mengen gelöst sein, wie wir in Abschnitt 13.6 besprechen werden.

13.1 Unabhängigkeitssysteme und Matroide

Definition 13.1. Ein Mengensystem (E, \mathcal{F}) ist ein **Unabhängigkeitssystem**, falls Folgendes gilt:

- (M1) $\emptyset \in \mathcal{F}$;
- (M2) Aus $X \subseteq Y \in \mathcal{F}$ folgt $X \in \mathcal{F}$.

Die Elemente von \mathcal{F} heißen **unabhängig** und die von $2^E \setminus \mathcal{F}$ heißen **abhängig**. Inkluisionsminimale abhängige Mengen heißen **Kreise**, inklusionsmaximale unabhängige Mengen heißen **Basen**. Für ein gegebenes $X \subseteq E$ heißen die inklusionsmaximalen unabhängigen Teilmengen von X die **Basen von X** .

Definition 13.2. Sei (E, \mathcal{F}) ein Unabhängigkeitssystem. Für $X \subseteq E$ definieren wir den **Rang** von X durch $r(X) := \max\{|Y| : Y \subseteq X, Y \in \mathcal{F}\}$. Ferner definieren wir den **Abschluss** von X durch $\sigma(X) := \{y \in E : r(X \cup \{y\}) = r(X)\}$.

In diesem Kapitel wird (E, \mathcal{F}) immer ein Unabhängigkeitssystem bezeichneten und $c : E \rightarrow \mathbb{R}$ eine Kostenfunktion. Wir konzentrieren uns auf die beiden folgenden Probleme:

MAXIMIERUNGSPROBLEM FÜR UNABHÄNGIGKEITSSYSTEME

Instanz: Ein Unabhängigkeitssystem (E, \mathcal{F}) und $c : E \rightarrow \mathbb{R}$.

Aufgabe: Bestimme ein $X \in \mathcal{F}$ mit $c(X) := \sum_{e \in X} c(e)$ maximal.

MINIMIERUNGSPROBLEM FÜR UNABHÄNGIGKEITSSYSTEME

Instanz: Ein Unabhängigkeitssystem (E, \mathcal{F}) und $c : E \rightarrow \mathbb{R}$.

Aufgabe: Bestimme eine Basis B mit $c(B)$ minimal.

Die Spezifizierung der Instanz ist etwas ungenau. Die Menge E und die Kostenfunktion c sind, wie immer, explizit gegeben. Die Familie \mathcal{F} wird jedoch normalerweise nicht in Form einer expliziten Liste seiner Elemente gegeben. Man setzt stattdessen ein Orakel voraus, mit dem man für ein gegebenes $F \subseteq E$ entscheidet, ob $F \in \mathcal{F}$. Wir werden uns mit dieser Frage in Abschnitt 13.4 weiter befassen.

Die folgende Liste zeigt, dass viele kombinatorische Optimierungsprobleme eine der obigen Formen haben:

(1) **MAXIMUM-WEIGHT-STABLE-SET-PROBLEM**

Gegeben sei ein Graph G und Gewichte $c : V(G) \rightarrow \mathbb{R}$. Bestimme eine stabile Menge X maximalen Gewichtes in G .

Hier ist $E = V(G)$ und $\mathcal{F} = \{F \subseteq E : F$ ist eine stabile Menge in $G\}$.

(2) **TSP**

Gegeben sei ein vollständiger ungerichteter Graph G und Gewichte $c : E(G) \rightarrow \mathbb{R}_+$. Bestimme einen Hamilton-Kreis minimalen Gewichtes in G .

Hier ist $E = E(G)$ und $\mathcal{F} = \{F \subseteq E : F$ ist eine Teilmenge der Kanten eines Hamilton-Kreises in $G\}$.

(3) **KÜRZESTE-WEGE-PROBLEM**

Gegeben sei ein Graph G (gerichtet oder ungerichtet), $c : E(G) \rightarrow \mathbb{R}$ und $s, t \in V(G)$ mit t erreichbar von s aus. Bestimme einen kürzesten $s-t$ -Weg in G bezüglich c .

Hier ist $E = E(G)$ und $\mathcal{F} = \{F \subseteq E : F$ ist eine Teilmenge der Kanten eines $s-t$ -Weges}.

(4) **KNAPSACK-PROBLEM**

Gegeben seien $n \in \mathbb{N}$ und nichtnegative Zahlen n, c_i, w_i ($1 \leq i \leq n$) und W . Bestimme eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} w_j \leq W$ und $\sum_{j \in S} c_j$ maximal.

Hier ist $E = \{1, \dots, n\}$ und $\mathcal{F} = \left\{F \subseteq E : \sum_{j \in F} w_j \leq W\right\}$.

(5) MINIMUM-SPANNING-TREE-PROBLEM

Gegeben sei ein zusammenhängender ungerichteter Graph G und Gewichte $c : E(G) \rightarrow \mathbb{R}$. Bestimme einen aufspannenden Baum minimalen Gewichtes in G .

Hier ist $E = E(G)$ und die Elemente von \mathcal{F} sind die Kantenmengen der Wälder in G .

(6) MAXIMUM-WEIGHT-FOREST-PROBLEM

Gegeben sei ein ungerichteter Graph G und Gewichte $c : E(G) \rightarrow \mathbb{R}$. Bestimme einen Wald maximalen Gewichtes in G .

Hier ist (E, \mathcal{F}) wie in (5).

(7) STEINERBAUM-PROBLEM

Gegeben sei ein zusammenhängender ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$ und eine Menge $T \subseteq V(G)$ von Terminalen. Bestimme einen Steinerbaum für T , d. h. einen Baum S mit $T \subseteq V(S)$ und $E(S) \subseteq E(G)$, so dass $c(E(S))$ minimal ist.

Hier ist $E = E(G)$ und \mathcal{F} enthält alle Teilmengen der Kantenmengen von Steinerbäumen für T .

(8) MAXIMUM-WEIGHT-BRANCHING-PROBLEM

Gegeben sei ein Digraph G und Gewichte $c : E(G) \rightarrow \mathbb{R}$. Bestimme ein Branching maximalen Gewichtes in G .

Hier ist $E = E(G)$ und \mathcal{F} enthält die Kantenmengen der Branchings in G .

(9) MAXIMUM-WEIGHT-MATCHING-PROBLEM

Gegeben sei ein ungerichteter Graph G und Gewichte $c : E(G) \rightarrow \mathbb{R}$. Bestimme ein Matching maximalen Gewichtes in G .

Hier ist $E = E(G)$ und \mathcal{F} ist die Menge der Matchings in G .

Diese Liste enthält sowohl *NP*-schwere Probleme ((1),(2),(4),(7)), als auch polynomiell lösbar Probleme ((5),(6),(8),(9)). Das Problem (3) ist in obiger Form *NP*-schwer, aber für nichtnegative Gewichte polynomiell lösbar. (Der Begriff „*NP*-schwer“ wird in Kapitel 15 eingeführt.)

Definition 13.3. Ein Unabhängigkeitssystem ist ein **Matroid**, falls Folgendes gilt:

(M3) Sind $X, Y \in \mathcal{F}$ mit $|X| > |Y|$, so gibt es ein $x \in X \setminus Y$ mit $Y \cup \{x\} \in \mathcal{F}$.

Der Name Matroid deutet an, dass es sich hier um eine Verallgemeinerung des Matrixbegriffes handelt. Dies wird bereits am ersten der folgenden Beispiele klar:

Proposition 13.4. Die folgenden Unabhängigkeitssysteme (E, \mathcal{F}) sind Matroide:

- (a) E ist die Spaltenmenge einer Matrix A über irgendeinem Körper und $\mathcal{F} := \{F \subseteq E : \text{Die Spalten in } F \text{ sind linear unabhängig bezüglich dieses Körpers}\}$.
- (b) E ist die Kantenmenge eines ungerichteten Graphen G und $\mathcal{F} := \{F \subseteq E : (V(G), F) \text{ ist ein Wald}\}$.
- (c) E ist eine endliche Menge, k eine nichtnegative ganze Zahl und $\mathcal{F} := \{F \subseteq E : |F| \leq k\}$.

- (d) *E ist die Kantenmenge eines ungerichteten Graphen G, S ist eine stabile Menge in G, $k_s \in \mathbb{Z}_+$ ($s \in S$) und $\mathcal{F} := \{F \subseteq E : |\delta_F(s)| \leq k_s \text{ für alle } s \in S\}$.*
- (e) *E ist die Kantenmenge eines Digraphen G, $S \subseteq V(G)$, $k_s \in \mathbb{Z}_+$ ($s \in S$) und $\mathcal{F} := \{F \subseteq E : |\delta_F^-(s)| \leq k_s \text{ für alle } s \in S\}$.*

Beweis: In allen fünf Fällen ist es klar, dass (E, \mathcal{F}) ein Unabhängigkeitssystem ist. Somit bleibt zu zeigen, dass (M3) gilt. Für (a) ist dies eine bekannte Tatsache der linearen Algebra und für (c) ist es trivial.

Um (M3) für (b) zu beweisen, seien $X, Y \in \mathcal{F}$. Angenommen, es wäre $Y \cup \{x\} \notin \mathcal{F}$ für alle $x \in X \setminus Y$. Wir werden zeigen, dass daraus der Widerspruch $|X| \leq |Y|$ folgt. Für jede Kante $x = \{v, w\} \in X$ sind v und w in derselben Zusammenhangskomponente von $(V(G), Y)$. Also ist jede Zusammenhangskomponente $Z \subseteq V(G)$ von $(V(G), X)$ eine Teilmenge einer Zusammenhangskomponente von $(V(G), Y)$. Somit ist die Anzahl p der Zusammenhangskomponenten des Waldes $(V(G), X)$ nicht kleiner als die Anzahl q der Zusammenhangskomponenten des Waldes $(V(G), Y)$. Damit folgt dann $|V(G)| - |X| = p \geq q = |V(G)| - |Y|$ oder $|X| \leq |Y|$.

Um (M3) für (d) zu beweisen, seien $X, Y \in \mathcal{F}$ mit $|X| > |Y|$. Sei $S' := \{s \in S : |\delta_Y(s)| = k_s\}$. Da $|X| > |Y|$ und $|\delta_X(s)| \leq k_s$ für alle $s \in S'$, gibt es ein $e \in X \setminus Y$ mit $e \notin \delta(s)$ für $s \in S'$. Somit ist $Y \cup \{e\} \in \mathcal{F}$.

Für (e) gilt derselbe Beweis bis auf das Ersetzen von δ durch δ^- . \square

Manche dieser Matroide haben besondere Namen: Das Matroid in (a) heißt das **Vektormatroid** von A. Sei \mathcal{M} ein Matroid. Gibt es eine Matrix A über dem Körper F, so dass \mathcal{M} das Vektormatroid von A ist, so heißt \mathcal{M} **repräsentierbar über F**. Es gibt Matroide, die über keinem Körper repräsentierbar sind.

Das Matroid in (b) heißt das **Kreismatroid** von G und wird gelegentlich mit $\mathcal{M}(G)$ bezeichnet. Ein Matroid, welches ein Kreismatroid eines Graphen ist, der auch Schleifen enthalten darf, heißt **graphisches Matroid**.

Die Matroide in (c) heißen **uniform**.

In unserer Liste von Unabhängigkeitssystemen am Anfang dieses Abschnitts kommen nur zwei Matroide vor, nämlich graphische Matroide in (5) und (6). Um zu zeigen, dass alle anderen Unabhängigkeitssysteme in obiger Liste im Allgemeinen keine Matroide sind, bedienen wir uns des folgenden Satzes (Aufgabe 1):

Satz 13.5. *Sei (E, \mathcal{F}) ein Unabhängigkeitssystem. Dann sind die folgenden drei Aussagen äquivalent:*

- (M3) *Sind $X, Y \in \mathcal{F}$ mit $|X| > |Y|$, so gibt es ein $x \in X \setminus Y$ mit $Y \cup \{x\} \in \mathcal{F}$.*
 (M3') *Sind $X, Y \in \mathcal{F}$ mit $|X| = |Y| + 1$, so gibt es ein $x \in X \setminus Y$ mit $Y \cup \{x\} \in \mathcal{F}$.*
 (M3'') *Für jedes $X \subseteq E$ haben sämtliche Basen von X dieselbe Kardinalität.*

Beweis: Trivialerweise gelten $(M3) \Leftrightarrow (M3')$ und $(M3) \Rightarrow (M3'')$. Um $(M3'') \Rightarrow (M3)$ zu beweisen, seien $X, Y \in \mathcal{F}$ mit $|X| > |Y|$. Nach $(M3'')$ ist Y keine Basis von $X \cup Y$. Somit gibt es ein $x \in (X \cup Y) \setminus Y = X \setminus Y$ mit $Y \cup \{x\} \in \mathcal{F}$. \square

Gelegentlich ist es nützlich, eine zweite Rangfunktion zu haben:

Definition 13.6. Sei (E, \mathcal{F}) ein Unabhängigkeitssystem. Für $X \subseteq E$ definieren wir die **untere Rangfunktion** durch

$$\rho(X) := \min\{|Y| : Y \subseteq X, Y \in \mathcal{F} \text{ und } Y \cup \{x\} \notin \mathcal{F} \text{ für alle } x \in X \setminus Y\}.$$

Der **Rangquotient** von (E, \mathcal{F}) wird definiert durch

$$q(E, \mathcal{F}) := \min_{F \subseteq E} \frac{\rho(F)}{r(F)}.$$

Proposition 13.7. Sei (E, \mathcal{F}) ein Unabhängigkeitssystem. Dann ist $q(E, \mathcal{F}) \leq 1$. Ferner gilt: (E, \mathcal{F}) ist genau dann ein Matroid, wenn $q(E, \mathcal{F}) = 1$.

Beweis: Es folgt $q(E, \mathcal{F}) \leq 1$ aus der Definition. Ferner ist $q(E, \mathcal{F}) = 1$ offensichtlich mit (M3'') äquivalent. \square

Um den Rangquotienten abzuschätzen, kann man den folgenden Satz heranziehen:

Satz 13.8. (Hausmann, Jenkyns und Korte [1980]) Sei (E, \mathcal{F}) ein Unabhängigkeitssystem. Gilt für alle $A \in \mathcal{F}$ und $e \in E$, dass $A \cup \{e\}$ höchstens p Kreise enthält, so folgt $q(E, \mathcal{F}) \geq \frac{1}{p}$.

Beweis: Sei $F \subseteq E$ und seien J, K zwei Basen von F . Wir werden zeigen, dass $\frac{|J|}{|K|} \geq \frac{1}{p}$.

Sei $J \setminus K = \{e_1, \dots, e_t\}$. Wir konstruieren eine Folge $K = K_0, K_1, \dots, K_t$ unabhängiger Teilmengen von $J \cup K$ mit $J \cap K \subseteq K_i$, $K_i \cap \{e_1, \dots, e_t\} = \{e_1, \dots, e_i\}$ und $|K_{i-1} \setminus K_i| \leq p$ für $i = 1, \dots, t$.

Da $K_i \cup \{e_{i+1}\}$ höchstens p Kreise enthält und jeder dieser Kreise nichtleeren Durchschnitt mit $K_i \setminus J$ hat (weil J unabhängig ist), gibt es ein $X \subseteq K_i \setminus J$ mit $|X| \leq p$ und $(K_i \setminus X) \cup \{e_{i+1}\} \in \mathcal{F}$. Setze $K_{i+1} := (K_i \setminus X) \cup \{e_{i+1}\}$.

Nun ist $J \subseteq K_t \in \mathcal{F}$. Da J eine Basis von F ist, folgt $J = K_t$. Damit haben wir

$$|K \setminus J| = \sum_{i=1}^t |K_{i-1} \setminus K_i| \leq pt = p |J \setminus K|,$$

womit $|K| \leq p |J|$ folgt. \square

Mit diesem Satz folgt $q(E, \mathcal{F}) \geq \frac{1}{2}$ für Beispiel (9) (siehe auch Aufgabe 1, Kapitel 10). Es gilt in der Tat: $q(E, \mathcal{F}) = \frac{1}{2}$ genau dann, wenn G einen Weg der Länge 3 als Teilgraph enthält (sonst ist $q(E, \mathcal{F}) = 1$). Für Beispiel (1) unserer Liste kann der Rangquotient beliebig klein werden (es sei z. B. G ein Stern). In Aufgabe 6 werden die Rangquotienten für weitere Unabhängigkeitssysteme besprochen.

13.2 Andere Matroidaxiome

In diesem Abschnitt werden wir andere Axiomensysteme zur Definition von Matroiden betrachten. Sie charakterisieren fundamentale Eigenschaften der Familie der Basen, der Rangfunktion, des Abschlussoperators und der Familie der Kreise eines Matroids.

Satz 13.9. *Sei E eine endliche Menge und $\mathcal{B} \subseteq 2^E$. Die Familie \mathcal{B} ist genau dann die Menge der Basen eines Matroids (E, \mathcal{F}) , wenn die folgenden beiden Bedingungen erfüllt sind:*

- (B1) $\mathcal{B} \neq \emptyset$;
- (B2) Für alle $B_1, B_2 \in \mathcal{B}$ und $x \in B_1 \setminus B_2$ gibt es ein $y \in B_2 \setminus B_1$ mit $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$.

Beweis: Die Menge der Basen eines Matroids erfüllt die Bedingungen (B1) (nach (M1)) und (B2): Für Basen B_1, B_2 und $x \in B_1 \setminus B_2$ folgt, dass $B_1 \setminus \{x\}$ unabhängig ist. Nach (M3) gibt es ein $y \in B_2 \setminus B_1$, so dass $(B_1 \setminus \{x\}) \cup \{y\}$ unabhängig ist. Letztere Menge ist sogar eine Basis, da alle Basen eines Matroids dieselbe Kardinalität haben.

Zur umgekehrten Richtung, erfülle \mathcal{B} die Bedingungen (B1) und (B2). Als erstes zeigen wir, dass alle Elemente aus \mathcal{B} dieselbe Kardinalität haben: Seien $B_1, B_2 \in \mathcal{B}$ mit $|B_1| > |B_2|$ und der Eigenschaft, dass $B_1 \cap B_2$ maximale Kardinalität hat, d.h., dass $|B_1 \cap B_2|$ maximal ist. Sei $x \in B_1 \setminus B_2$. Nach (B2) gibt es $y \in B_2 \setminus B_1$ mit $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$, im Widerspruch zur Maximalität von $|B_1 \cap B_2|$.

Nun sei

$$\mathcal{F} := \{F \subseteq E : \text{es gibt ein } B \in \mathcal{B} \text{ mit } F \subseteq B\}.$$

Es ist (E, \mathcal{F}) ein Unabhängigkeitssystem und \mathcal{B} die Familie seiner Basen. Um zu zeigen, dass (E, \mathcal{F}) die Bedingung (M3) erfüllt, seien $X, Y \in \mathcal{F}$ mit $|X| > |Y|$. Seien $X \subseteq B_1 \in \mathcal{B}$ und $Y \subseteq B_2 \in \mathcal{B}$, wobei B_1 und B_2 so gewählt wurden, dass $|B_1 \cap B_2|$ maximal ist. Ist $B_2 \cap (X \setminus Y) \neq \emptyset$, so sind wir fertig, da wir Y augmentieren können.

Der andere Fall, nämlich $B_2 \cap (X \setminus Y) = \emptyset$, ist unmöglich: Träfe er zu, so folgt

$$|B_1 \cap B_2| + |Y \setminus B_1| + |(B_2 \setminus B_1) \setminus Y| = |B_2| = |B_1| \geq |B_1 \cap B_2| + |X \setminus Y|.$$

Da $|X \setminus Y| > |Y \setminus X| \geq |Y \setminus B_1|$, folgt $(B_2 \setminus B_1) \setminus Y \neq \emptyset$. Sei also $y \in (B_2 \setminus B_1) \setminus Y$. Nach (B2) gibt es ein $x \in B_1 \setminus B_2$ mit $(B_2 \setminus \{y\}) \cup \{x\} \in \mathcal{B}$, im Widerspruch zur Maximalität von $|B_1 \cap B_2|$. \square

In Aufgabe 8 findet sich eine ähnliche Aussage. Eine besonders wichtige Eigenschaft von Matroiden ist die Tatsache, dass die Rangfunktion submodular ist:

Satz 13.10. Sei E eine endliche Menge und $r : 2^E \rightarrow \mathbb{Z}_+$. Dann sind die folgenden drei Aussagen äquivalent:

- (a) Es ist r die Rangfunktion eines Matroids (E, \mathcal{F}) (und $\mathcal{F} = \{F \subseteq E : r(F) = |F|\}$).
- (b) Für alle $X, Y \subseteq E$ gilt:
 - (R1) $r(X) \leq |X|$;
 - (R2) Ist $X \subseteq Y$, so gilt $r(X) \leq r(Y)$;
 - (R3) $r(X \cup Y) + r(X \cap Y) \leq r(X) + r(Y)$.
- (c) Für alle $X \subseteq E$ und $x, y \in E$ gilt:
 - (R1') $r(\emptyset) = 0$;
 - (R2') $r(X) \leq r(X \cup \{y\}) \leq r(X) + 1$;
 - (R3') Ist $r(X \cup \{x\}) = r(X \cup \{y\}) = r(X)$, so gilt $r(X \cup \{x, y\}) = r(X)$.

Beweis: (a) \Rightarrow (b): Ist r eine Rangfunktion eines Unabhängigkeitssystems (E, \mathcal{F}) , so gelten (R1) und (R2) offensichtlich. Ist (E, \mathcal{F}) ein Matroid, so gilt auch (R3): Seien $X, Y \subseteq E$ und sei A eine Basis von $X \cap Y$. Nach (M3) kann A zu einer Basis $A \cup B$ von X und zu einer Basis $(A \cup B) \cup C$ von $X \cup Y$ erweitert werden. Dann ist $A \cup C$ eine unabhängige Teilmenge von Y und es folgt:

$$\begin{aligned} r(X) + r(Y) &\geq |A \cup B| + |A \cup C| \\ &= 2|A| + |B| + |C| \\ &= |A \cup B \cup C| + |A| \\ &= r(X \cup Y) + r(X \cap Y). \end{aligned}$$

(b) \Rightarrow (c): Aus (R1) folgt (R1'). Aus (R2) folgt $r(X) \leq r(X \cup \{y\})$. Aus (R3) und (R1) folgt:

$$r(X \cup \{y\}) \leq r(X) + r(\{y\}) - r(X \cap \{y\}) \leq r(X) + r(\{y\}) \leq r(X) + 1,$$

womit (R2') bewiesen ist.

Es ist (R3') trivial für $x = y$. Für $x \neq y$ folgt mit (R2) und (R3):

$$2r(X) \leq r(X) + r(X \cup \{x, y\}) \leq r(X \cup \{x\}) + r(X \cup \{y\}),$$

womit (R3') bewiesen ist.

(c) \Rightarrow (a): Sei $r : 2^E \rightarrow \mathbb{Z}_+$ eine die Bedingungen (R1')–(R3') erfüllende Funktion. Sei ferner

$$\mathcal{F} := \{F \subseteq E : r(F) = |F|\}.$$

Wir behaupten, dass (E, \mathcal{F}) ein Matroid ist. Aus (R1') folgt (M1). Aus (R2') folgt $r(X) \leq |X|$ für alle $X \subseteq E$. Für $Y \in \mathcal{F}$, $y \in Y$ und $X := Y \setminus \{y\}$ folgt

$$|X| + 1 = |Y| = r(Y) = r(X \cup \{y\}) \leq r(X) + 1 \leq |X| + 1,$$

somit ist $X \in \mathcal{F}$. Damit folgt (M2).

Nun seien $X, Y \in \mathcal{F}$ mit $|X| = |Y| + 1$. Sei $X \setminus Y = \{x_1, \dots, x_k\}$. Angenommen, (M3') wäre verletzt, d. h. $r(Y \cup \{x_i\}) = |Y|$ für $i = 1, \dots, k$. Nach (R3') folgt dann $r(Y \cup \{x_1, x_i\}) = r(Y)$ für $i = 2, \dots, k$. Wiederholte Anwendung dieses Schrittes ergibt $r(Y) = r(Y \cup \{x_1, \dots, x_k\}) = r(X \cup Y) \geq r(X)$, ein Widerspruch.

Somit ist (E, \mathcal{F}) in der Tat ein Matroid. Um zu zeigen, dass r die Rangfunktion dieses Matroids ist, müssen wir beweisen, dass $r(X) = \max\{|Y| : Y \subseteq X, r(Y) = |Y|\}$ für alle $X \subseteq E$. Sei also $X \subseteq E$, und sei Y eine kardinalitätsmaximale Teilmenge von X mit $r(Y) = |Y|$. Für alle $x \in X \setminus Y$ haben wir $r(Y \cup \{x\}) < |Y| + 1$, nach (R2') folgt somit $r(Y \cup \{x\}) = |Y|$. Mittels wiederholter Anwendung von (R3') folgt $r(X) = |Y|$. \square

Satz 13.11. *Sei E eine endliche Menge und $\sigma : 2^E \rightarrow 2^E$ eine Funktion. Es ist σ genau dann der Abschlussoperator eines Matroids (E, \mathcal{F}) , wenn die folgenden vier Bedingungen für alle $X, Y \subseteq E$ und $x, y \in E$ gelten:*

- (S1) $X \subseteq \sigma(X)$;
- (S2) Ist $X \subseteq Y \subseteq E$, so gilt $\sigma(X) \subseteq \sigma(Y)$;
- (S3) $\sigma(X) = \sigma(\sigma(X))$;
- (S4) Ist $y \notin \sigma(X)$ und $y \in \sigma(X \cup \{x\})$, so gilt $x \in \sigma(X \cup \{y\})$.

Beweis: Ist σ der Abschlussoperator eines Matroids, so ist (S1) trivial.

Für $X \subseteq Y$ und $z \in \sigma(X)$ haben wir mit (R3) und (R2):

$$\begin{aligned} r(X) + r(Y) &= r(X \cup \{z\}) + r(Y) \\ &\geq r((X \cup \{z\}) \cap Y) + r(X \cup \{z\} \cup Y) \\ &\geq r(X) + r(Y \cup \{z\}), \end{aligned}$$

also ist $z \in \sigma(Y)$, womit (S2) folgt.

Mittels wiederholter Anwendung von (R3') haben wir $r(\sigma(X)) = r(X)$ für alle X , womit (S3) folgt.

Um (S4) zu beweisen, nehme man an, es gäbe X, x, y mit $y \notin \sigma(X)$, $y \in \sigma(X \cup \{x\})$ und $x \notin \sigma(X \cup \{y\})$. Dann folgt $r(X \cup \{y\}) = r(X) + 1$, $r(X \cup \{x, y\}) = r(X \cup \{x\})$ und $r(X \cup \{x, y\}) = r(X \cup \{y\}) + 1$. Also ist $r(X \cup \{x\}) = r(X) + 2$, im Widerspruch zu (R2').

Zum Beweis der umgekehrten Richtung, sei $\sigma : 2^E \rightarrow 2^E$ eine die Bedingungen (S1)–(S4) erfüllende Funktion. Sei ferner

$$\mathcal{F} := \{X \subseteq E : x \notin \sigma(X \setminus \{x\}) \text{ für alle } x \in X\}.$$

Wir behaupten, dass (E, \mathcal{F}) ein Matroid ist.

Es ist (M1) trivial. Für $X \subseteq Y \in \mathcal{F}$ und $x \in X$ folgt $x \notin \sigma(Y \setminus \{x\}) \supseteq \sigma(X \setminus \{x\})$, somit ist $X \in \mathcal{F}$ und es folgt (M2). Zum Beweis von (M3) benötigen wir die folgende Aussage:

Behauptung: Für $X \in \mathcal{F}$ und $Y \subseteq E$ mit $|X| > |Y|$ folgt $X \not\subseteq \sigma(Y)$.

Diese Behauptung beweisen wir mittels Induktion über $|Y \setminus X|$. Ist $|Y \setminus X| = 0$, d. h. $Y \subseteq X$, so sei $x \in X \setminus Y$. Da $X \in \mathcal{F}$, haben wir $x \notin \sigma(X \setminus \{x\}) \supseteq \sigma(Y)$ nach (S2). Damit folgt $x \in X \setminus \sigma(Y)$, wie erwünscht.

Ist $|Y \setminus X| > 0$, so sei $y \in Y \setminus X$. Nach der Induktionsvoraussetzung gibt es ein $x \in X \setminus \sigma(Y \setminus \{y\})$. Ist $x \notin \sigma(Y)$, so sind wir fertig. Im gegenteiligen Fall haben wir $x \in \sigma(Y \setminus \{y\})$, aber $x \in \sigma(Y) = \sigma((Y \setminus \{y\}) \cup \{y\})$, somit folgt nach (S4), dass $y \in \sigma((Y \setminus \{y\}) \cup \{x\})$. Mit (S1) folgt $Y \subseteq \sigma((Y \setminus \{y\}) \cup \{x\})$, und mit (S2) und (S3) folgt daraus $\sigma(Y) \subseteq \sigma((Y \setminus \{y\}) \cup \{x\})$. Mit der Induktionsvoraussetzung angewendet auf X und $(Y \setminus \{y\}) \cup \{x\}$ (beachte, dass $x \neq y$) folgt $X \not\subseteq \sigma((Y \setminus \{y\}) \cup \{x\})$, also gilt $X \not\subseteq \sigma(Y)$ wie erwünscht.

Damit ist die Behauptung bewiesen. Nun können wir (M3) leicht verifizieren. Seien $X, Y \in \mathcal{F}$ mit $|X| > |Y|$. Nach der Behauptung gibt es ein $x \in X \setminus \sigma(Y)$. Für jedes $z \in Y \cup \{x\}$ haben wir nun $z \notin \sigma(Y \setminus \{z\})$, denn es gilt $Y \in \mathcal{F}$ und $x \notin \sigma(Y) = \sigma(Y \setminus \{x\})$. Nach (S4) folgt aus $z \notin \sigma(Y \setminus \{z\})$ und $x \notin \sigma(Y)$, dass $z \notin \sigma((Y \setminus \{z\}) \cup \{x\}) \supseteq \sigma((Y \cup \{x\}) \setminus \{z\})$. Somit ist $Y \cup \{x\} \in \mathcal{F}$.

Also ist (M3) in der Tat erfüllt und (E, \mathcal{F}) ist ein Matroid, etwa mit der Rangfunktion r und dem Abschlussoperator σ' . Es bleibt zu zeigen, dass $\sigma = \sigma'$.

Definitionsgemäß gilt $\sigma'(X) = \{y \in E : r(X \cup \{y\}) = r(X)\}$ und

$$r(X) = \max\{|Y| : Y \subseteq X, y \notin \sigma(Y \setminus \{y\}) \text{ für alle } y \in Y\}$$

für alle $X \subseteq E$.

Sei $X \subseteq E$. Um $\sigma'(X) \subseteq \sigma(X)$ zu beweisen, sei $z \in \sigma'(X) \setminus X$. Sei Y eine Basis von X . Da $r(Y \cup \{z\}) \leq r(X \cup \{z\}) = r(X) = |Y| < |Y \cup \{z\}|$, haben wir $y \in \sigma((Y \cup \{z\}) \setminus \{y\})$ für irgendein $y \in Y \cup \{z\}$. Ist $y = z$, so folgt $z \in \sigma(Y)$. Im gegenteiligen Fall folgt auch $z \in \sigma(Y)$, nämlich mit (S4) und da $y \notin \sigma(Y \setminus \{y\})$. Mit (S2) haben wir dann $z \in \sigma(X)$. Mit (S1) folgt hieraus $\sigma'(X) \subseteq \sigma(X)$.

Nun sei $z \notin \sigma'(X)$, d.h. $r(X \cup \{z\}) > r(X)$. Ist Y eine Basis von $X \cup \{z\}$, so folgt $z \in Y$ und $|Y \setminus \{z\}| = |Y| - 1 = r(X \cup \{z\}) - 1 = r(X)$. Somit ist $Y \setminus \{z\}$ eine Basis von X , woraus $X \subseteq \sigma'(Y \setminus \{z\}) \subseteq \sigma(Y \setminus \{z\})$ folgt, also gilt $\sigma(X) \subseteq \sigma(Y \setminus \{z\})$. Da $z \notin \sigma(Y \setminus \{z\})$, haben wir $z \notin \sigma(X)$. \square

Satz 13.12. Sei E eine endliche Menge und $\mathcal{C} \subseteq 2^E$. Die Familie \mathcal{C} ist genau dann die Menge der Kreise eines Unabhängigkeitssystems (E, \mathcal{F}) , wobei $\mathcal{F} = \{F \subset E : \text{es gibt kein } C \in \mathcal{C} \text{ mit } C \subseteq F\}$, wenn die beiden folgenden Bedingungen gelten:

- (C1) $\emptyset \notin \mathcal{C}$;
- (C2) Für alle $C_1, C_2 \in \mathcal{C}$ mit $C_1 \subseteq C_2$ gilt $C_1 = C_2$.

Ferner gilt: Ist \mathcal{C} die Menge der Kreise eines Unabhängigkeitssystems (E, \mathcal{F}) , so sind die folgenden vier Aussagen äquivalent:

- (a) (E, \mathcal{F}) ist ein Matroid.
- (b) Für alle $X \in \mathcal{F}$ und $e \in E$ enthält $X \cup \{e\}$ höchstens einen Kreis.
- (C3) Für alle $C_1, C_2 \in \mathcal{C}$ mit $C_1 \neq C_2$ und alle $e \in C_1 \cap C_2$ gibt es ein $C_3 \in \mathcal{C}$ mit $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$.
- (C3') Für alle $C_1, C_2 \in \mathcal{C}$, $e \in C_1 \cap C_2$ und $f \in C_1 \setminus C_2$ gibt es ein $C_3 \in \mathcal{C}$ mit $f \in C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$.

Beweis: Nach Definition erfüllt die Familie der Kreise eines Unabhängigkeitsystems die Bedingungen (C1) und (C2). Erfüllt \mathcal{C} die Bedingung (C1), so ist (E, \mathcal{F}) ein Unabhängigkeitssystem. Erfüllt \mathcal{C} auch (C2), so ist \mathcal{C} die Menge der Kreise dieses Unabhängigkeitssystems.

(a) \Rightarrow (C3'): Sei \mathcal{C} die Familie der Kreise eines Matroids und seien $C_1, C_2 \in \mathcal{C}$, $e \in C_1 \cap C_2$ und $f \in C_1 \setminus C_2$. Mit zweimaliger Anwendung von (R3) folgt:

$$\begin{aligned} & |C_1| - 1 + r((C_1 \cup C_2) \setminus \{e, f\}) + |C_2| - 1 \\ = & r(C_1) + r((C_1 \cup C_2) \setminus \{e, f\}) + r(C_2) \\ \geq & r(C_1) + r((C_1 \cup C_2) \setminus \{f\}) + r(C_2 \setminus \{e\}) \\ \geq & r(C_1 \setminus \{f\}) + r(C_1 \cup C_2) + r(C_2 \setminus \{e\}) \\ = & |C_1| - 1 + r(C_1 \cup C_2) + |C_2| - 1. \end{aligned}$$

Somit ist $r((C_1 \cup C_2) \setminus \{e, f\}) = r(C_1 \cup C_2)$. Sei B eine Basis von $(C_1 \cup C_2) \setminus \{e, f\}$. Dann enthält $B \cup \{f\}$ einen Kreis C_3 mit $f \in C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$, wie erwünscht.

(C3') \Rightarrow (C3): Dies ist trivial.

(C3) \Rightarrow (b): Ist $X \in \mathcal{F}$ und enthält $X \cup \{e\}$ zwei Kreise C_1, C_2 , so folgt aus (C3), dass $(C_1 \cup C_2) \setminus \{e\} \notin \mathcal{F}$. Es ist $(C_1 \cup C_2) \setminus \{e\}$ jedoch eine Teilmenge von X .

(b) \Rightarrow (a): Dies folgt aus Satz 13.8 und Proposition 13.7. \square

Die Eigenschaft (b) werden wir besonders häufig benutzen. Ist $X \in \mathcal{F}$ und $e \in E$ mit $X \cup \{e\} \notin \mathcal{F}$, so bezeichnen wir mit $C(X, e)$ den eindeutig bestimmten Kreis in $X \cup \{e\}$. Ist $X \cup \{e\} \in \mathcal{F}$, so setzen wir $C(X, e) := \emptyset$.

13.3 Dualität

Ein weiterer grundlegender Begriff in der Theorie der Matroide ist die Dualität.

Definition 13.13. Sei (E, \mathcal{F}) ein Unabhängigkeitssystem. Das **Dual** von (E, \mathcal{F}) ist das Mengensystem (E, \mathcal{F}^*) mit

$$\mathcal{F}^* = \{F \subseteq E : \text{es gibt eine Basis } B \text{ von } (E, \mathcal{F}) \text{ mit } F \cap B = \emptyset\}.$$

Offensichtlich ist das Dual eines Unabhängigkeitssystems wieder ein Unabhängigkeitssystem.

Proposition 13.14. $(E, \mathcal{F}^{**}) = (E, \mathcal{F})$.

Beweis: Es gilt: $F \in \mathcal{F}^{**} \Leftrightarrow$ es gibt eine Basis B^* von (E, \mathcal{F}^*) mit $F \cap B^* = \emptyset \Leftrightarrow$ es gibt eine Basis B von (E, \mathcal{F}) mit $F \cap (E \setminus B) = \emptyset \Leftrightarrow F \in \mathcal{F}$. \square

Satz 13.15. Sei (E, \mathcal{F}) ein Unabhängigkeitssystem, (E, \mathcal{F}^*) sein Dual, und seien r und r^* die entsprechenden Rangfunktionen. Dann gilt:

- (a) Es ist (E, \mathcal{F}) ein Matroid genau dann, wenn (E, \mathcal{F}^*) ein Matroid ist. (Whitney [1935])
 (b) Ist (E, \mathcal{F}) ein Matroid, so gilt $r^*(F) = |F| + r(E \setminus F) - r(E)$ für $F \subseteq E$.

Beweis: Nach Proposition 13.14 brauchen wir (a) nur in einer Richtung zu beweisen. Sei (E, \mathcal{F}) ein Matroid. Wir definieren $q : 2^E \rightarrow \mathbb{Z}_+$ durch $q(F) := |F| + r(E \setminus F) - r(E)$ und behaupten, dass q die Eigenschaften (R1), (R2) und (R3) besitzt. Mit dieser Behauptung und Satz 13.10 folgt dann, dass q die Rangfunktion eines Matroids ist. Offensichtlich gilt: $q(F) = |F|$ genau dann, wenn $F \in \mathcal{F}^*$, woraus $q = r^*$ folgt. Damit sind (a) und (b) bewiesen.

Nun beweisen wir die obige Behauptung. Es hat r die Eigenschaft (R2), also hat q die Eigenschaft (R1). Um zu zeigen, dass q die Eigenschaft (R2) hat, sei $X \subseteq Y \subseteq E$. Da (E, \mathcal{F}) ein Matroid ist, hat r die Eigenschaft (R3), somit gilt

$$r(E \setminus X) + 0 = r((E \setminus Y) \cup (Y \setminus X)) + r(\emptyset) \leq r(E \setminus Y) + r(Y \setminus X).$$

Daraus folgt (da r die Eigenschaft (R1) hat)

$$r(E \setminus X) - r(E \setminus Y) \leq r(Y \setminus X) \leq |Y \setminus X| = |Y| - |X|,$$

also gilt $q(X) \leq q(Y)$.

Es bleibt zu zeigen, dass q die Eigenschaft (R3) hat. Sei $X, Y \subseteq E$. Da r die Eigenschaft (R3) hat, folgt

$$\begin{aligned} & q(X \cup Y) + q(X \cap Y) \\ &= |X \cup Y| + |X \cap Y| + r(E \setminus (X \cup Y)) + r(E \setminus (X \cap Y)) - 2r(E) \\ &= |X| + |Y| + r((E \setminus X) \cap (E \setminus Y)) + r((E \setminus X) \cup (E \setminus Y)) - 2r(E) \\ &\leq |X| + |Y| + r(E \setminus X) + r(E \setminus Y) - 2r(E) \\ &= q(X) + q(Y). \end{aligned}$$

□

Oben haben wir das Kreismatroid $\mathcal{M}(G)$ eines beliebigen Graphen G eingeführt und dieses Matroid hat natürlich ein Dual. Für einen eingebetteten planaren Graphen G haben wir ein planares Dual G^* (welches im Allgemeinen von der Einbettung von G abhängt). Interessanterweise stimmen diese beiden Dualitätsbegriffe überein:

Satz 13.16. Sei G ein zusammenhängender planarer Graph mit einer beliebigen planaren Einbettung und G^* das planare Dual von G . Dann gilt

$$\mathcal{M}(G^*) = (\mathcal{M}(G))^*.$$

(Formal sind diese zwei Matroide isomorph: Sie sind bis auf Umbenennung der Elemente der Grundmenge identisch.)

Beweis: Für $T \subseteq E(G)$ setzen wir $\bar{T}^* := \{e^* : e \in E(G) \setminus T\}$, wobei e^* die der Kante e entsprechende duale Kante ist. Wir müssen beweisen:

Behauptung: Es ist T die Kantenmenge eines aufspannenden Baumes in G genau dann, wenn \overline{T}^* die Kantenmenge eines aufspannenden Baumes in G^* ist.

Da $(G^*)^* = G$ (nach Proposition 2.42) und $(\overline{T}^*)^* = T$ ist, brauchen wir nur eine Richtung der Behauptung zu beweisen.

Sei also $T \subseteq E(G)$, wobei \overline{T}^* die Kantenmenge eines aufspannenden Baumes in G^* ist. Erstens: Der Graph $(V(G), T)$ ist zusammenhängend, denn sonst würde eine Zusammenhangskomponente einen Schnitt definieren, dessen Dual einen Kreis in \overline{T}^* enthält (Satz 2.43). Zweitens: $(V(G), T)$ enthält keinen Kreis, denn sonst wäre die duale Kantenmenge ein Schnitt und $(V(G^*), \overline{T}^*)$ unzusammenhängend. Somit ist $(V(G), T)$ ein aufspannender Baum in G . \square

Damit folgt: Ist G planar, so ist $(\mathcal{M}(G))^*$ ein graphisches Matroid. Gilt für einen beliebigen Graphen G , dass $(\mathcal{M}(G))^*$ ein graphisches Matroid ist, etwa $(\mathcal{M}(G))^* = \mathcal{M}(G')$, so ist G' offensichtlich ein abstraktes Dual von G . Mit Aufgabe 40, Kapitel 2, gilt auch die Umkehrung: G ist genau dann planar, wenn G ein abstraktes Dual besitzt (Whitney [1933]). Daraus folgt: $(\mathcal{M}(G))^*$ ist genau dann graphisch, wenn G planar ist.

Beachte, dass die Eulersche Formel (Satz 2.32) direkt aus Satz 13.16 abgeleitet werden kann: Sei G ein zusammenhängender planarer Graph mit einer planaren Einbettung und sei $\mathcal{M}(G)$ das Kreismatroid von G . Nach Satz 13.15 (b) folgt $r(E(G)) + r^*(E(G)) = |E(G)|$. Aus $r(E(G)) = |V(G)| - 1$ (= Anzahl der Kanten in einem aufspannenden Baum) und $r^*(E(G)) = |V(G^*)| - 1$ (nach Satz 13.16) folgt, dass die Anzahl der Gebiete von G gleich $|V(G^*)| = |E(G)| - |V(G)| + 2$ ist. Dies ist die Eulersche Formel. (Wir haben jedoch die Eulersche Formel bereits im Beweis von Proposition 2.42 benutzt, und letzteres Resultat dann im Beweis von Satz 13.16.)

Die Dualitätstheorie der Unabhängigkeitssysteme hat auch einige schöne Anwendungen in der polyedrischen Kombinatorik. Ein Mengensystem (E, \mathcal{F}) heißt ein **Clutter**, falls $X \not\subset Y$ für alle $X, Y \in \mathcal{F}$. Ist (E, \mathcal{F}) ein Clutter, so definieren wir seinen **blockierenden Clutter** durch

$$BL(E, \mathcal{F}) := (E, \{X \subseteq E : X \cap Y \neq \emptyset \text{ für alle } Y \in \mathcal{F}, \text{ und } X \text{ inklusionsminimal bezüglich dieser Eigenschaft}\}).$$

Für ein Unabhängigkeitssystem (E, \mathcal{F}) und sein Dual (E, \mathcal{F}^*) sei \mathcal{B} bzw. \mathcal{B}^* die Familie der Basen und \mathcal{C} bzw. \mathcal{C}^* die Familie der Kreise. (Jeder Clutter kommt auf diese beiden Arten zustande, bis auf die Fälle $\mathcal{F} = \emptyset$ oder $\mathcal{F} = \{\emptyset\}$.) Aus den Definitionen folgt sofort, dass $(E, \mathcal{B}^*) = BL(E, \mathcal{C})$ und $(E, \mathcal{C}^*) = BL(E, \mathcal{B})$. Mit Proposition 13.14 folgt hieraus $BL(BL(E, \mathcal{F})) = (E, \mathcal{F})$ für jeden Clutter (E, \mathcal{F}) . Es folgen einige Clutter-Beispiele (E, \mathcal{F}) zusammen mit ihren blockierenden Cluttern (E, \mathcal{F}') . Für jedes dieser Beispiele ist $E = E(G)$ für einen Graphen G .

- (1) \mathcal{F} ist die Menge der aufspannenden Bäume, \mathcal{F}' ist die Menge der inklusionsminimalen Schnitte;

- (2) \mathcal{F} ist die Menge der Arboreszenzen mit Wurzel r , \mathcal{F}' ist die Menge der inklusionsminimalen r -Schnitte;
- (3) \mathcal{F} ist die Menge der s - t -Wege, \mathcal{F}' ist die Menge der s und t trennenden inklusionsminimalen Schnitte (dieses Beispiel gilt für ungerichtete Graphen und für Digraphen);
- (4) \mathcal{F} ist die Menge der Kreise in einem ungerichteten Graphen, \mathcal{F}' ist die Menge der Komplemente maximaler Wälder;
- (5) \mathcal{F} ist die Menge der Kreise in einem Digraphen, \mathcal{F}' ist die Menge der inklusionsminimalen Feedback-Kantenmengen (eine Feedback-Kantenmenge ist eine Menge von Kanten, deren Entfernen den Digraphen azyklisch macht);
- (6) \mathcal{F} ist die Menge der inklusionsminimalen Kantenmengen, deren Kontraktion den Digraphen stark zusammenhängend macht, \mathcal{F}' ist die Menge der inklusionsminimalen gerichteten Schnitte;
- (7) \mathcal{F} ist die Menge der inklusionsminimalen T -Joins, \mathcal{F}' ist die Menge der inklusionsminimalen T -Schnitte.

Sämtliche blockierende Relationen in diesen Beispielen sind einfach zu verifizieren:

- (1) und (2) folgen sofort aus den Sätzen 2.4 und 2.5, (3), (4) und (5) sind trivial,
- (6) folgt aus Korollar 2.7 und (7) aus Proposition 12.7.

In manchen Fällen liefert der blockierende Clutter eine polyedrische Charakterisierung des MINIMIERUNGSPROBLEMS FÜR UNABHÄNGIGKEITSSYSTEME für nichtnegative Kostenfunktionen:

Definition 13.17. Sei (E, \mathcal{F}) ein Clutter, (E, \mathcal{F}') sein blockierender Clutter und P die konvexe Hülle der Inzidenzvektoren der Elemente von \mathcal{F} . Wir sagen, dass (E, \mathcal{F}) die **Max-Flow-Min-Cut-Eigenschaft** hat, falls

$$\left\{ x + y : x \in P, y \in \mathbb{R}_+^E \right\} = \left\{ x \in \mathbb{R}_+^E : \sum_{e \in B} x_e \geq 1 \text{ für alle } B \in \mathcal{F}' \right\}.$$

Beispiele hierfür sind die Beispiele (2) und (7) in der obigen Liste (nach den Sätzen 6.15 und 12.18), aber auch die Beispiele (3) und (6) (siehe Aufgabe 11). Der folgende Satz verbindet die obige Überdeckungsformulierung mit einer Packungsformulierung des dualen Problems und erlaubt die Ableitung einiger weiterer Min-Max-Relationen:

Satz 13.18. (Fulkerson [1971], Lehman [1979]) Sei (E, \mathcal{F}) ein Clutter und (E, \mathcal{F}') sein blockierender Clutter. Dann sind die folgenden drei Aussagen äquivalent:

- (a) (E, \mathcal{F}) hat die Max-Flow-Min-Cut-Eigenschaft;
- (b) (E, \mathcal{F}') hat die Max-Flow-Min-Cut-Eigenschaft;
- (c) $\min\{c(A) : A \in \mathcal{F}\} = \max\{\mathbf{1}y : y \in \mathbb{R}_+^{\mathcal{F}'}, \sum_{B \in \mathcal{F}': e \in B} y_B \leq c(e) \text{ für alle } e \in E\}$ für jedes $c : E \rightarrow \mathbb{R}_+$.

Beweis: Da $BL(E, \mathcal{F}') = BL(BL(E, \mathcal{F})) = (E, \mathcal{F})$, genügt es, (a) \Rightarrow (c) \Rightarrow (b) zu beweisen. Die letzte Implikation (b) \Rightarrow (a) folgt dann durch Rollentausch von \mathcal{F} und \mathcal{F}' .

(a) \Rightarrow (c): Nach Korollar 3.33 gilt für jedes $c : E \rightarrow \mathbb{R}_+$:

$$\min\{c(A) : A \in \mathcal{F}\} = \min\{cx : x \in P\} = \min\left\{c(x + y) : x \in P, y \in \mathbb{R}_+^E\right\},$$

wobei P die konvexe Hülle der Inzidenzvektoren der Elemente von \mathcal{F} ist. Zusammen mit der Max-Flow-Min-Cut-Eigenschaft und dem LP-Dualitätssatz (Satz 3.20) folgt hieraus, dass (c) gilt.

(c) \Rightarrow (b): Sei P' die konvexe Hülle der Inzidenzvektoren der Elemente von \mathcal{F}' . Wir müssen zeigen, dass

$$\left\{x + y : x \in P', y \in \mathbb{R}_+^E\right\} = \left\{x \in \mathbb{R}_+^E : \sum_{e \in A} x_e \geq 1 \text{ für alle } A \in \mathcal{F}\right\}.$$

Da „ \subseteq “ nach der Definition von blockierenden Cluttern trivial ist, müssen wir nur die umgekehrte Inklusion zeigen. Sei $c \in \mathbb{R}_+^E$ ein Vektor mit $\sum_{e \in A} c_e \geq 1$ für alle $A \in \mathcal{F}$. Mit (c) haben wir

$$\begin{aligned} 1 &\leq \min\{c(A) : A \in \mathcal{F}\} \\ &= \max\left\{\mathbf{1}y : y \in \mathbb{R}_+^{\mathcal{F}'}, \sum_{B \in \mathcal{F}' : e \in B} y_B \leq c(e) \text{ für alle } e \in E\right\}, \end{aligned}$$

also sei $y \in \mathbb{R}_+^{\mathcal{F}'}$ ein Vektor mit $\mathbf{1}y = 1$ und $\sum_{B \in \mathcal{F}' : e \in B} y_B \leq c(e)$ für alle $e \in E$. Dann definiert $x_e := \sum_{B \in \mathcal{F}' : e \in B} y_B$ ($e \in E$) einen Vektor $x \in P'$ mit $x \leq c$, womit $c \in \{x + y : x \in P', y \in \mathbb{R}_+^E\}$ bewiesen ist. \square

Aus diesem Satz folgt z. B. das Max-Flow-Min-Cut-Theorem (Satz 8.6) ziemlich direkt: Sei (G, u, s, t) ein Netzwerk. Nach Aufgabe 1, Kapitel 7, ist die minimale Länge eines s - t -Weges in (G, u) gleich der maximalen Anzahl von s - t -Schnitten mit der Eigenschaft, dass jede Kante e in höchstens $u(e)$ von ihnen enthalten ist. Somit hat der Clutter der s - t -Wege (Beispiel (3) in der obigen Liste) die Max-Flow-Min-Cut-Eigenschaft, und dasselbe gilt für seinen blockierenden Clutter. Mittels Anwendung von (c) auf den Clutter der s - t -Schnitte minimaler Kapazität folgt nun das Max-Flow-Min-Cut-Theorem.

Beachte jedoch, dass Satz 13.18 nicht die Existenz eines ganzzahligen Vektors garantiert, für den das Maximum in (c) angenommen wird, auch nicht wenn c ganzzahlig ist. Der Clutter der T -Joins für $G = K_4$ und $T = V(G)$ zeigt, dass es einen solchen im Allgemeinen nicht gibt.

13.4 Der Greedy-Algorithmus

Sei wiederum (E, \mathcal{F}) ein Unabhängigkeitssystem und $c : E \rightarrow \mathbb{R}_+$. Wir betrachten das MAXIMIERUNGSPROBLEM für (E, \mathcal{F}, c) und formulieren zwei „Greedy-Algorithmen“. Wir brauchen hier keine negativen Gewichte zu betrachten, da negativ gewichtete Elemente niemals in einer optimalen Lösung vorkommen.

Wir gehen davon aus, dass es für (E, \mathcal{F}) ein Orakel gibt. Für den ersten Algorithmus sei dies einfach ein **Unabhängigkeits-Orakel**, d. h. ein Orakel, welches für eine gegebene Menge $F \subseteq E$ entscheidet, ob $F \in \mathcal{F}$ oder nicht.

BEST-IN-GREEDY-ALGORITHMUS

Input: Ein Unabhängigkeitssystem (E, \mathcal{F}) , gegeben durch ein Unabhängigkeits-Orakel. Gewichte $c : E \rightarrow \mathbb{R}_+$.

Output: Eine Menge $F \in \mathcal{F}$.

-
- ① Sortiere $E = \{e_1, e_2, \dots, e_n\}$, so dass $c(e_1) \geq c(e_2) \geq \dots \geq c(e_n)$.
 - ② Setze $F := \emptyset$.
 - ③ **For** $i := 1$ **to** n **do:** **If** $F \cup \{e_i\} \in \mathcal{F}$ **then** setze $F := F \cup \{e_i\}$.
-

Der zweite Algorithmus benötigt ein komplizierteres Orakel. Für eine gegebene Menge $F \subseteq E$ entscheidet dieses Orakel, ob F eine Basis enthält. Ein solches Orakel nennen wir ein **Basis-Obermengen-Orakel**.

WORST-OUT-GREEDY-ALGORITHMUS

Input: Ein Unabhängigkeitssystem (E, \mathcal{F}) , gegeben durch ein Basis-Obermengen-Orakel. Gewichte $c : E \rightarrow \mathbb{R}_+$.

Output: Eine Basis F von (E, \mathcal{F}) .

-
- ① Sortiere $E = \{e_1, e_2, \dots, e_n\}$, so dass $c(e_1) \leq c(e_2) \leq \dots \leq c(e_n)$.
 - ② Setze $F := E$.
 - ③ **For** $i := 1$ **to** n **do:** **If** $F \setminus \{e_i\}$ enthält eine Basis **then** setze $F := F \setminus \{e_i\}$.
-

Bevor wir diese beiden Algorithmen analysieren, wollen wir die beiden zugehörigen Orakel etwas näher betrachten. Eine interessante Frage ist, ob solche Orakel polynomiell äquivalent sein können, d. h. ob das eine Orakel durch einen das andere Orakel benutzenden polynomiellen Orakel-Algorithmus simuliert werden kann. Das Unabhängigkeits-Orakel und das Basis-Obermengen-Orakel scheinen nicht polynomiell äquivalent zu sein:

Betrachte das Unabhängigkeitssystem für das TSP (Beispiel (2) in der in Abschnitt 13.1 gegebenen Liste). Hier ist es einfach (siehe Aufgabe 14) zu entscheiden, ob eine Kantenmenge unabhängig ist, d. h. Teilmenge eines Hamilton-Kreises ist (beachte, dass ein vollständiger Graph vorliegt). Andererseits ist es ein schweres

Problem zu entscheiden, ob eine Kantenmenge einen Hamilton-Kreis enthält (dies ist ein *NP*-vollständiges Problem; siehe Satz 15.25).

Andererseits ist es in dem Unabhängigkeitssystem für das KÜRZESTE-WEGE-PROBLEM (Beispiel (3)) leicht zu entscheiden, ob eine Kantenmenge einen s - t -Weg enthält. Hier ist unbekannt, wie man in polynomieller Zeit entscheiden kann, ob eine gegebene Menge unabhängig (d. h. Teilmenge eines s - t -Weges) ist (Korte und Monma [1979] haben *NP*-Vollständigkeit bewiesen).

Für Matroide sind die beiden Orakel polynomiell äquivalent. Weitere äquivalente Orakel sind das **Rang-Orakel** und das **Abschluss-Orakel**, die den Rang bzw. den Abschluss einer gegebenen Teilmenge von E liefern (Aufgabe 17).

Es gibt aber auch – sogar für Matroide – weitere ganz natürliche, aber nicht polynomiell äquivalente, Orakel. Zum Beispiel ist das Orakel zur Bestimmung, ob eine gegebene Menge eine Basis ist, schwächer als das Unabhängigkeits-Orakel. Andererseits ist das Orakel zur Bestimmung der minimalen Kardinalität einer abhängigen Teilmenge einer gegebenen Menge $F \subseteq E$ stärker als das Unabhängigkeits-Orakel (Hausmann und Korte [1981]).

Analog kann man die beiden Greedy-Algorithmen auch für das MINIMIERUNGSPROBLEM formulieren. Man sieht leicht, dass der BEST-IN-GREEDY für das MAXIMIERUNGSPROBLEM für (E, \mathcal{F}, c) dem WORST-OUT-GREEDY für das MINIMIERUNGSPROBLEM für (E, \mathcal{F}^*, c) entspricht: Das Hinzufügen eines Elementes zu F im BEST-IN-GREEDY entspricht dem Entfernen eines Elementes aus F im WORST-OUT-GREEDY. Beachte, dass KRUSKALS ALGORITHMUS (siehe Abschnitt 6.1) ein BEST-IN-GREEDY-Algorithmus für das MINIMIERUNGSPROBLEM in einem Kreismatroid ist.

Der Rest dieses Abschnitts enthält einige Resultate über die Qualität einer mit den Greedy-Algorithmen gefundenen Lösung.

Satz 13.19. (Jenkyns [1976], Korte und Hausmann [1978]) *Sei (E, \mathcal{F}) ein Unabhängigkeitssystem. Für gegebenes $c : E \rightarrow \mathbb{R}_+$ bezeichne $G(E, \mathcal{F}, c)$ die Kosten einer mit dem BEST-IN-GREEDY für das MAXIMIERUNGSPROBLEM gefundenen Lösung und $\text{OPT}(E, \mathcal{F}, c)$ die Kosten einer optimalen Lösung. Dann gilt*

$$q(E, \mathcal{F}) \leq \frac{G(E, \mathcal{F}, c)}{\text{OPT}(E, \mathcal{F}, c)} \leq 1$$

für alle $c : E \rightarrow \mathbb{R}_+$. Ferner gibt es eine Kostenfunktion, mit welcher die untere Schranke angenommen wird.

Beweis: Sei $E = \{e_1, e_2, \dots, e_n\}$, $c : E \rightarrow \mathbb{R}_+$ und $c(e_1) \geq c(e_2) \geq \dots \geq c(e_n)$. Sei G_n die mit dem BEST-IN-GREEDY gefundene Lösung (für diese Sortierung von E), und sei O_n eine optimale Lösung. Definiere $E_j := \{e_1, \dots, e_j\}$, $G_j := G_n \cap E_j$ und $O_j := O_n \cap E_j$ ($j = 0, \dots, n$). Setze $d_n := c(e_n)$ und $d_j := c(e_j) - c(e_{j+1})$ für $j = 1, \dots, n-1$.

Da $O_j \in \mathcal{F}$, folgt $|O_j| \leq r(E_j)$. Da G_j eine Basis von E_j ist, folgt $|G_j| \geq \rho(E_j)$. Diese beiden Ungleichungen ergeben

$$\begin{aligned}
c(G_n) &= \sum_{j=1}^n (|G_j| - |G_{j-1}|) c(e_j) \\
&= \sum_{j=1}^n |G_j| d_j \\
&\geq \sum_{j=1}^n \rho(E_j) d_j \\
&\geq q(E, \mathcal{F}) \sum_{j=1}^n r(E_j) d_j \\
&\geq q(E, \mathcal{F}) \sum_{j=1}^n |O_j| d_j \\
&= q(E, \mathcal{F}) \sum_{j=1}^n (|O_j| - |O_{j-1}|) c(e_j) \\
&= q(E, \mathcal{F}) c(O_n).
\end{aligned} \tag{13.1}$$

Schließlich zeigen wir noch, dass die untere Schranke die bestmögliche ist. Wähle $F \subseteq E$ und Basen B_1, B_2 von F , so dass

$$\frac{|B_1|}{|B_2|} = q(E, \mathcal{F}).$$

Definiere

$$c(e) := \begin{cases} 1 & \text{für } e \in F \\ 0 & \text{für } e \in E \setminus F \end{cases}$$

und sortiere e_1, \dots, e_n so, dass $c(e_1) \geq c(e_2) \geq \dots \geq c(e_n)$ und $B_1 = \{e_1, \dots, e_{|B_1|}\}$. Dann ist $G(E, \mathcal{F}, c) = |B_1|$ und $\text{OPT}(E, \mathcal{F}, c) = |B_2|$, somit wird die untere Schranke angenommen. \square

Insbesondere haben wir den sogenannten Edmonds-Rado-Satz:

Satz 13.20. (Rado [1957], Edmonds [1971]) Ein Unabhängigkeitssystem (E, \mathcal{F}) ist genau dann ein Matroid, wenn der BEST-IN-GREEDY eine optimale Lösung für das MAXIMIERUNGSPROBLEM für (E, \mathcal{F}, c) für alle Kostenfunktionen $c : E \rightarrow \mathbb{R}_+$ bestimmt.

Beweis: Nach Satz 13.19 haben wir: $q(E, \mathcal{F}) < 1$ genau dann, wenn es eine Kostenfunktion $c : E \rightarrow \mathbb{R}_+$ gibt, für welche der BEST-IN-GREEDY keine optimale Lösung findet. Nach Proposition 13.7 folgt: $q(E, \mathcal{F}) < 1$ genau dann, wenn (E, \mathcal{F}) kein Matroid ist. \square

Dies ist einer der seltenen Fälle, wo wir eine Struktur durch ihr algorithmisches Verhalten definieren können. Wir erhalten auch eine polyedrische Beschreibung:

Satz 13.21. (Edmonds [1970]) Sei (E, \mathcal{F}) ein Matroid und $r : 2^E \rightarrow \mathbb{Z}_+$ seine Rangfunktion. Dann ist das **Matroid-Polytop** von (E, \mathcal{F}) , d. h. die konvexe Hülle der Inzidenzvektoren aller Elemente von \mathcal{F} , gleich

$$\left\{ x \in \mathbb{R}^E : x \geq 0, \sum_{e \in A} x_e \leq r(A) \text{ für alle } A \subseteq E \right\}.$$

Beweis: Offensichtlich enthält dieses Polytop die Inzidenzvektoren sämtlicher unabhängiger Mengen. Nach Korollar 3.32 bleibt nur noch zu zeigen, dass alle Ecken dieses Polytops ganzzahlig sind. Nach Satz 5.14 ist dies äquivalent dazu, dass wir zeigen:

$$\max \left\{ cx : x \geq 0, \sum_{e \in A} x_e \leq r(A) \text{ für alle } A \subseteq E \right\} \quad (13.2)$$

hat für jedes $c : E \rightarrow \mathbb{R}$ eine ganzzahlige optimale Lösung. O. B. d. A. können wir annehmen, dass $c(e) \geq 0$ für alle e , denn für alle $e \in E$ mit $c(e) < 0$ gilt: Jede optimale Lösung x von (13.2) hat $x_e = 0$.

Sei x eine optimale Lösung von (13.2). Ersetzen wir $|O_j|$ in (13.1) durch $\sum_{e \in E_j} x_e$ ($j = 0, \dots, n$), so erhalten wir $c(G_n) \geq \sum_{e \in E} c(e)x_e$. Somit liefert der BEST-IN-GREEDY eine Lösung, deren Inzidenzvektor eine weitere optimale Lösung von (13.2) ist. \square

Angewendet auf graphische Matroide, liefert dies auch Satz 6.13. Wie in diesem Spezialfall, haben wir auch im Allgemeinen vollständige duale Ganzzahligkeit (TDI). In Abschnitt 14.2 werden wir eine Verallgemeinerung dieses Resultats beweisen.

Die obige Bemerkung, dass der BEST-IN-GREEDY für das MAXIMIERUNGSPROBLEM für (E, \mathcal{F}, c) dem WORST-OUT-GREEDY für das MINIMIERUNGSPROBLEM für (E, \mathcal{F}^*, c) entspricht, legt die folgende duale Version von Satz 13.19 nahe:

Satz 13.22. (Korte und Monma [1979]) Sei (E, \mathcal{F}) ein Unabhängigkeitssystem. Für gegebenes $c : E \rightarrow \mathbb{R}_+$ sei $G(E, \mathcal{F}, c)$ eine mit dem WORST-OUT-GREEDY für das MINIMIERUNGSPROBLEM gefundene Lösung. Dann gilt

$$1 \leq \frac{G(E, \mathcal{F}, c)}{\text{OPT}(E, \mathcal{F}, c)} \leq \max_{F \subseteq E} \frac{|F| - \rho^*(F)}{|F| - r^*(F)} \quad (13.3)$$

für alle $c : E \rightarrow \mathbb{R}_+$, wobei r^* bzw. ρ^* die Rangfunktion bzw. die untere Rangfunktion des dualen Unabhängigkeitssystems (E, \mathcal{F}^*) ist. Ferner gibt es eine Kostenfunktion, für welche die obere Schranke angenommen wird.

Beweis: Wir benutzen dieselbe Notation wie im Beweis von Satz 13.19. Konstruktionsgemäß enthält $G_j \cup (E \setminus E_j)$ eine Basis von E , aber $(G_j \cup (E \setminus E_j)) \setminus \{e\}$ enthält für kein $e \in G_j$ ($j = 1, \dots, n$) eine Basis von E . Anders ausgedrückt: $E_j \setminus G_j$ ist eine Basis von E_j bezüglich (E, \mathcal{F}^*) , somit gilt $|E_j| - |G_j| \geq \rho^*(E_j)$.

Da $O_n \subseteq E \setminus (E_j \setminus O_j)$ und O_n eine Basis ist, ist $E_j \setminus O_j$ unabhängig in (E, \mathcal{F}^*) . Somit gilt $|E_j| - |O_j| \leq r^*(E_j)$.

Daraus folgt

$$\begin{aligned}|G_j| &\leq |E_j| - \rho^*(E_j) \quad \text{und} \\ |O_j| &\geq |E_j| - r^*(E_j).\end{aligned}$$

Die obere Schranke erhalten wir wie in (13.1). Es bleibt nur noch zu zeigen, dass diese Schranke kleinstmöglich ist. Setze

$$c(e) := \begin{cases} 1 & \text{für } e \in F \\ 0 & \text{für } e \in E \setminus F \end{cases},$$

wobei $F \subseteq E$ eine Menge ist, in der das Maximum in (13.3) angenommen wird. Sei B_1 eine Basis von F bezüglich (E, \mathcal{F}^*) , mit $|B_1| = \rho^*(F)$. Sortieren wir e_1, \dots, e_n so, dass $c(e_1) \geq c(e_2) \geq \dots \geq c(e_n)$ und $B_1 = \{e_1, \dots, e_{|B_1|}\}$, dann folgt $G(E, \mathcal{F}, c) = |F| - |B_1|$ und $\text{OPT}(E, \mathcal{F}, c) = |F| - r^*(F)$. \square

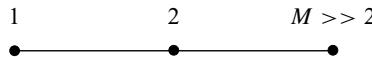


Abbildung 13.1.

Wenden wir den WORST-OUT-GREEDY auf das MAXIMIERUNGSPROBLEM bzw. den BEST-IN-GREEDY auf das MINIMIERUNGSPROBLEM an, so gibt es keine positive untere bzw. endliche obere Schranke für $\frac{G(E, \mathcal{F}, c)}{\text{OPT}(E, \mathcal{F}, c)}$. Um dies zu sehen, betrachte man das Problem der Bestimmung einer inklusionsminimalen Knotenüberdeckung maximalen Gewichtes oder einer inklusionsmaximalen stabilen Menge minimalen Gewichtes in dem in Abb. 13.1 dargestellten einfachen Graphen.

Für Matroide spielt es jedoch keine Rolle, ob wir den BEST-IN-GREEDY oder den WORST-OUT-GREEDY benutzen: Da alle Basen dieselbe Kardinalität haben, ist das MINIMIERUNGSPROBLEM für (E, \mathcal{F}, c) äquivalent zum MAXIMIERUNGSPROBLEM für (E, \mathcal{F}, c') , wobei $c'(e) := M - c(e)$ für alle $e \in E$ und $M := 1 + \max\{c(e) : e \in E\}$. Somit löst KRUSKALS ALGORITHMUS (Abschnitt 6.1) das MINIMUM-SPANNING-TREE-PROBLEM optimal.

Der Edmonds-Rado-Satz (Satz 13.20) liefert auch die folgende Charakterisierung optimaler k -elementiger Lösungen des MAXIMIERUNGSPROBLEMS:

Satz 13.23. *Sei (E, \mathcal{F}) ein Matroid, $c : E \rightarrow \mathbb{R}$, $k \in \mathbb{N}$ und $X \in \mathcal{F}$ mit $|X| = k$. Dann gilt $c(X) = \max\{c(Y) : Y \in \mathcal{F}, |Y| = k\}$ genau dann, wenn die beiden folgenden Bedingungen erfüllt sind:*

- (a) Für alle $y \in E \setminus X$ mit $X \cup \{y\} \notin \mathcal{F}$ und alle $x \in C(X, y)$ gilt $c(x) \geq c(y)$;
- (b) Für alle $y \in E \setminus X$ mit $X \cup \{y\} \in \mathcal{F}$ und alle $x \in X$ gilt $c(x) \geq c(y)$.

Beweis: Die Notwendigkeit ist trivial: Wird eine der beiden Bedingungen für irgendwelche y und x verletzt, so hat die k -elementige Menge $X' := (X \cup \{y\}) \setminus \{x\} \in \mathcal{F}$ größeres Gewicht als X .

Um zu sehen, dass die beiden Bedingungen hinreichend sind, sei $\mathcal{F}' := \{F \in \mathcal{F} : |F| \leq k\}$ und $c'(e) := c(e) + M$ für alle $e \in E$, wobei $M = \max\{|c(e)| : e \in E\}$. Sortiere $E = \{e_1, \dots, e_n\}$ so, dass $c'(e_1) \geq \dots \geq c'(e_n)$ und dass für beliebiges i gilt: Aus $c'(e_i) = c'(e_{i+1})$ und $e_{i+1} \in X$ folgt $e_i \in X$ (d.h. unter Elementen gleichen Gewichtes kommen Elemente von X zuerst dran).

Sei X' die mit dem BEST-IN-GREEDY gefundene Lösung für die Instanz (E, \mathcal{F}', c') , wobei ① die Elemente wie angegeben sortiert. Da (E, \mathcal{F}') ein Matroid ist, folgt aus dem Edmonds-Rado-Satz (Satz 13.20):

$$\begin{aligned} c(X') + kM &= c'(X') = \max\{c'(Y) : Y \in \mathcal{F}'\} \\ &= \max\{c(Y) : Y \in \mathcal{F}, |Y| = k\} + kM. \end{aligned}$$

Wir schließen mit dem Beweis, dass $X = X'$. Wir wissen, dass $|X| = k = |X'|$. Angenommen, es wäre $X \neq X'$. Sei $e_i \in X' \setminus X$ mit minimalem i . Dann gilt $X \cap \{e_1, \dots, e_{i-1}\} = X' \cap \{e_1, \dots, e_{i-1}\}$. Ist nun $X \cup \{e_i\} \notin \mathcal{F}$, so folgt aus (a), dass $C(X, e_i) \subseteq X'$, ein Widerspruch. Ist andererseits $X \cup \{e_i\} \in \mathcal{F}$, so folgt aus (b), dass $X \subseteq X'$, was auch unmöglich ist. \square

Diesen Satz werden wir in Abschnitt 13.7 verwenden. Der durch das graphische Matroid (E, \mathcal{F}) und $k = r(E)$ gegebene Spezialfall ist ein Teil von Satz 6.3.

13.5 Der Schnitt von Matroiden

Definition 13.24. Gegeben seien zwei Unabhängigkeitssysteme (E, \mathcal{F}_1) und (E, \mathcal{F}_2) . Ihr Schnitt sei das Mengensystem $(E, \mathcal{F}_1 \cap \mathcal{F}_2)$.

Der Schnitt einer endlichen Anzahl von Unabhängigkeitssystemen wird analog definiert. Es ist klar, dass wieder ein Unabhängigkeitssystem resultiert.

Proposition 13.25. Jedes Unabhängigkeitssystem (E, \mathcal{F}) ist der Schnitt endlich vieler Matroide.

Beweis: Jeder Kreis C von (E, \mathcal{F}) definiert ein Matroid $(E, \{F \subseteq E : C \setminus F \neq \emptyset\})$ nach Satz 13.12 (dies folgt auch leicht aus der Definition). Der Schnitt aller dieser Matroide ist natürlich (E, \mathcal{F}) . \square

Da der Schnitt von Matroiden im Allgemeinen nicht wieder ein Matroid ist, besteht keine Hoffnung, eine optimale gemeinsame unabhängige Menge mit einem Greedy-Algorithmus bestimmen zu können. Das folgende Resultat, zusammen mit Satz 13.19, liefert jedoch eine Schranke für die mit dem BEST-IN-GREEDY gefundene Lösung:

Proposition 13.26. Ist (E, \mathcal{F}) der Schnitt von p Matroiden, so folgt $q(E, \mathcal{F}) \geq \frac{1}{p}$.

Beweis: Nach Satz 13.12(b) folgt: Für jedes $X \in \mathcal{F}$ und $e \in E$ enthält $X \cup \{e\}$ höchstens p Kreise. Die Aussage folgt nun mit Satz 13.8. \square

Besonders interessant sind Unabhängigkeitssysteme, die der Schnitt zweier Matroide sind. Das wichtigste Beispiel ist hier das Matching-Problem in einem bipartiten Graphen $G = (A \cup B, E)$. Nehmen wir $\mathcal{F} := \{F \subseteq E : F \text{ ist ein Matching in } G\}$, so ist (E, \mathcal{F}) der Schnitt der folgenden zwei Matroide. Seien

$$\begin{aligned}\mathcal{F}_1 &:= \{F \subseteq E : |\delta_F(x)| \leq 1 \text{ für alle } x \in A\} \quad \text{und} \\ \mathcal{F}_2 &:= \{F \subseteq E : |\delta_F(x)| \leq 1 \text{ für alle } x \in B\},\end{aligned}$$

dann sind (E, \mathcal{F}_1) und (E, \mathcal{F}_2) Matroide nach Proposition 13.4(d). Offensichtlich gilt $\mathcal{F} = \mathcal{F}_1 \cap \mathcal{F}_2$.

Ein zweites Beispiel ist das aus sämtlichen Branchings in einem Digraphen G bestehende Unabhängigkeitssystem (Beispiel (8) in der am Anfang von Abschnitt 13.1 gegebenen Liste). Hier sind die unabhängigen Mengen des einen Matroids alle Kantenmengen mit der Eigenschaft: In jedem Knoten des Digraphen kommt höchstens eine Kante (aus der betrachteten Kantenmenge) an (siehe Proposition 13.4(e)). Das zweite Matroid ist das Kreismatroid $\mathcal{M}(G)$ des zugrunde liegenden ungerichteten Graphen.

Wir werden nun Edmonds' Algorithmus für das folgende Problem beschreiben:

MATROID-INTERSEKTIONSPROBLEM

Instanz: Zwei Matroide $(E, \mathcal{F}_1), (E, \mathcal{F}_2)$, gegeben durch Unabhängigkeits-Orakel.

Aufgabe: Bestimme eine Menge $F \in \mathcal{F}_1 \cap \mathcal{F}_2$ mit maximalem $|F|$.

Wir beginnen mit dem folgenden Lemma. Wie früher, bezeichnet $C(X, e)$ für gegebenes $X \in \mathcal{F}$ und $e \in E$ den eindeutig bestimmten Kreis in $X \cup \{e\}$, falls $X \cup \{e\} \notin \mathcal{F}$, sonst ist $C(X, e) = \emptyset$.

Lemma 13.27. (Frank [1981]) *Sei (E, \mathcal{F}) ein Matroid und $X \in \mathcal{F}$. Seien $x_1, \dots, x_s \in X$ und $y_1, \dots, y_s \notin X$ mit*

- (a) $x_k \in C(X, y_k)$ für $k = 1, \dots, s$ und
- (b) $x_j \notin C(X, y_k)$ für $1 \leq j < k \leq s$.

Dann ist $(X \setminus \{x_1, \dots, x_s\}) \cup \{y_1, \dots, y_s\} \in \mathcal{F}$.

Beweis: Sei $X_r := (X \setminus \{x_1, \dots, x_r\}) \cup \{y_1, \dots, y_r\}$. Wir zeigen mittels Induktion, dass $X_r \in \mathcal{F}$ für alle r . Für $r = 0$ ist dies trivial. Angenommen, dass $X_{r-1} \in \mathcal{F}$ für ein $r \in \{1, \dots, s\}$. Ist $X_{r-1} \cup \{y_r\} \in \mathcal{F}$, so folgt $X_r \in \mathcal{F}$ sofort. Andernfalls enthält $X_{r-1} \cup \{y_r\}$ einen eindeutig bestimmten Kreis C (nach Satz 13.12(b)). Da $C(X, y_r) \subseteq X_{r-1} \cup \{y_r\}$ (nach (b)), folgt $C = C(X, y_r)$. Mit (a) folgt dann aber $x_r \in C(X, y_r) = C$, somit ist $X_r = (X_{r-1} \cup \{y_r\}) \setminus \{x_r\} \in \mathcal{F}$. \square

Die Idee von EDMONDS' MATROID-INTERSEKTION-ALGORITHMUS ist die folgende. Beginnend mit $X = \emptyset$, augmentieren wir X um ein Element pro Iteration. Da im Allgemeinen keine Hoffnung besteht, ein Element e mit $X \cup \{e\} \in \mathcal{F}_1 \cap \mathcal{F}_2$ finden zu können, suchen wir nach sogenannten „alternierenden Wegen“. Um dies zu erleichtern, führen wir einen Hilfsgraphen ein. Den Begriff $C(X, e)$ wenden wir auf (E, \mathcal{F}_i) an und schreiben $C_i(X, e)$ ($i = 1, 2$).

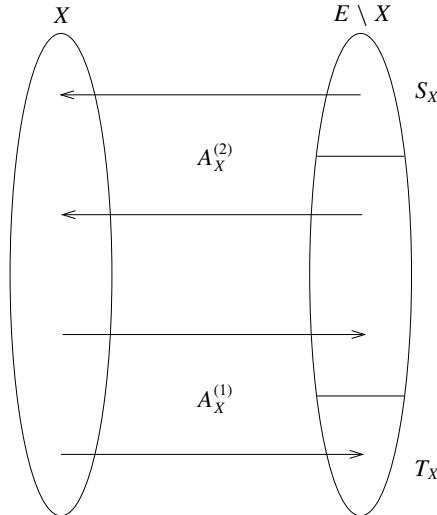


Abbildung 13.2.

Gegeben sei eine Menge $X \in \mathcal{F}_1 \cap \mathcal{F}_2$. Wir definieren einen Hilfsgraphen G_X wie folgt:

$$\begin{aligned} A_X^{(1)} &:= \{(x, y) : y \in E \setminus X, x \in C_1(X, y) \setminus \{y\}\}, \\ A_X^{(2)} &:= \{(y, x) : y \in E \setminus X, x \in C_2(X, y) \setminus \{y\}\}, \\ G_X &:= (E, A_X^{(1)} \cup A_X^{(2)}). \end{aligned}$$

Wir setzen

$$\begin{aligned} S_X &:= \{y \in E \setminus X : X \cup \{y\} \in \mathcal{F}_1\}, \\ T_X &:= \{y \in E \setminus X : X \cup \{y\} \in \mathcal{F}_2\}, \end{aligned}$$

(siehe Abb. 13.2) und suchen einen kürzesten Weg von S_X nach T_X . Ein solcher Weg erlaubt uns, die Menge X zu augmentieren. (Ist $S_X \cap T_X \neq \emptyset$, so haben wir einen Weg der Länge null und können X dann um irgendein Element in $S_X \cap T_X$ augmentieren.)

Lemma 13.28. Sei $X \in \mathcal{F}_1 \cap \mathcal{F}_2$. Seien $y_0, x_1, y_1, \dots, x_s, y_s$ die Knoten eines kürzesten y_0 - y_s -Weges in G_X (in dieser Reihenfolge), mit $y_0 \in S_X$ und $y_s \in T_X$. Dann ist

$$X' := (X \cup \{y_0, \dots, y_s\}) \setminus \{x_1, \dots, x_s\} \in \mathcal{F}_1 \cap \mathcal{F}_2.$$

Beweis: Zunächst zeigen wir, dass $X \cup \{y_0\}, x_1, \dots, x_s$ und y_1, \dots, y_s die Bedingungen (a) und (b) in Lemma 13.27 bezüglich \mathcal{F}_1 erfüllen. Beachte, dass $X \cup \{y_0\} \in \mathcal{F}_1$, da $y_0 \in S_X$. Die Bedingung (a) wird erfüllt, weil $(x_j, y_j) \in A_X^{(1)}$ für alle j . Die Bedingung (b) wird erfüllt, da der Weg sonst abgekürzt werden könnte. Damit folgt $X' \in \mathcal{F}_1$.

Zweitens zeigen wir, dass $X \cup \{y_s\}, x_s, x_{s-1}, \dots, x_1$ und y_{s-1}, \dots, y_1, y_0 die Bedingungen (a) und (b) in Lemma 13.27 bezüglich \mathcal{F}_2 erfüllen. Beachte, dass $X \cup \{y_s\} \in \mathcal{F}_2$, da $y_s \in T_X$. Die Bedingung (a) wird erfüllt, weil $(y_{j-1}, x_j) \in A_X^{(2)}$ für alle j . Die Bedingung (b) wird erfüllt, da der Weg sonst abgekürzt werden könnte. Damit folgt $X' \in \mathcal{F}_2$. \square

Nun werden wir beweisen, dass X bereits kardinalitätsmaximal ist, wenn es keinen S_X - T_X -Weg in G_X gibt. Dazu benötigen wir die folgende einfache Tatsache:

Proposition 13.29. Seien (E, \mathcal{F}_1) und (E, \mathcal{F}_2) zwei Matroide mit den Rangfunktionen r_1 und r_2 . Für jedes $F \in \mathcal{F}_1 \cap \mathcal{F}_2$ und jedes $Q \subseteq E$ gilt dann

$$|F| \leq r_1(Q) + r_2(E \setminus Q).$$

Beweis: Aus $F \cap Q \in \mathcal{F}_1$ folgt $|F \cap Q| \leq r_1(Q)$. Aus $F \setminus Q \in \mathcal{F}_2$ folgt $|F \setminus Q| \leq r_2(E \setminus Q)$ analog. Addition dieser beiden Ungleichungen ergibt die Aussage. \square

Lemma 13.30. $X \in \mathcal{F}_1 \cap \mathcal{F}_2$ ist kardinalitätsmaximal genau dann, wenn es keinen S_X - T_X -Weg in G_X gibt.

Beweis: Gibt es einen S_X - T_X -Weg, so auch einen kürzesten. Nun wenden wir Lemma 13.28 an und erhalten eine Menge $X' \in \mathcal{F}_1 \cap \mathcal{F}_2$ mit größerer Kardinalität.

Andernfalls sei R die Menge der von S_X in G_X erreichbaren Knoten (siehe Abb. 13.3). Wir haben $R \cap T_X = \emptyset$. Sei r_1 bzw. r_2 die Rangfunktion von \mathcal{F}_1 bzw. \mathcal{F}_2 .

Wir behaupten, dass $r_2(R) = |X \cap R|$. Angenommen, dies wäre nicht der Fall. Dann gibt es ein $y \in R \setminus X$ mit $(X \cap R) \cup \{y\} \in \mathcal{F}_2$. Da $X \cup \{y\} \notin \mathcal{F}_2$ (weil $y \notin T_X$), enthält der Kreis $C_2(X, y)$ ein Element $x \in X \setminus R$. Dann bedeutet $(y, x) \in A_X^{(2)}$ aber, dass es eine in R beginnende Kante gibt. Dies widerspricht der Definition von R .

Als Nächstes beweisen wir, dass $r_1(E \setminus R) = |X \setminus R|$. Angenommen, dies wäre nicht der Fall. Dann gibt es ein $y \in (E \setminus R) \setminus X$ mit $(X \setminus R) \cup \{y\} \in \mathcal{F}_1$. Da $X \cup \{y\} \notin \mathcal{F}_1$ (weil $y \notin S_X$), enthält der Kreis $C_1(X, y)$ ein Element $x \in X \cap R$. Dann bedeutet $(x, y) \in A_X^{(1)}$ aber, dass es eine in R beginnende Kante gibt. Dies widerspricht der Definition von R .

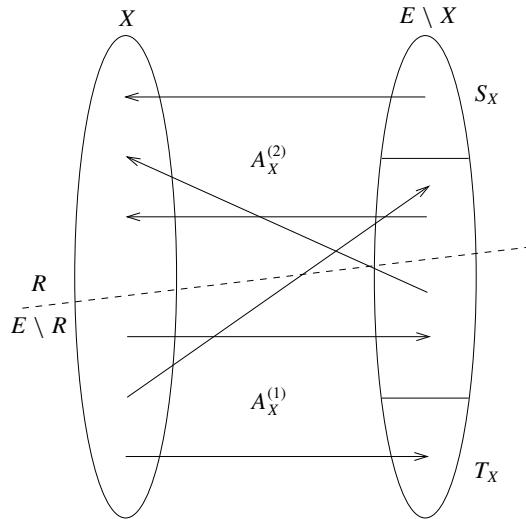


Abbildung 13.3.

Zusammen haben wir somit $|X| = r_2(R) + r_1(E \setminus R)$. Nach Proposition 13.29 folgt hieraus die Optimalität. \square

Der letzte Abschnitt dieses Beweises ergibt die folgende Min-Max-Gleichung:

Satz 13.31. (Edmonds [1970]) *Seien (E, \mathcal{F}_1) und (E, \mathcal{F}_2) zwei Matroide mit den Rangfunktionen r_1 und r_2 . Dann gilt*

$$\max \{|X| : X \in \mathcal{F}_1 \cap \mathcal{F}_2\} = \min \{r_1(Q) + r_2(E \setminus Q) : Q \subseteq E\}.$$

 \square

Wir sind nun bereit für eine detaillierte Beschreibung des Algorithmus.

EDMONDS' MATROID-INTERSEKTIONS-ALGORITHMUS

Input: Zwei Matroide (E, \mathcal{F}_1) und (E, \mathcal{F}_2) , gegeben durch Unabhängigkeits-Orakel.

Output: Eine Menge $X \in \mathcal{F}_1 \cap \mathcal{F}_2$ mit maximaler Kardinalität.

- ① Setze $X := \emptyset$.
- ② **For** jedes $y \in E \setminus X$ und $i \in \{1, 2\}$ **do:** Berechne
 $C_i(X, y) := \{x \in X \cup \{y\} : X \cup \{y\} \notin \mathcal{F}_i, (X \cup \{y\}) \setminus \{x\} \in \mathcal{F}_i\}$.
- ③ Berechne S_X , T_X und G_X , wie oben definiert.
- ④ Wende BFS an, um einen kürzesten S_X-T_X -Weg P in G_X zu bestimmen.
If es gibt keinen **then stop**.
- ⑤ Setze $X := X \triangle V(P)$ und **go to** ②.

Satz 13.32. EDMONDS' MATROID-INTERSEKTIONS-ALGORITHMUS löst das MATROID-INTERSEKTIONS-PROBLEM korrekt in $O(|E|^3\theta)$ -Zeit, wobei θ die maximale Komplexität der beiden Unabhängigkeits-Orakel ist.

Beweis: Die Korrektheit folgt aus den Lemmata 13.28 und 13.30. ② und ③ können in $O(|E|^2\theta)$ -Zeit ausgeführt werden, ④ in $O(|E|^2)$ -Zeit. Da es höchstens $|E|$ Augmentierungen gibt, ist die gesamte Komplexität $O(|E|^3\theta)$. \square

Schnellere Matroid-Intersektions-Algorithmen sind von Cunningham [1986] und von Gabow und Xu [1996] besprochen worden. Wir weisen darauf hin, dass das Problem der Bestimmung einer Menge mit maximaler Kardinalität im Schnitt dreier Matroide NP-schwer ist; siehe Aufgabe 17(c), Kapitel 15.

13.6 Matroid-Partitionierung

Anstelle des Schnittes von Matroiden betrachten wir nun deren Vereinigung. Diese wird wie folgt definiert:

Definition 13.33. Gegeben seien k Matroide $(E, \mathcal{F}_1), \dots, (E, \mathcal{F}_k)$. Eine Menge $X \subseteq E$ heißt **partitionierbar**, falls es eine Partition $X = X_1 \cup \dots \cup X_k$ mit $X_i \in \mathcal{F}_i$ für $i = 1, \dots, k$ gibt. Sei \mathcal{F} die Familie der partitionierbaren Teilmengen von E . Dann heißt (E, \mathcal{F}) die **Vereinigung** oder die **Summe** von $(E, \mathcal{F}_1), \dots, (E, \mathcal{F}_k)$.

Wir werden beweisen, dass die Vereinigung von Matroiden wieder ein Matroid ist. Ferner werden wir das folgende Problem mittels Matroid-Intersektion lösen:

MATROID-PARTITIONS-PROBLEM

- Instanz:* Eine Zahl $k \in \mathbb{N}$, k Matroide $(E, \mathcal{F}_1), \dots, (E, \mathcal{F}_k)$, gegeben durch Unabhängigkeits-Orakel.
- Aufgabe:* Bestimme eine partitionierbare Menge $X \subseteq E$ mit maximaler Kardinalität.

Der Hauptsatz der Matroid-Partitionierung ist der folgende

Satz 13.34. (Nash-Williams [1967]) Seien $(E, \mathcal{F}_1), \dots, (E, \mathcal{F}_k)$ Matroide mit den Rangfunktionen r_1, \dots, r_k und sei (E, \mathcal{F}) ihre Vereinigung. Dann ist (E, \mathcal{F}) ein Matroid und seine Rangfunktion r wird gegeben durch

$$r(X) = \min_{A \subseteq X} \left(|X \setminus A| + \sum_{i=1}^k r_i(A) \right).$$

Beweis: Es ist (E, \mathcal{F}) offensichtlich ein Unabhängigkeitssystem. Sei $X \subseteq E$. Zunächst zeigen wir, dass $r(X) = \min_{A \subseteq X} \left(|X \setminus A| + \sum_{i=1}^k r_i(A) \right)$.

Für jedes partitionierbare $Y \subseteq X$, d.h. $Y = Y_1 \cup \dots \cup Y_k$ mit $Y_i \in \mathcal{F}_i$ ($i = 1, \dots, k$), und für jedes $A \subseteq X$ haben wir

$$|Y| = |Y \setminus A| + |Y \cap A| \leq |X \setminus A| + \sum_{i=1}^k |Y_i \cap A| \leq |X \setminus A| + \sum_{i=1}^k r_i(A),$$

somit folgt $r(X) \leq \min_{A \subseteq X} \left(|X \setminus A| + \sum_{i=1}^k r_i(A) \right)$.

Sei andererseits $X' := X \times \{1, \dots, k\}$. Wir definieren nun zwei Matroide auf X' . Für $Q \subseteq X'$ und $i \in \{1, \dots, k\}$ schreiben wir $Q_i := \{e \in X : (e, i) \in Q\}$. Sei

$$\mathcal{I}_1 := \{Q \subseteq X' : Q_i \in \mathcal{F}_i \text{ für alle } i = 1, \dots, k\}$$

und

$$\mathcal{I}_2 := \{Q \subseteq X' : Q_i \cap Q_j = \emptyset \text{ für alle } i \neq j\}.$$

Offensichtlich sind (X', \mathcal{I}_1) und (X', \mathcal{I}_2) Matroide und ihre Rangfunktionen sind $s_1(Q) := \sum_{i=1}^k r_i(Q_i)$ und $s_2(Q) := \left| \bigcup_{i=1}^k Q_i \right|$ für $Q \subseteq X'$.

Nun kann die Familie der partitionierbaren Teilmengen von X folgendermaßen geschrieben werden:

$$\begin{aligned} \{A \subseteq X : \text{es gibt eine Funktion } f : A \rightarrow \{1, \dots, k\} \\ \text{mit } \{(e, f(e)) : e \in A\} \in \mathcal{I}_1 \cap \mathcal{I}_2\}. \end{aligned}$$

Somit ist die maximale Kardinalität einer partitionierbaren Menge gleich der maximalen Kardinalität einer gemeinsamen unabhängigen Menge in \mathcal{I}_1 und \mathcal{I}_2 . Nach Satz 13.31 ist diese maximale Kardinalität gleich $\min \{s_1(Q) + s_2(X' \setminus Q) : Q \subseteq X'\}$. Wird dieses Minimum in $Q \subseteq X'$ angenommen, so haben wir, indem wir $A := Q_1 \cap \dots \cap Q_k$ setzen:

$$r(X) = s_1(Q) + s_2(X' \setminus Q) = \sum_{i=1}^k r_i(Q_i) + \left| X \setminus \bigcap_{i=1}^k Q_i \right| \geq \sum_{i=1}^k r_i(A) + |X \setminus A|.$$

Somit haben wir eine Menge $A \subseteq X$ mit der Eigenschaft $\sum_{i=1}^k r_i(A) + |X \setminus A| \leq r(X)$ gefunden.

Damit gilt die Formel für die Rangfunktion r . Schließlich zeigen wir noch, dass r submodular ist. Nach Satz 13.10 folgt dann, dass (E, \mathcal{F}) ein Matroid ist. Um die Submodularität zu beweisen, seien $X, Y \subseteq E$ und $A \subseteq X, B \subseteq Y$ mit $r(X) = |X \setminus A| + \sum_{i=1}^k r_i(A)$ und $r(Y) = |Y \setminus B| + \sum_{i=1}^k r_i(B)$. Dann gilt

$$\begin{aligned} & r(X) + r(Y) \\ &= |X \setminus A| + |Y \setminus B| + \sum_{i=1}^k (r_i(A) + r_i(B)) \\ &\geq |(X \cup Y) \setminus (A \cup B)| + |(X \cap Y) \setminus (A \cap B)| + \sum_{i=1}^k (r_i(A \cup B) + r_i(A \cap B)) \\ &\geq r(X \cup Y) + r(X \cap Y). \end{aligned}$$

□

Die Konstruktion im obigen Beweis (Edmonds [1970]) führt das MATROID-PARTITIONS-PROBLEM auf das MATROID-INTERSEKTIONS-PROBLEM zurück. Eine Reduktion in der umgekehrten Richtung ist auch möglich (Aufgabe 22), somit können diese beiden Probleme als äquivalent betrachtet werden.

Wir weisen darauf hin, dass wir eine kardinalitätsmaximale unabhängige Menge in der Vereinigung einer beliebigen endlichen Anzahl von Matroiden auf effiziente Weise finden können, während dasselbe Problem für den Schnitt von mehr als zwei Matroiden nicht effizient lösbar scheint.

13.7 Gewichteter Schnitt von Matroiden

Wir werden nun die allgemeinere gewichtete Version des Schnittes von Matroiden betrachten.

GEWICHTETES MATROID-INTERSEKTIONS-PROBLEM

Instanz: Zwei Matroide (E, \mathcal{F}_1) und (E, \mathcal{F}_2) , gegeben durch Unabhängigkeits-Orakel. Gewichte $c : E \rightarrow \mathbb{R}$.

Aufgabe: Bestimme eine Menge $X \in \mathcal{F}_1 \cap \mathcal{F}_2$ mit maximalem Gewicht $c(X)$.

Wir werden einen von Frank [1981] stammenden primal-dualen Algorithmus für dieses Problem beschreiben. Dieser verallgemeinert EDMONDS' MATROID-INTERSEKTIONS-ALGORITHMUS. Wiederum beginnen wir mit $X := X_0 = \emptyset$ und erhöhen die Kardinalität um eins pro Iteration. Damit erhalten wir Mengen $X_0, \dots, X_m \in \mathcal{F}_1 \cap \mathcal{F}_2$ mit $|X_k| = k$ ($k = 0, \dots, m$) und $m = \max\{|X| : X \in \mathcal{F}_1 \cap \mathcal{F}_2\}$. Jedes X_k wird optimal sein, d.h.

$$c(X_k) = \max\{c(X) : X \in \mathcal{F}_1 \cap \mathcal{F}_2, |X| = k\}. \quad (13.4)$$

Zum Schluss brauchen wir dann nur eine optimale Menge unter den Mengen X_0, \dots, X_m zu wählen.

Die Hauptidee ist die Aufspaltung der Gewichtsfunktion. Zu jedem Zeitpunkt haben wir zwei Funktionen $c_1, c_2 : E \rightarrow \mathbb{R}$ mit $c_1(e) + c_2(e) = c(e)$ für alle $e \in E$. Für jedes k werden wir sicherstellen, dass

$$c_i(X_k) = \max\{c_i(X) : X \in \mathcal{F}_i, |X| = k\} \quad (i = 1, 2). \quad (13.5)$$

Aus dieser Eigenschaft folgt offensichtlich (13.4). Um (13.5) zu gewährleisten, verwenden wir das Optimalitätskriterium von Satz 13.23. Statt G_X , S_X und T_X , betrachten wir nur einen Teilgraphen \tilde{G} und Teilmengen \tilde{S} und \tilde{T} .

GEWICHTETER MATROID-INTERSEKTIONS-ALGORITHMUS

Input: Zwei Matroide (E, \mathcal{F}_1) und (E, \mathcal{F}_2) , gegeben durch Unabhängigkeits-Orakel. Gewichte $c : E \rightarrow \mathbb{R}$.

Output: Eine Menge $X \in \mathcal{F}_1 \cap \mathcal{F}_2$ maximalen Gewichtes.

- ① Setze $k := 0$ und $X_0 := \emptyset$. Setze $c_1(e) := c(e)$ und $c_2(e) := 0$ für alle $e \in E$.
- ② For jedes $y \in E \setminus X_k$ und $i \in \{1, 2\}$ do: Berechne
 $C_i(X_k, y) := \{x \in X_k \cup \{y\} : X_k \cup \{y\} \notin \mathcal{F}_i, (X_k \cup \{y\}) \setminus \{x\} \in \mathcal{F}_i\}$.
- ③ Berechne

$$\begin{aligned} A^{(1)} &:= \{(x, y) : y \in E \setminus X_k, x \in C_1(X_k, y) \setminus \{y\}\}, \\ A^{(2)} &:= \{(y, x) : y \in E \setminus X_k, x \in C_2(X_k, y) \setminus \{y\}\}, \\ S &:= \{y \in E \setminus X_k : X_k \cup \{y\} \in \mathcal{F}_1\}, \\ T &:= \{y \in E \setminus X_k : X_k \cup \{y\} \in \mathcal{F}_2\}. \end{aligned}$$

- ④ Berechne

$$\begin{aligned} m_1 &:= \max\{c_1(y) : y \in S\}, \\ m_2 &:= \max\{c_2(y) : y \in T\}, \\ \bar{S} &:= \{y \in S : c_1(y) = m_1\}, \\ \bar{T} &:= \{y \in T : c_2(y) = m_2\}, \\ \bar{A}^{(1)} &:= \{(x, y) \in A^{(1)} : c_1(x) = c_1(y)\}, \\ \bar{A}^{(2)} &:= \{(y, x) \in A^{(2)} : c_2(x) = c_2(y)\}, \\ \bar{G} &:= (E, \bar{A}^{(1)} \cup \bar{A}^{(2)}). \end{aligned}$$

- ⑤ Wende BFS an, um die Menge R der von \bar{S} aus in \bar{G} erreichbaren Knoten zu bestimmen.
- ⑥ If $R \cap \bar{T} \neq \emptyset$ then bestimme einen \bar{S} - \bar{T} -Weg P in \bar{G} mit minimaler Kantenanzahl, setze $X_{k+1} := X_k \Delta V(P)$ und $k := k + 1$ und go to ②.
- ⑦ Berechne

$$\begin{aligned} \varepsilon_1 &:= \min\{c_1(x) - c_1(y) : (x, y) \in \delta_{A^{(1)}}^+(R)\}, \\ \varepsilon_2 &:= \min\{c_2(x) - c_2(y) : (y, x) \in \delta_{A^{(2)}}^+(R)\}, \\ \varepsilon_3 &:= \min\{m_1 - c_1(y) : y \in S \setminus R\}, \\ \varepsilon_4 &:= \min\{m_2 - c_2(y) : y \in T \cap R\}, \\ \varepsilon &:= \min\{\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4\}, \end{aligned}$$

(wobei $\min \emptyset := \infty$).

- ⑧ If $\varepsilon < \infty$ then setze $c_1(x) := c_1(x) - \varepsilon$ und $c_2(x) := c_2(x) + \varepsilon$ für alle $x \in R$. Go to ④.
If $\varepsilon = \infty$ then sei X die Menge maximalen Gewichtes unter den Mengen X_0, X_1, \dots, X_k . Stop.

Siehe Edmonds [1979] und Lawler [1976] für frühere Versionen dieses Algorithmus.

Um zu zeigen, dass dieser Algorithmus korrekt arbeitet, benötigen wir die folgende Verallgemeinerung von Lemma 13.27:

Lemma 13.35. (Frank [1981]) *Sei (E, \mathcal{F}) ein Matroid, $c : E \rightarrow \mathbb{R}$, und $X \in \mathcal{F}$. Seien $x_1, \dots, x_l \in X$ und $y_1, \dots, y_l \notin X$ mit*

- (a) $x_j \in C(X, y_j)$ und $c(x_j) = c(y_j)$ für $j = 1, \dots, l$ und
- (b) $x_i \notin C(X, y_j)$ oder $c(x_i) > c(y_j)$ für $1 \leq i < j \leq l$ und
- (c) $x_i \notin C(X, y_j)$ oder $c(x_i) \geq c(y_j)$ für $1 \leq j < i \leq l$.

Dann ist $(X \setminus \{x_1, \dots, x_l\}) \cup \{y_1, \dots, y_l\} \in \mathcal{F}$.

Beweis: Der Beweis erfolgt wiederum mittels Induktion über l . Der Fall $l = 1$ ist trivial nach (a). Sei $\mu = \min_{i=1}^l c(x_i)$ und h der kleinste Index, für den das Minimum angenommen wird. Sei $X' := (X \setminus \{x_h\}) \cup \{y_h\}$. Nach (a) gilt $X' \in \mathcal{F}$. Wir werden zeigen, dass $C(X', y_j) = C(X, y_j)$ für alle $j \neq h$; dann gelten (a) und (b) auch für X' und die restlichen Indizes $\{1, \dots, l\} \setminus \{h\}$. Mittels Induktion ist der Beweis damit erbracht.

Sei $j \neq h$. Angenommen, es gelte $C(X', y_j) \neq C(X, y_j)$. Dann ist $x_h \in C(X, y_j)$. Falls nun $h < j$, dann folgt aus (a) und (b), dass $\mu = c(x_h) > c(y_j) = c(x_j)$, was ein Widerspruch zur Definition von μ ist. Falls jedoch $h > j$, dann folgt aus (a) und (c), dass $\mu = c(x_h) \geq c(y_j) = c(x_j)$, was ein Widerspruch zur Wahl von h ist. \square

Satz 13.36. (Frank [1981]) *Der GEWICHTETE MATROID-INTERSEKTIONSAALGORITHMUS löst das GEWICHTETE MATROID-INTERSEKTIONSPROBLEM korrekt in $O(|E|^4 + |E|^3\theta)$ -Zeit, wobei θ die maximale Komplexität der beiden Unabhängigkeits-Orakel ist.*

Beweis: Sei m der letzte Wert von k (wir werden später zeigen, dass der Algorithmus terminiert). Der Algorithmus berechnet Mengen X_0, X_1, \dots, X_m . Terminiert der Algorithmus, so haben wir $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon_4 = \infty$, also ist T nicht erreichbar von S in $(E, A^{(1)} \cup A^{(2)})$. Nach Lemma 13.30 folgt dann $m = |X_m| = \max\{|X| : X \in \mathcal{F}_1 \cap \mathcal{F}_2\}$.

Zum Beweis der Korrektheit zeigen wir, dass zu jedem Zeitpunkt $X_k \in \mathcal{F}_1 \cap \mathcal{F}_2$ und (13.5) gilt. Da $c = c_1 + c_2$ stets gilt, folgt $c(X_k) = \max\{c(X) : X \in \mathcal{F}_1 \cap \mathcal{F}_2, |X| = k\}$ für alle $k = 0, \dots, m$, und somit die Korrektheit.

Am Anfang ist $k = 0$, $X_k = \emptyset \in \mathcal{F}_1 \cap \mathcal{F}_2$, und (13.5) ist trivial. Wir zeigen nun, dass die Eigenschaften bei jeder Iteration des Algorithmus erhalten bleiben.

Zunächst bemerken wir, dass nach (13.5) und Satz 13.23(a) Folgendes gilt:

- (i) $c_1(x) > c_1(y)$ für alle $(x, y) \in A^{(1)} \setminus \bar{A}^{(1)}$, und
- (ii) $c_2(x) > c_2(y)$ für alle $(y, x) \in A^{(2)} \setminus \bar{A}^{(2)}$.

Als Erstes betrachten wir eine Iteration in der $R \cap \bar{T} \neq \emptyset$. Dann bestimmen wir einen Weg P in ⑤, etwa mit den Knoten $y_0, x_1, y_1, \dots, x_l, y_l$ in dieser Reihenfolge.

Wie in Lemma 13.28 zeigen wir, dass (E, \mathcal{F}_1) , c_1 , $X \cup \{y_0\}$, x_1, \dots, x_l und y_1, \dots, y_l die Voraussetzungen von Lemma 13.35 erfüllen: $X \cup \{y_0\} \in \mathcal{F}_1$ folgt aus $y_0 \in S$. Bedingung (a) in Lemma 13.35 folgt aus $(x_j, y_j) \in \bar{A}^{(1)}$ für $j = 1, \dots, l$,

(c) folgt aus (i), und (b) folgt aus (i) und der Tatsache, dass P ein kürzester Weg in \tilde{G} ist. Somit ist $X_{k+1} \in \mathcal{F}_1$.

Analog erfüllen (E, \mathcal{F}_2) , c_2 , $X_k \cup \{y_l\}$, x_l, \dots, x_1 und y_{l-1}, \dots, y_0 die Voraussetzungen von Lemma 13.35. Hier folgt $X_k \cup \{y_l\} \in \mathcal{F}_2$, da $y_l \in T$. Bedingung (a) in Lemma 13.35 folgt aus $(y_{j-1}, x_j) \in \bar{A}^{(2)}$ für $j = 1, \dots, l$, (c) folgt aus (ii), und (b) folgt wieder aus (ii) und der Tatsache, dass P ein kürzester Weg in \tilde{G} ist. Somit ist $X_{k+1} \in \mathcal{F}_2$.

Also haben wir gezeigt, dass $X_{k+1} \in \mathcal{F}_1 \cap \mathcal{F}_2$. Um zu zeigen, dass X_{k+1} (13.5) erfüllt, benutzen wir Satz 13.23.

Nach Definition von \tilde{G} haben wir $c_1(X_{k+1}) = c_1(X_k) + c_1(y_0)$ und $c_2(X_{k+1}) = c_2(X_k) + c_2(y_l)$. Da X_k (13.5) erfüllt, gelten die Bedingungen (a) und (b) aus Satz 13.23 bezüglich X_k und sowohl \mathcal{F}_1 als auch \mathcal{F}_2 .

Nach Definition von \tilde{S} und $y_0 \in \tilde{S}$ gelten die beiden Bedingungen weiterhin für $X_k \cup \{y_0\}$ und \mathcal{F}_1 . Somit folgt $c_1(X_{k+1}) = c_1(X_k \cup \{y_0\}) = \max\{c_1(Y) : Y \in \mathcal{F}_1, |Y| = k+1\}$. Und nach Definition von \tilde{T} und $y_l \in \tilde{T}$ gelten die Bedingungen (a) und (b) von Satz 13.23 weiterhin für $X_k \cup \{y_l\}$ und \mathcal{F}_2 , woraus $c_2(X_{k+1}) = c_2(X_k \cup \{y_l\}) = \max\{c_2(Y) : Y \in \mathcal{F}_2, |Y| = k+1\}$ folgt. Anders ausgedrückt: (13.5) gilt tatsächlich für X_{k+1} .

Nun betrachten wir eine Iteration in welcher $R \cap \tilde{T} = \emptyset$, also ändern wir c_1 und c_2 in ⑧. Zunächst zeigen wir, dass $\varepsilon > 0$. Die Definition von R ergibt $\delta_{\tilde{G}}^+(R) = \emptyset$, womit aus (i) und (ii) sofort folgt, dass $\varepsilon_1 > 0$ und $\varepsilon_2 > 0$.

Es gilt $m_1 \geq c_1(y)$ für alle $y \in S$. Gilt zusätzlich $y \notin R$, so folgt $y \notin \tilde{S}$, also ist $m_1 > c_1(y)$. Damit folgt $\varepsilon_3 > 0$. Ebenso erhalten wir $\varepsilon_4 > 0$ (benutze $\tilde{T} \cap R = \emptyset$). Somit gilt $\varepsilon > 0$.

Wir können nun beweisen, dass (13.5) unter ⑧ erhalten bleibt. Sei c'_1 das modifizierte c_1 , d.h.

$$c'_1(x) := \begin{cases} c_1(x) - \varepsilon & \text{für } x \in R \\ c_1(x) & \text{für } x \notin R \end{cases}.$$

Wie werden beweisen, dass X_k und c'_1 die Bedingungen (a) und (b) in Satz 13.23 bezüglich \mathcal{F}_1 erfüllen.

Zum Beweis von (a) sei $y \in E \setminus X_k$ und $x \in C_1(X_k, y) \setminus \{y\}$. Angenommen, es gelte $c'_1(x) < c'_1(y)$. Da $c_1(x) \geq c_1(y)$ und $\varepsilon > 0$, folgt $x \in R$ und $y \notin R$. Da auch $(x, y) \in A^{(1)}$ gilt, folgt $\varepsilon \leq \varepsilon_1 \leq c_1(x) - c_1(y) = (c'_1(x) + \varepsilon) - c'_1(y)$, ein Widerspruch.

Zum Beweis von (b) sei $x \in X_k$ und $y \in E \setminus X_k$ mit $X_k \cup \{y\} \in \mathcal{F}_1$. Angenommen, es gelte $c'_1(y) > c'_1(x)$. Da $c_1(y) \leq m_1 \leq c_1(x)$, folgt $x \in R$ und $y \notin R$. Da $y \in S$, folgt $\varepsilon \leq \varepsilon_3 \leq m_1 - c_1(y) \leq c_1(x) - c_1(y) = (c'_1(x) + \varepsilon) - c'_1(y)$ ein Widerspruch.

Sei c'_2 das modifizierte c_2 , d.h.

$$c'_2(x) := \begin{cases} c_2(x) + \varepsilon & \text{für } x \in R \\ c_2(x) & \text{für } x \notin R \end{cases}.$$

Wir werden zeigen, dass X_k und c'_2 die Bedingungen (a) und (b) in Satz 13.23 bezüglich \mathcal{F}_2 erfüllen.

Zum Beweis von (a) sei $y \in E \setminus X_k$ und $x \in C_2(X_k, y) \setminus \{y\}$. Angenommen, es gelte $c'_2(x) < c'_2(y)$. Da $c_2(x) \geq c_2(y)$, folgt $y \in R$ und $x \notin R$. Da auch $(y, x) \in A^{(2)}$, folgt $\varepsilon \leq \varepsilon_2 \leq c_2(x) - c_2(y) = c'_2(x) - (c'_2(y) - \varepsilon)$, ein Widerspruch.

Zum Beweis von (b) sei $x \in X_k$ und $y \in E \setminus X_k$ mit $X_k \cup \{y\} \in \mathcal{F}_2$. Angenommen, es gelte $c'_2(y) > c'_2(x)$. Da $c_2(y) \leq m_2 \leq c_2(x)$, folgt $y \in R$ und $x \notin R$. Da $y \in T$, folgt $\varepsilon \leq \varepsilon_4 \leq m_2 - c_2(y) \leq c_2(x) - c_2(y) = c'_2(x) - (c'_2(y) - \varepsilon)$, ein Widerspruch.

Damit haben wir bewiesen, dass (13.5) während ⑧ nicht verletzt wird. Somit folgt, dass der Algorithmus korrekt arbeitet.

Nun wenden wir uns seiner Laufzeit zu. Beachte, dass nach Aktualisierung der Gewichte in ⑧ die neuen Mengen \bar{S} , \bar{T} , bzw. R , wie sie daraufhin in ④ und ⑤ berechnet werden, Obermengen der alten Mengen \bar{S} , \bar{T} , bzw. R sind. Ist $\varepsilon = \varepsilon_4 < \infty$, so erfolgt eine Augmentierung (k wird erhöht). Andernfalls wächst die Kardinalität von R sofort (in ⑤) um mindestens eins. Somit werden ④ – ⑧ weniger als $|E|$ mal zwischen zwei Augmentierungen wiederholt.

Die Laufzeit von ④ – ⑧ ist $O(|E|^2)$ und die Gesamlaufzeit zwischen zwei Augmentierungen ist $O(|E|^3)$ plus $O(|E|^2)$ Orakelaufrufe (in ②). Da $m \leq |E|$ Augmentierungen erfolgen, folgt die angegebene Gesamlaufzeit. \square

Die Laufzeit kann leicht auf $O(|E|^3\theta)$ verbessert werden (Aufgabe 24).

Aufgaben

1. Man beweise, dass sämtliche Unabhängigkeitssysteme außer (5) und (6) in der am Anfang von Abschnitt 13.1 gegebenen Liste im Allgemeinen keine Matroide sind.
2. Man zeige, dass das uniforme Matroid mit vier Elementen und Rang 2 kein graphisches Matroid ist.
3. Man beweise, dass jedes graphische Matroid über jedem beliebigen Körper repräsentierbar ist.
4. Sei G ein ungerichteter Graph, $K \in \mathbb{N}$ und \mathcal{F} die Familie derjenigen Teilmengen von $E(G)$, die die Vereinigung von K Wäldern sind. Man beweise, dass $(E(G), \mathcal{F})$ ein Matroid ist.
5. Sei G ein bipartiter Graph mit Bipartition $V(G) = A \dot{\cup} B$ und (A, \mathcal{F}) ein Matroid. Man beweise, dass das folgende Mengensystem ein Matroid ist: $(B, \{Y \subseteq B : Y = \emptyset \text{ oder } \exists X \in \mathcal{F} : G[X \cup Y] \text{ hat ein perfektes Matching}\})$.
6. Man berechne größtmögliche untere Schranken für die Rangquotienten der in der Liste am Anfang von Abschnitt 13.1 angegebenen Unabhängigkeitssysteme.
7. Sei \mathcal{S} eine endliche Familie endlicher Mengen. Eine Menge T heißt eine Transversale von \mathcal{S} , falls es eine Bijektion $\Phi : T \rightarrow \mathcal{S}$ mit $t \in \Phi(t)$ für alle $t \in T$ gibt. (In Aufgabe 6, Kapitel 10, wird eine notwendige und hinreichende

Bedingung für die Existenz einer Transversalen angegeben.) Unter der Voraussetzung, dass \mathcal{S} eine Transversale hat, beweise man, dass die Familie der Transversalen von \mathcal{S} die Familie der Basen eines Matroids ist.

8. Sei E eine endliche Menge und $\mathcal{B} \subseteq 2^E$. Man zeige, dass \mathcal{B} genau dann die Menge der Basen eines Matroids (E, \mathcal{F}) ist, wenn Folgendes gilt:
 - (B1) $\mathcal{B} \neq \emptyset$;
 - (B2) Für $B_1, B_2 \in \mathcal{B}$ und $y \in B_2 \setminus B_1$ gibt es ein $x \in B_1 \setminus B_2$ mit $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$.
9. Sei G ein Graph. Sei \mathcal{F} die Familie derjenigen Mengen $X \subseteq V(G)$, für die es einen Knoten von X überdeckendes kardinalitätsmaximales Matching gibt. Man beweise, dass $(V(G), \mathcal{F})$ ein Matroid ist. Man gebe das duale Matroid an.
10. Man zeige, dass $\mathcal{M}(G^*) = (\mathcal{M}(G))^*$ auch für unzusammenhängende Graphen G gilt. Dies verallgemeinert Satz 13.16.

Hinweis: Man verwende Aufgabe 37(a), Kapitel 2.

11. Man zeige, dass die Clutter in (3) und (6) der in Abschnitt 13.3 angegebenen Liste die Max-Flow-Min-Cut-Eigenschaft besitzen. (Man verwende Satz 19.17.) Man zeige, dass die Clutter in (1), (4) und (5) der Liste die Max-Flow-Min-Cut-Eigenschaft im Allgemeinen nicht besitzen.
- * 12. Ein Clutter (E, \mathcal{F}) heißt binär, falls es für alle $X_1, \dots, X_k \in \mathcal{F}$ mit k ungerade ein $Y \in \mathcal{F}$ mit $Y \subseteq X_1 \triangle \dots \triangle X_k$ gibt. Man beweise, dass der Clutter der inklusionsminimalen T -Joins und der Clutter der inklusionsminimalen T -Schnitte (Beispiel (7) der in Abschnitt 13.3 angegebenen Liste) binär sind. Man beweise, dass ein Clutter genau dann binär ist, wenn $|A \cap B|$ ungerade für alle $A \in \mathcal{F}$ und $B \in \mathcal{F}'$ ist, wobei (E, \mathcal{F}') der blockierende Clutter ist. Man folgere hieraus, dass ein Clutter genau dann binär ist, wenn sein blockierender Clutter binär ist.
Bemerkung: Seymour [1977] hat die binären Clutter mit der Max-Flow-Min-Cut-Eigenschaft klassifiziert.
- * 13. Sei P ein nach rechts oben unbeschränktes Polyeder, d. h. es gilt $x + y \in P$ für alle $x \in P$ und $y \geq 0$. Das blockierende Polyeder von P ist $B(P) := \{z : z^\top x \geq 1 \text{ für alle } x \in P\}$. Man beweise, dass $B(P)$ wieder ein nach rechts oben unbeschränktes Polyeder ist und dass $B(B(P)) = P$.
Bemerkung: Man vergleiche dies mit Satz 4.22.
14. Wie kann man (in polynomieller Zeit) prüfen, ob eine gegebene Menge von Kanten eines vollständigen Graphen G Teilmenge eines Hamilton-Kreises in G ist?
15. Man beweise: Für ein Matroid (E, \mathcal{F}) maximiert der BEST-IN-GREEDY jede Bottleneck-Funktion $c(F) = \min\{c_e : e \in F\}$ über der Menge der Basen.
16. Sei (E, \mathcal{F}) ein Matroid und $c : E \rightarrow \mathbb{R}$ mit $c(e) \neq c(e')$ für alle $e \neq e'$ und $c(e) \neq 0$ für alle e . Man beweise, dass sowohl das MAXIMIERUNGS- als auch das MINIMIERUNGSPROBLEM für (E, \mathcal{F}, c) eine eindeutig bestimmte optimale Lösung hat.

- * 17. Man beweise, dass für Matroide die folgenden Orakel polynomiell äquivalent sind: das Unabhängigkeits-, Basis-Obermengen-, Abschluss- und Rang-Orakel.
Hinweis: Um zu zeigen, dass man das Rang-Orakel auf das Unabhängigkeits-Orakel zurückführen kann, verwende man den BEST-IN-GREEDY. Um zu zeigen, dass man das Unabhängigkeits-Orakel auf das Basis-Obermengen-Orakel zurückführen kann, verwende man den WORST-OUT-GREEDY.
 (Hausmann und Korte [1981])
- 18. Sei (E, \mathcal{F}) ein Matroid mit Rangfunktion r , E' eine endliche Menge und $f : E \rightarrow E'$. Sei $\mathcal{F}' := \{f(F) : F \in \mathcal{F}\}$. Man beweise:
 - (E', \mathcal{F}') ist ein Matroid.
 - Die Rangfunktion r' von (E', \mathcal{F}') wird gegeben durch

$$r(X) = \min \left\{ |X \setminus W| + r(f^{-1}(W)) : W \subseteq X \right\}$$

für alle $X \subseteq E'$.

Hinweis: Man verwende Satz 13.31.

(Nash-Williams [1967])

- 19. In einem gegebenen ungerichteten Graphen G möchte man die Kanten so mit einer Mindestanzahl von Farben färben, dass für keinen Kreis C von G alle Kanten von C dieselbe Farbe haben. Man zeige, dass es für dieses Problem einen polynomiellen Algorithmus gibt.
- 20. Seien $(E, \mathcal{F}_1), \dots, (E, \mathcal{F}_k)$ Matroide mit den Rangfunktionen r_1, \dots, r_k . Man beweise, dass eine Menge $X \subseteq E$ genau dann partitionierbar ist, wenn $|A| \leq \sum_{i=1}^k r_i(A)$ für alle $A \subseteq X$. Man zeige, dass Satz 6.20 ein Spezialfall hiervon ist.
 (Edmonds und Fulkerson [1965])
- 21. Sei (E, \mathcal{F}) ein Matroid mit der Rangfunktion r . Man beweise unter Verwendung von Satz 13.34:
 - (E, \mathcal{F}) hat k paarweise disjunkte Basen genau dann, wenn $kr(A) + |E \setminus A| \geq kr(E)$ für alle $A \subseteq E$.
 - (E, \mathcal{F}) hat k unabhängige Mengen, deren Vereinigung gleich E ist, genau dann, wenn $kr(A) \geq |A|$ für alle $A \subseteq E$.
 Man zeige, dass die Sätze 6.20 und 6.17 Spezialfälle sind.
- 22. Seien (E, \mathcal{F}_1) und (E, \mathcal{F}_2) zwei Matroide. Sei X eine kardinalitätsmaximale partitionierbare Teilmenge bezüglich (E, \mathcal{F}_1) und (E, \mathcal{F}_2^*) : $X = X_1 \dot{\cup} X_2$ mit $X_1 \in \mathcal{F}_1$ und $X_2 \in \mathcal{F}_2^*$. Sei $B_2 \supseteq X_2$ eine Basis von \mathcal{F}_2^* . Man beweise, dass dann $X \setminus B_2$ eine Menge maximaler Kardinalität in $\mathcal{F}_1 \cap \mathcal{F}_2$ ist.
 (Edmonds [1970])
- 23. Sei (E, \mathcal{S}) ein Mengensystem und (E, \mathcal{F}) ein Matroid mit der Rangfunktion r . Man zeige, dass \mathcal{S} genau dann eine in (E, \mathcal{F}) unabhängige Transversale hat, wenn $r(\bigcup_{B \in \mathcal{B}} B) \geq |\mathcal{B}|$ für alle $\mathcal{B} \subseteq \mathcal{S}$.
Hinweis: Man beschreibe zunächst mittels Satz 13.34 die Rangfunktion desjenigen Matroids, dessen unabhängige Mengen gerade die Transversalen sind (Aufgabe 7). Dann wende man Satz 13.31 an.
 (Rado [1942])

24. Man zeige, dass die Laufzeit des GEWICHTETEN MATROID-INTERSEKTIONS-ALGORITHMUS (siehe Satz 13.36) auf $O(|E|^3\theta)$ verbessert werden kann.
25. Seien (E, \mathcal{F}_1) und (E, \mathcal{F}_2) zwei Matroide und $c : E \rightarrow \mathbb{R}$. Seien $X_0, \dots, X_m \in \mathcal{F}_1 \cap \mathcal{F}_2$ mit $|X_k| = k$ und $c(X_k) = \max\{c(X) : X \in \mathcal{F}_1 \cap \mathcal{F}_2, |X| = k\}$ für alle k . Man beweise, dass

$$c(X_{k+1}) - c(X_k) \leq c(X_k) - c(X_{k-1})$$

für alle $k = 1, \dots, m-2$.

(Krogdahl [unveröffentlicht])

26. Man betrachte das folgende Problem. Gegeben sei ein Digraph G mit Kanten gewichten. Für einen gegebenen Knoten $s \in V(G)$ und eine Zahl k bestimme man einen Teilgraphen H minimalen Gewichtes von G , der k paarweise kantendisjunkte Wege von s aus zu jedem anderen Knoten enthält. Man zeige, dass dieses Problem auf das GEWICHTETE MATROID-INTERSEKTIONS-PROBLEM zurückgeführt werden kann.

Hinweis: Siehe Aufgabe 26, Kapitel 6, und Aufgabe 4 dieses Kapitels.

(Edmonds [1970]; Frank und Tardos [1989]; Gabow [1995])

27. Seien A und B zwei endliche Mengen der Kardinalität $n \in \mathbb{N}$, sei $G = (A \cup B, \{\{a, b\} : a \in A, b \in B\})$ der vollständige bipartite Graph, $\bar{a} \in A$ und $c : E(G) \rightarrow \mathbb{R}$ eine Kostenfunktion. Sei \mathcal{T} die Familie der Kantenmengen aller aufspannenden Bäume T in G mit $|\delta_T(a)| = 2$ für alle $a \in A \setminus \{\bar{a}\}$. Man zeige, dass man ein Element von \mathcal{T} mit minimalen Kosten in $O(n^7)$ -Zeit berechnen kann. Wie viele Kanten werden mit \bar{a} inzident sein?

Literatur

Allgemeine Literatur:

- Bixby, R.E., und Cunningham, W.H. [1995]: Matroid optimization and algorithms. In: Handbook of Combinatorics; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, Hrsg.), Elsevier, Amsterdam, 1995
- Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., und Schrijver, A. [1998]: Combinatorial Optimization. Wiley, New York 1998, Kapitel 8
- Faigle, U. [1987]: Matroids in combinatorial optimization. In: Combinatorial Geometries (N. White, Hrsg.), Cambridge University Press, 1987
- Frank, A. [2011]: Connections in Combinatorial Optimization. Oxford University Press, Oxford 2011
- Gondran, M., und Minoux, M. [1984]: Graphs and Algorithms. Wiley, Chichester 1984, Kapitel 9
- Lawler, E.L. [1976]: Combinatorial Optimization; Networks and Matroids. Holt, Rinehart and Winston, New York 1976, Kapitel 7 und 8
- Oxley, J.G. [1992]: Matroid Theory. Oxford University Press, Oxford 1992
- von Randow, R. [1975]: Introduction to the Theory of Matroids. Springer, Berlin 1975
- Recski, A. [1989]: Matroid Theory and its Applications. Springer, Berlin 1989
- Schrijver, A. [2003]: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin 2003, Kapitel 39–42
- Welsh, D.J.A. [1976]: Matroid Theory. Academic Press, London 1976

Zitierte Literatur:

- Cunningham, W.H. [1986] : Improved bounds for matroid partition and intersection algorithms. *SIAM Journal on Computing* 15 (1986), 948–957
- Edmonds, J. [1970]: Submodular functions, matroids and certain polyhedra. In: Combinatorial Structures and Their Applications; Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications 1969 (R. Guy, H. Hanani, N. Sauer, J. Schönheim, Hrsg.), Gordon and Breach, New York 1970, pp. 69–87
- Edmonds, J. [1971]: Matroids and the greedy algorithm. *Mathematical Programming* 1 (1971), 127–136
- Edmonds, J. [1979]: Matroid intersection. In: Discrete Optimization I; Annals of Discrete Mathematics 4 (P.L. Hammer, E.L. Johnson, B.H. Korte, Hrsg.), North-Holland, Amsterdam 1979, pp. 39–49
- Edmonds, J., und Fulkerson, D.R. [1965]: Transversals and matroid partition. *Journal of Research of the National Bureau of Standards B* 69 (1965), 67–72
- Frank, A. [1981]: A weighted matroid intersection algorithm. *Journal of Algorithms* 2 (1981), 328–336
- Frank, A., und Tardos, É. [1989]: An application of submodular flows. *Linear Algebra and Its Applications* 114/115 (1989), 329–348
- Fulkerson, D.R. [1971]: Blocking and anti-blocking pairs of polyhedra. *Mathematical Programming* 1 (1971), 168–194
- Gabow, H.N. [1995]: A matroid approach to finding edge connectivity and packing arborescences. *Journal of Computer and System Sciences* 50 (1995), 259–273
- Gabow, H.N., und Xu, Y. [1996]: Efficient theoretic and practical algorithms for linear matroid intersection problems. *Journal of Computer and System Sciences* 53 (1996), 129–147
- Hausmann, D., Jenkyns, T.A., und Korte, B. [1980]: Worst case analysis of greedy type algorithms for independence systems. *Mathematical Programming Study* 12 (1980), 120–131
- Hausmann, D., und Korte, B. [1981]: Algorithmic versus axiomatic definitions of matroids. *Mathematical Programming Study* 14 (1981), 98–111
- Jenkyns, T.A. [1976]: The efficiency of the greedy algorithm. *Proceedings of the 7th S-E Conference on Combinatorics, Graph Theory, and Computing, Utilitas Mathematica*, Winnipeg 1976, pp. 341–350
- Korte, B., und Hausmann, D. [1978]: An analysis of the greedy algorithm for independence systems. In: Algorithmic Aspects of Combinatorics; *Annals of Discrete Mathematics* 2 (B. Alspach, P. Hell, D.J. Miller, Hrsg.), North-Holland, Amsterdam 1978, pp. 65–74
- Korte, B., und Monma, C.L. [1979]: Some remarks on a classification of oracle-type algorithms. In: Numerische Methoden bei graphentheoretischen und kombinatorischen Problemen; Band 2 (L. Collatz, G. Meinardus, W. Wetterling, Hrsg.), Birkhäuser, Basel 1979, pp. 195–215
- Lehman, A. [1979]: On the width-length inequality. *Mathematical Programming* 17 (1979), 403–417
- Nash-Williams, C.S.J.A. [1967]: An application of matroids to graph theory. In: Theory of Graphs; *Proceedings of an International Symposium in Rome 1966* (P. Rosenstiehl, Hrsg.), Gordon and Breach, New York, 1967, pp. 263–265
- Rado, R. [1942]: A theorem on independence relations. *Quarterly Journal of Math. Oxford* 13 (1942), 83–89

- Rado, R. [1957]: Note on independence functions. *Proceedings of the London Mathematical Society* 7 (1957), 300–320
- Seymour, P.D. [1977]: The matroids with the Max-Flow Min-Cut property. *Journal of Combinatorial Theory B* 23 (1977), 189–222
- Whitney, H. [1933]: Planar graphs. *Fundamenta Mathematicae* 21 (1933), 73–84
- Whitney, H. [1935]: On the abstract properties of linear dependence. *American Journal of Mathematics* 57 (1935), 509–533



14 Verallgemeinerungen von Matroiden

Es gibt einige interessante Verallgemeinerungen von Matroiden. Unabhängigkeitsysteme haben wir in Abschnitt 13.1 bereits kennengelernt; sie entstehen durch Weglassen des Axioms (M3). In Abschnitt 14.1 betrachten wir Greedoide; sie entstehen durch Weglassen des Axioms (M2) (anstatt von (M3)). Ferner gibt es gewisse mit Matroiden und submodularen Funktionen verwandte Polytope - sogenannte Polymatroida - die zu starken Verallgemeinerungen von wichtigen Sätzen führen; diese werden wir in Abschnitt 14.2 betrachten. In den Abschnitten 14.3 und 14.4 werden wir zwei verschiedene Zugänge zu dem Problem der Minimierung einer beliebigen submodularen Funktion kennenlernen: einen über die ELLIPSOIDMETHODE und einen anderen mittels eines kombinatorischen Algorithmus. Für den wichtigen Spezialfall einer symmetrischen submodularen Funktion werden wir in Abschnitt 14.5 einen einfacheren Algorithmus angeben. Schließlich werden wir in Abschnitt 14.6 das Problem der Maximierung einer submodularen Funktion betrachten.

14.1 Greedoide

Definitionsgemäß ist ein Mengensystem (E, \mathcal{F}) genau dann ein Matroid, wenn es die folgenden drei Eigenschaften hat:

- (M1) $\emptyset \in \mathcal{F}$;
- (M2) Für $X \subseteq Y \in \mathcal{F}$ gilt $X \in \mathcal{F}$;
- (M3) Sind $X, Y \in \mathcal{F}$ mit $|X| > |Y|$, so gibt es ein $x \in X \setminus Y$ mit $Y \cup \{x\} \in \mathcal{F}$.

Lassen wir (M3) weg, so haben wir ein Unabhängigkeitssystem. Diese wurden in den Abschnitten 13.1 und 13.4 besprochen. Nun lassen wir stattdessen (M2) weg:

Definition 14.1. Ein **Greedoid** ist ein Mengensystem (E, \mathcal{F}) mit den Eigenschaften (M1) und (M3).

Anstatt der Subkclusionseigenschaft (M2) haben wir Erreichbarkeit: Wir nennen ein Mengensystem (E, \mathcal{F}) **erreichbar**, falls $\emptyset \in \mathcal{F}$ und es für jedes $X \in \mathcal{F} \setminus \{\emptyset\}$ ein $x \in X$ mit $X \setminus \{x\} \in \mathcal{F}$ gibt. Greedoide sind erreichbare Mengensysteme (die Erreichbarkeit folgt direkt aus (M1) und (M3)). Obwohl Greedoide verallgemeinerte Matroide sind, beinhalten sie eine reiche Struktur und verallgemeinern ihrerseits

viele verschiedene, auf den ersten Blick nicht verwandte Begriffe. Wir beginnen mit dem folgenden Resultat:

Satz 14.2. *Sei (E, \mathcal{F}) ein erreichbares Mengensystem. Die beiden folgenden Aussagen sind äquivalent:*

- (a) *Für $X \subseteq Y \subset E$ und $z \in E \setminus Y$ mit $X \cup \{z\} \in \mathcal{F}$ und $Y \in \mathcal{F}$ gilt $Y \cup \{z\} \in \mathcal{F}$;*
- (b) *\mathcal{F} ist abgeschlossen unter Vereinigungsbildung.*

Beweis: (a) \Rightarrow (b): Seien $X, Y \in \mathcal{F}$; wir beweisen $X \cup Y \in \mathcal{F}$. Sei Z eine inklusionsmaximale Menge mit $Z \in \mathcal{F}$ und $X \subseteq Z \subseteq X \cup Y$. Angenommen, es wäre $Y \setminus Z \neq \emptyset$. Mittels wiederholter Anwendung der Erreichbarkeitseigenschaft auf Y erhalten wir eine Menge $Y' \in \mathcal{F}$ mit $Y' \subseteq Z$ und ein Element $y \in Y \setminus Z$ mit $Y' \cup \{y\} \in \mathcal{F}$. Wenden wir nun (a) auf Z, Y' und y an, so folgt $Z \cup \{y\} \in \mathcal{F}$, im Widerspruch zur Wahl von Z .

(b) \Rightarrow (a): Diese Richtung ist trivial. \square

Hat (E, \mathcal{F}) die in Satz 14.2 angegebenen Eigenschaften, so heißt (E, \mathcal{F}) ein **Antimatroid**.

Proposition 14.3. *Jedes Antimatroid ist ein Greedoid.*

Beweis: Sei (E, \mathcal{F}) ein Antimatroid, d. h. (E, \mathcal{F}) ist erreichbar und \mathcal{F} ist abgeschlossen unter Vereinigungsbildung. Zum Beweis von (M3), seien $X, Y \in \mathcal{F}$ mit $|X| > |Y|$. Da (E, \mathcal{F}) erreichbar ist, gibt es eine Reihenfolge $X = \{x_1, \dots, x_n\}$ mit $\{x_1, \dots, x_i\} \in \mathcal{F}$ für $i = 0, \dots, n$. Sei $i \in \{1, \dots, n\}$ der kleinste Index mit $x_i \notin Y$; dann folgt $Y \cup \{x_i\} = Y \cup \{x_1, \dots, x_i\} \in \mathcal{F}$ (da \mathcal{F} abgeschlossen unter Vereinigungsbildung ist). \square

Es gibt eine äquivalente Definition von Antimatroiden über einen Abschlussoperator:

Proposition 14.4. *Sei (E, \mathcal{F}) ein Mengensystem mit $\emptyset \in \mathcal{F}$ und \mathcal{F} abgeschlossen unter Vereinigungsbildung. Ferner sei*

$$\tau(A) := \bigcap \{X \subseteq E : A \subseteq X, E \setminus X \in \mathcal{F}\}.$$

Dann ist τ ein Abschlussoperator, d. h. τ erfüllt die in Satz 13.11 angegebenen Bedingungen (S1)–(S3).

Beweis: Sei $X \subseteq Y \subseteq E$. Es folgt trivialerweise, dass $X \subseteq \tau(X) \subseteq \tau(Y)$. Zum Beweis von (S3): Angenommen, es gäbe ein $y \in \tau(\tau(X)) \setminus \tau(X)$. Dann haben wir $y \in Y$ für alle $Y \subseteq E$ mit $\tau(X) \subseteq Y$ und $E \setminus Y \in \mathcal{F}$. Es gibt aber ein $Z \subseteq E \setminus \{y\}$ mit $X \subseteq Z$ und $E \setminus Z \in \mathcal{F}$. Daraus folgt $\tau(X) \not\subseteq Z$, ein Widerspruch. \square

Satz 14.5. *Sei (E, \mathcal{F}) ein Mengensystem mit $\emptyset \in \mathcal{F}$ und \mathcal{F} abgeschlossen unter Vereinigungsbildung. Es ist (E, \mathcal{F}) genau dann erreichbar, wenn der in Proposition 14.4 definierte Abschlussoperator τ die folgende Anti-Austausch-Eigenschaft besitzt: Für $X \subseteq E$ und $y, z \in E \setminus \tau(X)$ mit $y \neq z$ und $z \in \tau(X \cup \{y\})$ gilt $y \notin \tau(X \cup \{z\})$.*

Beweis: Ist (E, \mathcal{F}) erreichbar, so gilt (M3) nach Proposition 14.3. Zum Beweis der Anti-Austausch-Eigenschaft, sei $X \subseteq E$, $B := E \setminus \tau(X)$ und $y, z \in B$ mit $z \notin A := E \setminus \tau(X \cup \{y\})$. Beachte, dass $A \in \mathcal{F}$, $B \in \mathcal{F}$ und $A \subseteq B \setminus \{y, z\}$.

Wenden wir (M3) auf A und B an, so erhalten wir ein Element $b \in B \setminus A \subseteq E \setminus (X \cup A)$ mit $A \cup \{b\} \in \mathcal{F}$. Es ist $A \cup \{b\}$ nicht Teilmenge von $E \setminus (X \cup \{y\})$, denn sonst wäre $\tau(X \cup \{y\}) \subseteq E \setminus (A \cup \{b\})$, im Widerspruch zu $\tau(X \cup \{y\}) = E \setminus A$. Somit ist $b = y$. Also haben wir $A \cup \{y\} \in \mathcal{F}$, und daraus folgt $\tau(X \cup \{z\}) \subseteq E \setminus (A \cup \{y\})$. Damit haben wir bewiesen, dass $y \notin \tau(X \cup \{z\})$.

Zum Beweis der Umkehrung, sei $A \in \mathcal{F} \setminus \{\emptyset\}$ und $X := E \setminus A$. Wir haben $\tau(X) = X$. Sei $a \in A$ mit $|\tau(X \cup \{a\})|$ minimal. Wir behaupten, dass $\tau(X \cup \{a\}) = X \cup \{a\}$, d. h. $A \setminus \{a\} \in \mathcal{F}$.

Angenommen, es gelte das Gegenteil: Es gibt $b \in \tau(X \cup \{a\}) \setminus (X \cup \{a\})$. Nach der Anti-Austausch-Eigenschaft haben wir $a \notin \tau(X \cup \{b\})$. Ferner gilt

$$\tau(X \cup \{b\}) \subseteq \tau(\tau(X \cup \{a\}) \cup \{b\}) = \tau(\tau(X \cup \{a\})) = \tau(X \cup \{a\}).$$

Damit ist $\tau(X \cup \{b\})$ eine echte Teilmenge von $\tau(X \cup \{a\})$, im Widerspruch zur Wahl von a . \square

Die in Satz 14.5 angegebene Anti-Austausch-Eigenschaft unterscheidet sich von (S4). Während (S4) aus Satz 13.11 eine Eigenschaft von linearen Hüllen im \mathbb{R}^n ist, betrifft die Anti-Austausch-Eigenschaft konvexe Hüllen im \mathbb{R}^n : Ist $y \neq z$, $z \notin \text{conv}(X)$ und $z \in \text{conv}(X \cup \{y\})$, so ist offensichtlich $y \notin \text{conv}(X \cup \{z\})$. Somit folgt für eine beliebige endliche Menge $E \subset \mathbb{R}^n$, dass $(E, \{X \subseteq E : X \cap \text{conv}(E \setminus X) = \emptyset\})$ ein Antimatroid ist.

Greedoide verallgemeinern sowohl Matroide als auch Antimatroide, sie besitzen aber auch weitere interessante Strukturen. Ein Beispiel ist die in EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS verwendete Blütenstruktur (Aufgabe 1). Ein weiteres grundlegendes Beispiel ist:

Proposition 14.6. *Sei G ein Graph (gerichtet oder ungerichtet) und $r \in V(G)$. Sei \mathcal{F} die Familie der Kantenmengen aller Arboreszenzen in G mit Wurzel r , oder aller r enthaltenden Bäume in G (nicht unbedingt aufspannend). Dann ist $(E(G), \mathcal{F})$ ein Greedoid.*

Beweis: Es ist (M1) trivial. Wir beweisen nun (M3) für den gerichteten Fall; der Beweis für den ungerichteten Fall folgt analog. Seien (X_1, F_1) und (X_2, F_2) zwei Arboreszenzen in G mit Wurzel r und mit $|F_1| > |F_2|$. Dann folgt $|X_1| = |F_1| + 1 > |F_2| + 1 = |X_2|$, also sei $x \in X_1 \setminus X_2$. Der r - x -Weg in (X_1, F_1) enthält eine Kante (v, w) mit $v \in X_2$ und $w \notin X_2$. Diese Kante kann zu (X_2, F_2) hinzugefügt werden. Daraus folgt $F_2 \cup \{(v, w)\} \in \mathcal{F}$. \square

Dieses Greedoid heißt das gerichtete (ungerichtete) Branching-Greedoid von G .

Das Problem der Bestimmung eines aufspannenden Baumes maximalen Gewichtes in einem zusammenhängenden Graphen G mit nichtnegativen Gewichten ist das MAXIMIERUNGSPROBLEM für das Kreismatroid $\mathcal{M}(G)$. Der BEST-IN-GREEDY-ALGORITHMUS ist in diesem Fall nichts anderes als KRUSKALS ALGORITHMUS. Hier haben wir eine zweite Formulierung desselben Problems: Bestimme eine Menge F maximalen Gewichtes mit $F \in \mathcal{F}$, wobei $(E(G), \mathcal{F})$ das ungerichtete Branching-Greedoid von G ist.

Wir werden nun einen allgemeinen Greedy-Algorithmus für Greedoide formulieren. Für den Spezialfall Matroide ist dieser gerade der in Abschnitt 13.4 besprochene BEST-IN-GREEDY-ALGORITHMUS. Für den Fall eines ungerichteten Branching-Greedoide mit einer modularen Kostenfunktion c ist er PRIMS ALGORITHMUS:

GREEDY-ALGORITHMUS FÜR GREEDOIDE

Input: Ein Greedoid (E, \mathcal{F}) und eine Funktion $c : 2^E \rightarrow \mathbb{R}$, gegeben durch ein Orakel, welches für ein gegebenes $X \subseteq E$ entscheidet, ob $X \in \mathcal{F}$, und $c(X)$ liefert.
Output: Eine Menge $F \in \mathcal{F}$.

-
- ① Setze $F := \emptyset$.
 - ② Sei $e \in E \setminus F$ mit $F \cup \{e\} \in \mathcal{F}$ und maximalem $c(F \cup \{e\})$;
If es gibt kein solches e **then stop**.
 - ③ Setze $F := F \cup \{e\}$ und **go to** ②.
-

Dieser Algorithmus liefert keineswegs immer optimale Lösungen, sogar für modulare Kostenfunktionen c nicht. Wenigstens können wir aber diejenigen Greedoide charakterisieren, für die er optimale Lösungen liefert:

Satz 14.7. *Sei (E, \mathcal{F}) ein Greedoid. Der GREEDY-ALGORITHMUS FÜR GREEDOIDE bestimmt eine Menge $F \in \mathcal{F}$ maximalen Gewichtes für jede modulare Kostenfunktion $c : 2^E \rightarrow \mathbb{R}_+$ genau dann, wenn (E, \mathcal{F}) die sogenannte starke Austauscheigenschaft hat: Für alle $A, B \in \mathcal{F}$ mit $A \subseteq B$ und B inklusionsmaximal und alle $x \in E \setminus B$ mit $A \cup \{x\} \in \mathcal{F}$ gibt es ein $y \in B \setminus A$ mit $A \cup \{y\} \in \mathcal{F}$ und $(B \setminus \{y\}) \cup \{x\} \in \mathcal{F}$.*

Beweis: Sei (E, \mathcal{F}) ein Greedoid mit der starken Austauscheigenschaft. Sei $c : E \rightarrow \mathbb{R}_+$ und $A = \{a_1, \dots, a_l\}$ die von dem GREEDY-ALGORITHMUS FÜR GREEDOIDE gefundene Lösung, wobei die Elemente in der Reihenfolge a_1, \dots, a_l gewählt wurden.

Sei $B = \{a_1, \dots, a_k\} \dot{\cup} B'$ eine optimale Lösung mit k maximal. Angenommen, es sei $k < l$. Wenden wir die starke Austauscheigenschaft auf $\{a_1, \dots, a_k\}$, B und a_{k+1} an, so folgt: Es gibt ein $y \in B'$ mit $\{a_1, \dots, a_k, y\} \in \mathcal{F}$ und $(B \setminus y) \cup \{a_{k+1}\} \in \mathcal{F}$. Nach Wahl von a_{k+1} in ② des GREEDY-ALGORITHMUS FÜR GREEDOIDE haben wir $c(a_{k+1}) \geq c(y)$, woraus $c((B \setminus y) \cup \{a_{k+1}\}) \geq c(B)$ folgt, im Widerspruch zur Wahl von B .

Sei umgekehrt (E, \mathcal{F}) ein Greedoid, welches die starke Austauscheigenschaft nicht besitzt, d. h. es gibt $A, B \in \mathcal{F}$ mit B inklusionsmaximal in \mathcal{F} und $A \subseteq B$ und ein $x \in E \setminus B$ mit $A \cup \{x\} \in \mathcal{F}$, so dass für alle $y \in B \setminus A$ mit $A \cup \{y\} \in \mathcal{F}$ folgt: $(B \setminus y) \cup \{x\} \notin \mathcal{F}$.

Sei $Y := \{y \in B \setminus A : A \cup \{y\} \in \mathcal{F}\}$. Setze $c(e) := 2$ für $e \in B \setminus Y$, $c(e) := 1$ für $e \in Y \cup \{x\}$ und $c(e) := 0$ für $e \in E \setminus (B \cup \{x\})$. Dann könnte der GREEDY-ALGORITHMUS FÜR GREEDOIDE die Elemente von A zuerst wählen (diese haben Gewicht 2) und danach x . Irgendwann terminiert er dann mit einer Menge $F \in \mathcal{F}$, die nicht optimal sein kann, da $c(F) \leq c(B \cup \{x\}) - 2 < c(B \cup \{x\}) - 1 = c(B)$ und $B \in \mathcal{F}$. \square

In der Tat ist die Optimierung modularer Funktionen über allgemeinen Greedoiden NP-schwer. Dies folgt aus der folgenden Tatsache, zusammen mit Korollar 15.24:

Proposition 14.8. *Das Problem, für einen gegebenen ungerichteten Graphen G und $k \in \mathbb{N}$ zu entscheiden, ob G eine Knotenüberdeckung der Kardinalität k besitzt, kann linear auf das folgende Problem zurückgeführt werden: Für ein Greedoid (E, \mathcal{F}) (gegeben durch ein Mitglieds-Orakel) und eine Funktion $c : E \rightarrow \mathbb{R}_+$ bestimme man ein $F \in \mathcal{F}$ mit maximalem $c(F)$.*

Beweis: Sei G ein ungerichteter Graph und $k \in \mathbb{N}$. Setze $D := V(G) \dot{\cup} E(G)$ und

$$\mathcal{F} := \{X \subseteq D : \text{für jedes } e = \{v, w\} \in E(G) \cap X \text{ gilt } v \in X \text{ oder } w \in X\}.$$

Es ist (D, \mathcal{F}) ein Antimatroid: Es ist erreichbar und \mathcal{F} ist abgeschlossen unter Vereinigungsbildung. Insbesondere ist (D, \mathcal{F}) ein Greedoid nach Proposition 14.3.

Nun betrachten wir $\mathcal{F}' := \{X \in \mathcal{F} : |X| \leq |E(G)| + k\}$. Da (M1) und (M3) erhalten bleiben, ist (D, \mathcal{F}') auch ein Greedoid. Setze $c(e) := 1$ für $e \in E(G)$ und $c(v) := 0$ für $v \in V(G)$. Dann folgt: Es gibt eine Menge $F \in \mathcal{F}'$ mit $c(F) = |E(G)|$ genau dann, wenn G eine Knotenüberdeckung der Größe k enthält. \square

Andererseits gibt es interessante Funktionen, die über allgemeinen Greedoiden maximiert werden können, z. B. Bottleneck-Funktionen $c(F) := \min\{c'(e) : e \in F\}$ für irgendeine $c' : E \rightarrow \mathbb{R}_+$ über den inklusionsmaximalen Elementen von \mathcal{F} (Aufgabe 2). Siehe das Buch von Korte, Lovász und Schrader [1991] für weitere Resultate auf diesem Gebiet.

14.2 Polymatroide

Aus Satz 13.10 ist uns die enge Beziehung zwischen Matroiden und submodularen Funktionen bekannt. Submodulare Funktionen definieren die folgende interessante Klasse von Polyedern:

Definition 14.9. Ein Polymatroid ist ein Polytop des Typs

$$P(f) := \left\{ x \in \mathbb{R}^E : x \geq 0, \sum_{e \in A} x_e \leq f(A) \text{ für alle } A \subseteq E \right\},$$

wobei E eine endliche Menge und $f : 2^E \rightarrow \mathbb{R}_+$ eine submodulare Funktion ist.

Es ist nicht schwer zu sehen, dass man f für jedes Polymatroid so wählen kann, dass $f(\emptyset) = 0$ und f monoton ist (Aufgabe 6; eine Funktion $f : 2^E \rightarrow \mathbb{R}$ heißt **monoton**, falls $f(X) \leq f(Y)$ für $X \subseteq Y \subseteq E$). Edmonds' ursprüngliche Definition war anders; siehe Aufgabe 7. Ferner weisen wir darauf hin, dass man gelegentlich das Paar (E, f) ein Polymatroid nennt und nicht das Polytop.

Ist f die Rangfunktion eines Matroids, so ist $P(f)$ die konvexe Hülle der Inzidenzvektoren der unabhängigen Mengen dieses Matroids (Satz 13.21). Wir wissen bereits, dass der BEST-IN-GREEDY jede lineare Funktion über einem Matroid-Polytop optimiert. Es gibt einen ähnlichen Greedy-Algorithmus für allgemeine Polymatroide. Wir setzen nun voraus, dass f monoton ist:

POLYMATROID-GREEDY-ALGORITHMUS

Input: Eine endliche Menge E und eine submodulare monotone Funktion $f : 2^E \rightarrow \mathbb{R}_+$ mit $f(\emptyset) \geq 0$ (gegeben durch ein Orakel). Ein Vektor $c \in \mathbb{R}^E$.

Output: Ein Vektor $x \in P(f)$ mit maximalem cx .

-
- ① Sortiere $E = \{e_1, \dots, e_n\}$ so, dass $c(e_1) \geq \dots \geq c(e_k) > 0 \geq c(e_{k+1}) \geq \dots \geq c(e_n)$.
 - ② If $k \geq 1$ then setze $x(e_1) := f(\{e_1\})$.
Setze $x(e_i) := f(\{e_1, \dots, e_i\}) - f(\{e_1, \dots, e_{i-1}\})$ für $i = 2, \dots, k$.
Setze $x(e_i) := 0$ für $i = k+1, \dots, n$.
-

Proposition 14.10. Sei $E = \{e_1, \dots, e_n\}$ und $f : 2^E \rightarrow \mathbb{R}$ eine submodulare Funktion mit $f(\emptyset) \geq 0$. Sei ferner $b : E \rightarrow \mathbb{R}$ mit $b(e_1) \leq f(\{e_1\})$ und $b(e_i) \leq f(\{e_1, \dots, e_i\}) - f(\{e_1, \dots, e_{i-1}\})$ für $i = 2, \dots, n$. Dann ist $\sum_{a \in A} b(a) \leq f(A)$ für alle $A \subseteq E$.

Beweis: Der Beweis erfolgt mittels Induktion über $i = \max\{j : e_j \in A\}$. Die Aussage ist trivial für $A = \emptyset$ und $A = \{e_1\}$. Ist $i \geq 2$, so folgt $\sum_{a \in A} b(a) = \sum_{a \in A \setminus \{e_i\}} b(a) + b(e_i) \leq f(A \setminus \{e_i\}) + b(e_i) \leq f(A \setminus \{e_i\}) + f(\{e_1, \dots, e_i\}) - f(\{e_1, \dots, e_{i-1}\}) \leq f(A)$, wobei die erste Ungleichung nach der Induktionsvoraussetzung und die dritte aus der Submodularität folgt. \square

Satz 14.11. Der POLYMATROID-GREEDY-ALGORITHMUS bestimmt ein $x \in P(f)$ mit maximalem cx korrekt. Ist f ganzzahlig, so auch x .

Beweis: Sei $x \in \mathbb{R}^E$ der Output des POLYMATROID-GREEDY-ALGORITHMUS für E , f und c . Ist f ganzzahlig, so ist x definitionsgemäß auch ganzzahlig. Da f monoton ist, folgt $x \geq 0$, und somit ist $x \in P(f)$ nach Proposition 14.10.

Nun sei $y \in \mathbb{R}_+^E$ mit $cy > cx$. Wie im Beweis von Satz 13.19 setzen wir $d_j := c(e_j) - c(e_{j+1})$ ($j = 1, \dots, k-1$) und $d_k := c(e_k)$. Damit folgt

$$\sum_{j=1}^k d_j \sum_{i=1}^j x(e_i) = cx < cy \leq \sum_{j=1}^k c(e_j)y(e_j) = \sum_{j=1}^k d_j \sum_{i=1}^j y(e_i).$$

Da $d_j \geq 0$ für alle j , gibt es einen Index $j \in \{1, \dots, k\}$ mit $\sum_{i=1}^j y(e_i) > \sum_{i=1}^j x(e_i)$. Da aber $\sum_{i=1}^j x(e_i) = f(\{e_1, \dots, e_j\})$, folgt $y \notin P(f)$. \square

Wie bei Matroiden, können wir auch den Schnitt zweier Polymatroiden betrachten. Der folgende Polymatroidschnitt-Satz hat viele Auswirkungen:

Satz 14.12. (Edmonds [1970,1979]) *Sei E eine endliche Menge und seien $f, g : 2^E \rightarrow \mathbb{R}_+$ submodulare Funktionen. Dann ist das folgende System TDI:*

$$\begin{aligned} x &\geq 0 \\ \sum_{e \in A} x_e &\leq f(A) \quad (A \subseteq E) \\ \sum_{e \in A} x_e &\leq g(A) \quad (A \subseteq E). \end{aligned}$$

Beweis: Betrachte das folgende primal-duale LP-Paar:

$$\max \left\{ cx : x \geq 0, \sum_{e \in A} x_e \leq f(A) \text{ und } \sum_{e \in A} x_e \leq g(A) \text{ für alle } A \subseteq E \right\}$$

und

$$\min \left\{ \sum_{A \subseteq E} (f(A)y_A + g(A)z_A) : y, z \geq 0, \sum_{A \subseteq E, e \in A} (y_A + z_A) \geq c_e \text{ für alle } e \in E \right\}.$$

Zum Beweis der Aussage benutzen wir Lemma 5.24.

Sei $c : E(G) \rightarrow \mathbb{Z}$ und y, z eine optimale duale Lösung mit

$$\sum_{A \subseteq E} (y_A + z_A)|A||E \setminus A| \tag{14.1}$$

so klein wie möglich. Wir behaupten, dass $\mathcal{F} := \{A \subseteq E : y_A > 0\}$ eine Kette ist, d.h. für alle $A, B \in \mathcal{F}$ gilt entweder $A \subseteq B$ oder $B \subseteq A$.

Zum Beweis dieser Behauptung nehmen wir an, es seien $A, B \in \mathcal{F}$ mit $A \cap B \neq \emptyset$ und $A \cap B \neq B$. Setze $\epsilon := \min\{y_A, y_B\}$, $y'_A := y_A - \epsilon$, $y'_B := y_B - \epsilon$, $y'_{A \cap B} := y_{A \cap B} + \epsilon$, $y'_{A \cup B} := y_{A \cup B} + \epsilon$ und $y'(S) := y(S)$ für alle weiteren $S \subseteq E$. Da y', z eine zulässige duale Lösung ist, ist sie auch optimal (f ist submodular). Dies widerspricht jedoch der Wahl von y , da (14.1) für y', z kleiner ist.

Analog zeigt man, dass $\mathcal{F}' := \{A \subseteq E : z_A > 0\}$ eine Kette ist. Nun seien M und M' Matrizen, deren Spalten nach den Elementen von E indiziert sind und deren Zeilen die Inzidenzvektoren der Elemente von \mathcal{F} bzw. \mathcal{F}' sind. Nach Lemma 5.24 genügt es zu zeigen, dass $(\begin{smallmatrix} M \\ M' \end{smallmatrix})$ vollständig-unimodular ist.

Hier verwenden wir Ghouila-Houris Satz 5.25. Sei \mathcal{R} eine Zeilenmenge, etwa $\mathcal{R} = \{A_1, \dots, A_p, B_1, \dots, B_q\}$, mit $A_1 \supseteq \dots \supseteq A_p$ und $B_1 \supseteq \dots \supseteq B_q$. Sei $\mathcal{R}_1 := \{A_i : i \text{ ungerade}\} \cup \{B_i : i \text{ gerade}\}$ und $\mathcal{R}_2 := \mathcal{R} \setminus \mathcal{R}_1$. Da für jedes $e \in E$ gilt: $\{R \in \mathcal{R} : e \in R\} = \{A_1, \dots, A_{p_e}\} \cup \{B_1, \dots, B_{q_e}\}$ für irgendein $p_e \in \{0, \dots, p\}$ und $q_e \in \{0, \dots, q\}$, ist die Summe der Zeilen in \mathcal{R}_1 minus die Summe der Zeilen in \mathcal{R}_2 ein Vektor, dessen Komponenten nur $-1, 0$, oder 1 sind. Somit ist das Kriterium von Satz 5.25 erfüllt. \square

Man kann lineare Funktionen über Polymatroidschnitten optimieren. Dies ist jedoch nicht so einfach wie bei einem einzelnen Polymatroid. Wir können aber die ELLIPSOIDMETHODE verwenden, falls wir das SEPARATIONS-PROBLEM für jedes Polymatroid lösen können. Diese Frage werden wir in Abschnitt 14.3 wieder aufgreifen.

Korollar 14.13. (Edmonds [1970]) *Seien (E, \mathcal{M}_1) und (E, \mathcal{M}_2) zwei Matroide mit den Rangfunktionen r_1 und r_2 . Dann ist die konvexe Hülle der Inzidenzvektoren der Elemente von $\mathcal{M}_1 \cap \mathcal{M}_2$ das Polytop*

$$\left\{ x \in \mathbb{R}_+^E : \sum_{e \in A} x_e \leq \min\{r_1(A), r_2(A)\} \text{ für alle } A \subseteq E \right\}.$$

Beweis: Da r_1 und r_2 nach Satz 13.10 nichtnegativ und submodular sind, ist das obige Ungleichungssystem nach Satz 14.12 TDI. Da r_1 und r_2 ganzzahlig sind, ist das Polytop nach Korollar 5.16 ganzzahlig. Da $r_1(A) \leq |A|$ für alle $A \subseteq E$, sind die Ecken (deren konvexe Hülle nach Korollar 3.32 gleich dem Polytop ist) 0-1-Vektoren und folglich Inzidenzvektoren von gemeinsamen unabhängigen Mengen (Elemente aus $\mathcal{M}_1 \cap \mathcal{M}_2$). Andererseits erfüllt jeder solche Inzidenzvektor die Ungleichungen (nach Definition der Rangfunktionen). \square

Natürlich folgt hieraus die Beschreibung des Matroid-Polytops (Satz 13.21), indem man $\mathcal{M}_1 = \mathcal{M}_2$ setzt. Aus Satz 14.12 folgen einige weitere Resultate:

Korollar 14.14. (Edmonds [1970]) *Sei E eine endliche Menge und seien $f, g : 2^E \rightarrow \mathbb{R}_+$ monotone submodulare Funktionen mit $f(\emptyset) = g(\emptyset) = 0$. Dann gilt*

$$\max\{\mathbf{1}x : x \in P(f) \cap P(g)\} = \min_{A \subseteq E} (f(A) + g(E \setminus A)).$$

Ferner gilt: Sind f und g ganzzahlig, so gibt es ein ganzzahliges das Maximum annehmendes x .

Beweis: Nach Satz 14.12 hat das Dual von

$$\max\{\mathbf{1}x : x \in P(f) \cap P(g)\},$$

nämlich

$$\min \left\{ \sum_{A \subseteq E} (f(A)y_A + g(A)z_A) : y, z \geq 0, \sum_{A \subseteq E, e \in A} (y_A + z_A) \geq 1 \text{ für alle } e \in E \right\},$$

eine ganzzahlige optimale Lösung y, z . Sei $B := \bigcup_{A: y_A \geq 1} A$ und $C := \bigcup_{A: z_A \geq 1} A$. Es folgt, dass $B \cup C = E$. Da f und g submodular und nichtnegativ sind und $f(\emptyset) = g(\emptyset) = 0$, haben wir

$$\sum_{A \subseteq E} (f(A)y_A + g(A)z_A) \geq f(B) + g(C).$$

Da $E \setminus B \subseteq C$ und g monoton ist, beträgt die rechte Seite mindestens $f(B) + g(E \setminus B)$, womit „ \geq “ der Aussage bewiesen ist.

Die andere Richtung, nämlich „ \leq “ der Aussage, ist trivial, da wir für jedes $A \subseteq E$ eine zulässige duale Lösung y, z bekommen, indem wir $y_A := 1, z_{E \setminus A} := 1$ und alle weiteren Komponenten gleich Null setzen.

Die Ganzzahligkeit folgt direkt aus Satz 14.12 und Korollar 5.16. \square

Satz 13.31 ist ein Spezialfall. Ferner erhalten wir:

Korollar 14.15. (Frank [1982]) *Sei E eine endliche Menge und seien $f, g : 2^E \rightarrow \mathbb{R}$ mit f supermodular, g submodular und $f \leq g$. Dann gibt es eine modulare Funktion $h : 2^E \rightarrow \mathbb{R}$ mit $f \leq h \leq g$. Sind f und g ganzzahlig, so kann h ganzzahlig gewählt werden.*

Beweis: O. B. d. A. gilt $f(\emptyset) = g(\emptyset)$ und $f(E) = g(E)$. Sei $M := 2 \max\{|f(A)| + |g(A)| : A \subseteq E\}$. Setze $f'(A) := g(E) - f(E \setminus A) + M|A|$ und $g'(A) := g(A) - f(\emptyset) + M|A|$ für alle $A \subseteq E$. Es sind f' und g' nichtnegativ, submodular und monoton. Wenden wir Korollar 14.14 an, so bekommen wir

$$\begin{aligned} & \max\{\mathbb{1}_x : x \in P(f') \cap P(g')\} \\ &= \min_{A \subseteq E} (f'(A) + g'(E \setminus A)) \\ &= \min_{A \subseteq E} (g(E) - f(E \setminus A) + M|A| + g(E \setminus A) - f(\emptyset) + M|E \setminus A|) \\ &\geq g(E) - f(\emptyset) + M|E|. \end{aligned}$$

Sei also $x \in P(f') \cap P(g')$ mit $\mathbb{1}_x = g(E) - f(\emptyset) + M|E|$. Sind f und g ganzzahlig, so kann x ganzzahlig gewählt werden. Sei $h'(A) := \sum_{e \in A} x_e$ und $h(A) := h'(A) + f(\emptyset) - M|A|$ für alle $A \subseteq E$. Die Funktion h ist modular. Ferner gilt für alle $A \subseteq E$, dass $h(A) \leq g'(A) + f(\emptyset) - M|A| = g(A)$ und $h(A) = \mathbb{1}_x - h'(E \setminus A) + f(\emptyset) - M|A| \geq g(E) + M|E| - M|A| - f'(E \setminus A) = f(A)$. \square

Die Ähnlichkeit mit konvexen und konkaven Funktionen ist offensichtlich; siehe auch Aufgabe 10.

14.3 Die Minimierung submodularer Funktionen

Das SEPARATIONS-PROBLEM für ein Polymatroid $P(f)$ und einen Vektor x lautet: Bestimme eine Menge A mit $f(A) < \sum_{e \in A} x(e)$. Somit reduziert sich dieses Problem auf die Bestimmung einer Menge A , die $g(A)$ minimiert, wobei $g(A) := f(A) - \sum_{e \in A} x(e)$. Beachte: Ist f submodular, so ist g auch submodular. Dadurch wird die Minimierung submodularer Funktionen zu einem interessanten Problem.

Eine weitere Motivation wäre vielleicht, dass die submodularen Funktionen als eine diskrete Version der konvexen Funktionen betrachtet werden können (Korollar 14.15 und Aufgabe 10). Wir haben in Abschnitt 8.7 bereits einen Spezialfall gelöst: Die Bestimmung eines kapazitätsminimalen Schnittes in einem ungerichteten Graphen kann als die Minimierung einer bestimmten symmetrischen submodularen Funktion $f : 2^U \rightarrow \mathbb{R}_+$ über $2^U \setminus \{\emptyset, U\}$ aufgefasst werden. Zunächst werden wir zeigen, wie man allgemeine submodulare Funktionen minimiert, bevor wir zu diesem Spezialfall zurückkehren. Der Einfachheit halber beschränken wir uns auf submodulare Funktionen mit ganzzahligen Werten.

MINIMIERUNGSPROBLEM SUBMODULARER FUNKTIONEN

Instanz: Eine endliche Menge U . Eine submodulare Funktion $f : 2^U \rightarrow \mathbb{R}$ (gegeben durch ein Orakel).

Aufgabe: Bestimme eine Teilmenge $X \subseteq U$ mit minimalem $f(X)$.

Grötschel, Lovász und Schrijver [1981] haben gezeigt, wie dieses Problem mit der ELLIPSOIDMETHODE gelöst werden kann. Die Idee ist, das Minimum mittels binärer Suche zu bestimmen; dies führt das Problem auf das SEPARATIONS-PROBLEM für ein Polymatroid zurück. Wegen der Äquivalenz von Separation und Optimierung (Abschnitt 4.6) genügt es also, lineare Funktionen über Polymatroiden zu optimieren. Dies kann aber leicht mit dem POLYMATROID-GREEDY-ALGORITHMUS bewerkstelligt werden. Als erstes benötigen wir eine obere Schranke für $|f(S)|$ mit $S \subseteq U$:

Proposition 14.16. Für jede submodulare Funktion $f : 2^U \rightarrow \mathbb{R}$ und jedes $S \subseteq U$ haben wir

$$f(U) - \sum_{u \in U} \max\{0, f(\{u\}) - f(\emptyset)\} \leq f(S) \leq f(\emptyset) + \sum_{u \in U} \max\{0, f(\{u\}) - f(\emptyset)\}.$$

Insbesondere kann man eine Zahl B mit $|f(S)| \leq B$ für alle $S \subseteq U$ in linearer Zeit berechnen, mit $|U| + 2$ Orakelaufrufen für f .

Beweis: Durch wiederholte Anwendung der Submodularität bekommen wir für $\emptyset \neq S \subseteq U$ (mit $x \in S$):

$$f(S) \leq -f(\emptyset) + f(S \setminus \{x\}) + f(\{x\}) \leq \dots \leq -|S|f(\emptyset) + f(\emptyset) + \sum_{x \in S} f(\{x\}),$$

und für $S \subset U$ (mit $y \in U \setminus S$):

$$\begin{aligned} f(S) &\geq -f(\{y\}) + f(S \cup \{y\}) + f(\emptyset) \geq \dots \\ &\geq - \sum_{y \in U \setminus S} f(\{y\}) + f(U) + |U \setminus S| f(\emptyset). \end{aligned}$$

□

Für den Rest dieses Abschnittes beschränken wir uns auf ganzzahlige Funktionen und zeigen, wie man das MINIMIERUNGSPROBLEM SUBMODULARER FUNKTIONEN löst mittels der Äquivalenz von Optimierung und Separation in der ungewöhnlichen Richtung.

Proposition 14.17. *Das folgende Problem kann in polynomieller Zeit gelöst werden: Für eine gegebene endliche Menge U , eine monotone submodulare Funktion $f : 2^U \rightarrow \mathbb{Z}_+$ (mittels Orakel) mit $f(S) > 0$ für $S \neq \emptyset$, eine Zahl $B \in \mathbb{N}$ mit $f(S) \leq B$ für alle $S \subseteq U$ und einen Vektor $x \in \mathbb{Z}_+^U$, entscheide man, ob $x \in P(f)$, oder lieferne eine Menge $S \subseteq U$ mit $\sum_{v \in S} x(v) > f(S)$.*

Beweis: Dies ist das SEPARATIONS-PROBLEM für das Polymatroid $P(f)$. Wir werden Satz 4.23 benutzen, da wir das Optimierungsproblem für $P(f)$ bereits gelöst haben: Der POLYMATROID-GREEDY-ALGORITHMUS maximiert jede lineare Funktion über $P(f)$ (Satz 14.11).

Wir müssen die Voraussetzungen für Satz 4.23 prüfen. Da der Nullvektor und die Einheitsvektoren alle in $P(f)$ liegen, können wir $x_0 := \epsilon \mathbf{1}$ als einen inneren Punkt nehmen, wobei $\epsilon = \frac{1}{|U|+1}$. Es ist $\text{size}(x_0) = O(|U| \log |U|)$. Ferner wird jede Ecke von $P(f)$ durch den POLYMATROID-GREEDY-ALGORITHMUS erzeugt (für irgendeine Zielfunktion; siehe Satz 14.11) und hat somit $\text{size } O(|U|(2 + \log B))$. Damit folgt, dass das SEPARATIONS-PROBLEM in polynomieller Zeit gelöst werden kann. Mit Satz 4.23 erhalten wir eine facettenbestimmende Ungleichung für $P(f)$, die von x verletzt wird, falls $x \notin P(f)$. Dies entspricht einer Menge $S \subseteq U$ mit $\sum_{v \in S} x(v) > f(S)$. □

Wenn f nicht monoton ist, können wir dieses Resultat nicht direkt anwenden. Stattdessen betrachten wir eine andere Funktion:

Proposition 14.18. *Sei $f : 2^U \rightarrow \mathbb{R}$ eine submodulare Funktion und $\beta \in \mathbb{R}$. Dann ist die Funktion $g : 2^U \rightarrow \mathbb{R}$, gegeben durch*

$$g(X) := f(X) - \beta + \sum_{e \in X} (f(U \setminus \{e\}) - f(U)),$$

submodular und monoton.

Beweis: Die Submodularität von g folgt direkt aus der Submodularität von f . Um zu zeigen, dass g monoton ist, sei $X \subset U$ und $e \in U \setminus X$. Dann folgt $g(X \cup \{e\}) - g(X) = f(X \cup \{e\}) - f(X) + f(U \setminus \{e\}) - f(U) \geq 0$, da f submodular ist. □

Satz 14.19. *Ganzzahlige submodulare Funktionen $f : 2^U \rightarrow \mathbb{Z}$ (gegeben durch ein Orakel) können in Laufzeit, die polynomiell in $|U| + \log \max\{|f(S)| : S \subseteq U\}$ ist, gelöst werden.*

Beweis: Sei U eine endliche Menge und angenommen, f wird durch ein Orakel gegeben. Berechne zunächst eine Zahl $B \in \mathbb{N}$ mit $\max\{|f(S)| : S \subseteq U\} \leq B \leq (2|U| + 1) \max\{|f(S)| : S \subseteq U\}$ (siehe Proposition 14.16). Da f submodular ist, folgt für jedes $e \in U$ und jedes $X \subseteq U \setminus \{e\}$:

$$f(\{e\}) - f(\emptyset) \geq f(X \cup \{e\}) - f(X) \geq f(U) - f(U \setminus \{e\}). \quad (14.2)$$

Gilt $f(\{e\}) - f(\emptyset) \leq 0$ für irgendein $e \in U$, so gibt es nach (14.2) eine optimale e enthaltende Menge S . In diesem Fall betrachten wir die durch $U' := U \setminus \{e\}$ und $f'(X) := f(X \cup \{e\})$ für $X \subseteq U \setminus \{e\}$ gegebene Instanz (U', B, f') , bestimmen eine Menge $S' \subseteq U'$ mit minimalem $f'(S')$ und bekommen $S := S' \cup \{e\}$ als Output.

Ähnlich verfahren wir, falls $f(U) - f(U \setminus \{e\}) \geq 0$: Dann gibt es nach (14.2) eine optimale e nicht enthaltende Menge S . In diesem Fall minimieren wir einfach f beschränkt auf $U \setminus \{e\}$. In beiden Fällen haben wir die Größe der zugrunde liegenden Menge verringert.

Somit können wir annehmen, dass $f(\{e\}) - f(\emptyset) > 0$ und $f(U \setminus \{e\}) - f(U) > 0$ für alle $e \in U$. Sei $x(e) := f(U \setminus \{e\}) - f(U)$. Für jede ganze Zahl β mit $-B \leq \beta \leq f(\emptyset)$ definieren wir $g(X) := f(X) - \beta + \sum_{e \in X} x(e)$. Nach Proposition 14.18 ist g submodular und monoton. Ferner haben wir $g(\emptyset) = f(\emptyset) - \beta \geq 0$ und $g(\{e\}) = f(\{e\}) - \beta + x(e) > 0$ für alle $e \in U$, also folgt $g(X) > 0$ für alle $\emptyset \neq X \subseteq U$. Nun wenden wir Proposition 14.17 an und prüfen, ob $x \in P(g)$. Falls ja, folgt $f(X) \geq \beta$ für alle $X \subseteq U$. Falls nein, haben wir eine Menge S mit $f(S) < \beta$.

Nun wenden wir binäre Suche an: Mit geeigneter Wahl von β bei jeder Iteration benötigen wir $O(\log(2B))$ Iterationen, um die Zahl $\beta^* \in \{-B, -B+1, \dots, f(\emptyset)\}$ zu bestimmen, für die $f(X) \geq \beta^*$ für alle $X \subseteq U$, aber $f(S) < \beta^* + 1$ für ein $S \subseteq U$ gilt. Diese Menge S minimiert f . \square

Der erste streng polynomielle Algorithmus stammt von Grötschel, Lovász und Schrijver [1988] und basiert auch auf der Ellipsoidmethode. Kombinatorische Algorithmen zur Lösung des MINIMIERUNGSPROBLEMS SUBMODULARER FUNKTIONEN in streng polynomieller Zeit sind von Schrijver [2000] und unabhängig auch von Iwata, Fleischer und Fujishige [2001], beide basierend auf Arbeiten von Cunningham [1984], beschrieben worden. Im nächsten Abschnitt werden wir Schrijvers Algorithmus betrachten.

14.4 Schrijvers Algorithmus

Gegeben sei eine endliche Menge U und eine submodulare Funktion $f : 2^U \rightarrow \mathbb{R}$. O. B. d. A. können wir annehmen, dass $U = \{1, \dots, n\}$ und $f(\emptyset) = 0$. In jedem Schritt speichert Schrijvers [2000] Algorithmus einen Punkt x in dem sogenannten, folgendermaßen definierten **Basispolyeder** von f :

$$\left\{ x \in \mathbb{R}^U : \sum_{u \in A} x(u) \leq f(A) \text{ für alle } A \subseteq U, \sum_{u \in U} x(u) = f(U) \right\}.$$

Es ist erwähnenswert, dass die Eckenmenge dieses Basispolyeders genau die Menge der Vektoren b^{\prec} für alle vollständigen Ordnungen \prec von U ist, wobei für $u \in U$

$$b^{\prec}(u) := f(\{v \in U : v \preceq u\}) - f(\{v \in U : v \prec u\}).$$

Diese Tatsache, die wir hier nicht gebrauchen werden, kann wie Satz 14.11 bewiesen werden (Aufgabe 14).

Der Punkt x wird immer als explizite Konvexitätskombination dieser Ecken geschrieben: $x = \lambda_1 b^{\prec_1} + \dots + \lambda_k b^{\prec_k}$. Anfangs kann man $k = 1$ und eine beliebige vollständige Ordnung wählen. Für eine gegebene vollständige Ordnung \prec und $s, u \in U$ bezeichne $\prec^{s,u}$ die aus \prec durch Verschiebung von u zur Stelle unmittelbar vor s hervorgehende vollständige Ordnung. Ferner bezeichne χ^u den Inzidenzvektor von u ($u \in U$).

SCHRIJVERS ALGORITHMUS

Input: Eine endliche Menge $U = \{1, \dots, n\}$. Eine submodulare Funktion $f : 2^U \rightarrow \mathbb{R}$ mit $f(\emptyset) = 0$ (gegeben durch ein Orakel).

Output: Eine Teilmenge $X \subseteq U$ mit minimalem $f(X)$.

- ① Setze $k := 1$, sei \prec_1 eine beliebige vollständige Ordnung für U , und setze $x := b^{\prec_1}$.
- ② Setze $D := (U, A)$, wobei $A = \{(u, v) : u \prec_i v \text{ für ein } i \in \{1, \dots, k\}\}$.
- ③ Sei $P := \{v \in U : x(v) > 0\}$ und $N := \{v \in U : x(v) < 0\}$, und sei X die Menge der nicht von P aus im Digraphen D erreichbaren Knoten.
If $N \subseteq X$ **then stop else** sei $d(v)$ die Distanz von P nach v in D .
- ④ Wähle den von P aus erreichbaren Knoten $t \in N$ mit $(d(t), t)$ lexikographisch maximal.
Wähle den maximalen Knoten s mit $(s, t) \in A$ und $d(s) = d(t) - 1$.
Sei $i \in \{1, \dots, k\}$ so, dass $\alpha := |\{v : s \prec_i v \preceq_i t\}|$ maximal ist (die Anzahl der dieses Maximum annehmenden Indizes wird mit β bezeichnet).
- ⑤ Berechne eine Zahl ϵ mit $0 \leq \epsilon \leq -x(t)$ und schreibe $x' := x + \epsilon(\chi^t - \chi^s)$ als explizite Konvexitätskombination von höchstens n Vektoren, gewählt aus der Vektorenmenge $b^{\prec_1}, \dots, b^{\prec_k}$ und $b^{\prec^{s,u}}$ für alle $u \in U$ mit $s \prec_i u \preceq_i t$, mit der zusätzlichen Eigenschaft, dass b^{\prec_i} nicht vorkommt, falls $x'(t) < 0$.
- ⑥ Setze $x := x'$, gebe den Vektoren in der Konvexitätskombination von x die neuen Namen $b^{\prec_1}, \dots, b^{\prec_{k'}}$, setze $k := k'$ und **go to** ②.

Satz 14.20. (Schrijver [2000]) SCHRIJVERS ALGORITHMUS arbeitet korrekt.

Beweis: Der Algorithmus terminiert, falls D keinen Weg von P nach N enthält, und liefert die Menge X der nicht von P aus erreichbaren Knoten als Output. Offensichtlich gilt $N \subseteq X \subseteq U \setminus P$, also folgt $\sum_{u \in X} x(u) \leq \sum_{u \in W} x(u)$ für jedes $W \subseteq U$. Auch endet keine Kante in X , also ist entweder $X = \emptyset$, oder es gibt für jedes $j \in \{1, \dots, k\}$ ein $v \in X$ mit $X = \{u \in U : u \preceq_j v\}$. Somit folgt

nach Definition: $\sum_{u \in X} b^{\prec_j}(u) = f(X)$ für alle $j \in \{1, \dots, k\}$. Ferner folgt mit Proposition 14.10: $\sum_{u \in W} b^{\prec_j}(u) \leq f(W)$ für alle $W \subseteq U$ und $j \in \{1, \dots, k\}$. Damit haben wir für jedes $W \subseteq U$,

$$\begin{aligned} f(W) &\geq \sum_{j=1}^k \lambda_j \sum_{u \in W} b^{\prec_j}(u) = \sum_{u \in W} \sum_{j=1}^k \lambda_j b^{\prec_j}(u) = \sum_{u \in W} x(u) \\ &\geq \sum_{u \in X} x(u) = \sum_{u \in X} \sum_{j=1}^k \lambda_j b^{\prec_j}(u) = \sum_{j=1}^k \lambda_j \sum_{u \in X} b^{\prec_j}(u) = f(X), \end{aligned}$$

womit bewiesen ist, dass X eine optimale Lösung ist. \square

Lemma 14.21. (Schrijver [2000]) *Jede Iteration kann in $O(n^3 + \gamma n^2)$ -Zeit erledigt werden, wobei γ die für einen Orakelauftruf benötigte Zeit ist.*

Beweis: Es genügt zu zeigen, dass (5) in $O(n^3 + \gamma n^2)$ -Zeit erledigt werden kann. Sei $x = \lambda_1 b^{\prec_1} + \dots + \lambda_k b^{\prec_k}$ und $s \prec_i t$. Zunächst beweisen wir die

Behauptung: Für ein geeignetes $\delta \geq 0$ kann $\delta(\chi^t - \chi^s)$ als Konvexitätskombination der Vektoren $b^{\prec_i^{s,v}} - b^{\prec_i}$ für $s \prec_i v \preceq_i t$ in $O(\gamma n^2)$ -Zeit geschrieben werden.

Um diese Behauptung zu beweisen, benötigen wir einige Vorbereitungen. Sei $s \prec_i v \preceq_i t$. Nach Definition gilt $b^{\prec_i^{s,v}}(u) = b^{\prec_i}(u)$ für $u \prec_i s$ oder $u \succ_i v$. Für $s \preceq_i u \prec_i v$ haben wir, da f submodular ist:

$$\begin{aligned} b^{\prec_i^{s,v}}(u) &= f(\{w \in U : w \preceq_i^{s,v} u\}) - f(\{w \in U : w \prec_i^{s,v} u\}) \\ &\leq f(\{w \in U : w \preceq_i u\}) - f(\{w \in U : w \prec_i u\}) = b^{\prec_i}(u). \end{aligned}$$

Ferner haben wir für $u = v$:

$$\begin{aligned} b^{\prec_i^{s,v}}(v) &= f(\{w \in U : w \preceq_i^{s,v} v\}) - f(\{w \in U : w \prec_i^{s,v} v\}) \\ &= f(\{w \in U : w \prec_i s\} \cup \{v\}) - f(\{w \in U : w \prec_i s\}) \\ &\geq f(\{w \in U : w \preceq_i v\}) - f(\{w \in U : w \prec_i v\}) \\ &= b^{\prec_i}(v). \end{aligned}$$

Beachte auch, dass $\sum_{u \in U} b^{\prec_i^{s,v}}(u) = f(U) = \sum_{u \in U} b^{\prec_i}(u)$.

Da die Behauptung trivial ist, falls $b^{\prec_i^{s,v}} = b^{\prec_i}$ für irgendein $s \prec_i v \preceq_i t$, können wir annehmen, dass $b^{\prec_i^{s,v}}(v) > b^{\prec_i}(v)$ für alle $s \prec_i v \preceq_i t$. Nun definieren wir rekursiv

$$\kappa_v := \frac{\chi^t - \sum_{v \prec_i w \preceq_i t} \kappa_w (b^{\prec_i^{s,w}}(v) - b^{\prec_i}(v))}{b^{\prec_i^{s,v}}(v) - b^{\prec_i}(v)} \geq 0$$

für $s \prec_i v \preceq_i t$. Dann erhalten wir $\sum_{s \prec_i v \preceq_i t} \kappa_v (b^{\prec_i^{s,v}} - b^{\prec_i}) = \chi^t - \chi^s$, weil $\sum_{s \prec_i v \preceq_i t} \kappa_v (b^{\prec_i^{s,v}}(u) - b^{\prec_i}(u)) = \sum_{u \preceq_i v \preceq_i t} \kappa_v (b^{\prec_i^{s,v}}(u) - b^{\prec_i}(u)) = \chi_u^t$ für alle $s \prec_i u \preceq_i t$ und die Summe über alle Komponenten gleich Null ist.

Setzen wir nun $\delta := \frac{1}{\sum_{s \prec_i v \preceq_i t} \kappa_b}$ und multiplizieren jedes κ_u mit δ , so folgt die Behauptung.

Betrachte nun $\epsilon := \min\{\lambda_i \delta, -x(t)\}$ und $x' := x + \epsilon(\chi^t - \chi^s)$. Ist $\epsilon = \lambda_i \delta \leq -x(t)$, so folgt $x' = \sum_{j=1}^k \lambda_j b^{\prec_j} + \lambda_i \sum_{s \prec_i v \preceq_i t} \kappa_b (b^{\prec_i^{s,v}} - b^{\prec_i})$, d.h. wir haben x' als Konvexitätskombination der b^{\prec_j} ($j \in \{1, \dots, k\} \setminus \{i\}$) und $b^{\prec_i^{s,v}}$ ($s \prec_i v \preceq_i t$) geschrieben. Ist $\epsilon = -x(t)$, so können wir b^{\prec_i} zusätzlich in die Konvexitätskombination einbeziehen.

Schließlich reduzieren wir diese Konvexitätskombination auf höchstens n Vektoren in $O(n^3)$ -Zeit, wie in Aufgabe 5, Kapitel 4, gezeigt wird. \square

Lemma 14.22. (Vygen [2003]) SCHRIJVERS ALGORITHMUS terminiert nach $O(n^5)$ Iterationen.

Beweis: Wird eine Kante (v, w) eingeführt, nachdem ein neuer Vektor $b^{\prec_i^{s,v}}$ in ⑤ einer Iteration hinzugefügt wurde, so gilt $s \preceq_i w \prec_i v \preceq_i t$ in dieser Iteration. Somit ist $d(w) \leq d(s) + 1 = d(t) \leq d(v) + 1$ in dieser Iteration, und die Einführung der neuen Kante kann die Distanz von P nach jedem $u \in U$ nicht verringern. Da ⑤ sicherstellt, dass kein Element jemals zu P hinzugefügt wird, wird die Distanz $d(u)$ für kein $u \in U$ jemals verringert.

Ein *Block* sei eine Folge von Iterationen, in der das Paar (s, t) konstant bleibt. Beachte, dass jeder Block aus $O(n^2)$ Iterationen besteht, da (α, β) bei jeder Iteration innerhalb eines Blocks lexicographisch abnimmt. Es bleibt zu zeigen, dass es $O(n^3)$ Blöcke gibt.

Ein Block kann nur dann enden, wenn mindestens einer der drei folgenden Schritte stattfindet (nach Wahl von t und s , weil eine Iteration mit $t = t^*$ keine in t^* endende Kante hinzufügt, und weil ein Knoten v nur dann zu N hinzukommen kann, wenn $v = s$ und somit $d(v) < d(t)$ gilt):

- (a) Die Distanz $d(v)$ wächst für ein $v \in U$.
- (b) Es wird t aus N entfernt.
- (c) Es wird (s, t) aus A entfernt.

Nun zählen wir die Anzahl der Blöcke dieser drei Typen. Offensichtlich gibt es $O(n^2)$ Blöcke des Typs (a).

Zur Anzahl der Blöcke des Typs (b) behaupten wir: Für jedes $t^* \in U$ gibt es $O(n^2)$ Iterationen mit $t = t^*$ und $x'(t) = 0$. Der Beweis ist einfach: Zwischen je zwei solchen Iterationen verändert sich $d(v)$ für ein $v \in U$, und dies findet $O(n^2)$ mal statt, da d -Werte nur wachsen können. Somit gibt es $O(n^3)$ Blöcke des Typs (b).

Schließlich zeigen wir, dass es $O(n^3)$ Blöcke des Typs (c) gibt. Es genügt zu zeigen: $d(t)$ verändert sich vor dem nächsten solchen Block mit dem Paar (s, t) .

Für gegebene $s, t \in U$ bezeichnen wir s als *t-langweilig*, falls $(s, t) \notin A$ oder $d(t) \leq d(s)$. Seien $s^*, t^* \in U$ und betrachte die Zeitspanne nachdem ein Block mit $s = s^*$ und $t = t^*$ wegen des Entfernen von (s^*, t^*) aus A zu Ende geht, bis zur nächsten Veränderung von $d(t^*)$. Wir werden beweisen, dass jedes $v \in \{s^*, \dots, n\}$

während dieser Zeitspanne t^* -langweilig ist. Dies wenden wir dann auf $v = s^*$ an, womit der Beweis beendet ist.

Am Anfang obiger Zeitspanne ist jedes $v \in \{s^*+1, \dots, n\}$ t^* -langweilig, wegen der Wahl von $s = s^*$ in der Iteration unmittelbar vor Beginn der Zeitspanne. Es ist s^* auch t^* -langweilig, da (s^*, t^*) aus A entfernt wird. Da $d(t^*)$ während der Zeitspanne konstant bleibt und $d(v)$ für kein v abnimmt, brauchen wir nur die Einführung neuer Kanten zu überprüfen.

Angenommen, es wird nach einer Iteration, die das Paar (s, t) wählt, für ein $v \in \{s^*, \dots, n\}$ die Kante (v, t^*) zu A hinzugefügt. Dann gilt, mit den anfänglichen Bemerkungen in diesem Beweis, $s \preceq_i t^* \prec_i v \preceq_i t$ in dieser Iteration, woraus $d(t^*) \leq d(s) + 1 = d(t) \leq d(v) + 1$ folgt. Nun machen wir eine Fallunterscheidung: Ist $s > v$, so folgt $d(t^*) \leq d(s)$: Entweder weil $t^* = s$, oder weil s t^* -langweilig war und $(s, t^*) \in A$. Ist $s < v$, so folgt $d(t) \leq d(v)$: Entweder weil $t = v$, oder wegen der Wahl von s und da $(v, t) \in A$. Also haben wir in beiden Fällen $d(t^*) \leq d(v)$ und somit bleibt v t^* -langweilig. \square

Aus Satz 14.20, Lemma 14.21 und Lemma 14.22 folgt:

Satz 14.23. *Das MINIMIERUNGSPROBLEM SUBMODULARER FUNKTIONEN kann in $O(n^8 + \gamma n^7)$ -Zeit gelöst werden, wobei γ die für einen Orakelauftrag benötigte Zeit ist.* \square

Iwata [2002] hat einen voll-kombinatorischen Algorithmus beschrieben (unter alleiniger Verwendung von Additionen, Subtraktionen, Vergleichen und Orakelaufrufen, aber keinen Multiplikationen oder Divisionen). Ferner hat er die Laufzeit verbessert (Iwata [2003]). Ein schnellerer streng polynomieller Algorithmus stammt von Orlin [2007]; er läuft in $O(n^6 + \gamma n^5)$ -Zeit.

Vor Kurzem haben Lee, Sidford und Wong [2015] einen streng polynomiellen Algorithmus beschrieben, der eine submodulare Funktion in $O(\gamma n^3 \log^2 n + n^4 \log^{O(1)} n)$ Laufzeit minimiert.

14.5 Symmetrische submodulare Funktionen

Eine submodulare Funktion $f : 2^U \rightarrow \mathbb{R}$ heißt **symmetrisch**, falls $f(A) = f(U \setminus A)$ für alle $A \subseteq U$. Für diesen Spezialfall ist das MINIMIERUNGSPROBLEM SUBMODULARER FUNKTIONEN trivial, da $2f(\emptyset) = f(\emptyset) + f(U) \leq f(A) + f(U \setminus A) = 2f(A)$ für alle $A \subseteq U$, woraus folgt, dass die leere Menge optimal ist. Also ist dieses Problem nur dann von Interesse, wenn man diesen Trivialfall ausschließt: Man möchte eine nichtleere echte Teilmenge A von U mit minimalem $f(A)$ bestimmen.

Durch Verallgemeinerung des in Abschnitt 8.7 gegebenen Algorithmus hat Queyranne [1998] einen relativ einfachen kombinatorischen Algorithmus für dieses Problem gefunden, der nur $O(n^3)$ Orakelaufrufe benutzt. Das folgende Lemma ist eine Verallgemeinerung von Lemma 8.41 (Aufgabe 16):

Lemma 14.24. Für eine gegebene symmetrische submodulare Funktion $f : 2^U \rightarrow \mathbb{R}$ mit $n := |U| \geq 2$ können wir zwei Elemente $x, y \in U$ mit $x \neq y$ und $f(\{x\}) = \min\{f(X) : x \in X \subseteq U \setminus \{y\}\}$ in $O(n^2\theta)$ -Zeit bestimmen, wobei θ die Zeitschranke des Orakels für f ist.

Beweis: Wir bilden eine Reihenfolge $U = \{u_1, \dots, u_n\}$ mittels folgender Schritte für $k = 1, \dots, n - 1$. Angenommen, die Folge u_1, \dots, u_{k-1} sei bereits gebildet, und es sei $U_{k-1} := \{u_1, \dots, u_{k-1}\}$. Für $C \subseteq U$ setzen wir

$$w_k(C) := f(C) - \frac{1}{2}(f(C \setminus U_{k-1}) + f(C \cup U_{k-1}) - f(U_{k-1})).$$

Beachte, dass w_k auch symmetrisch ist. Sei u_k ein $w_k(\{u_k\})$ maximierendes Element aus $U \setminus U_{k-1}$.

Sei auch noch u_n das einzige Element in $U \setminus \{u_1, \dots, u_{n-1}\}$. Offensichtlich kann die Reihenfolge u_1, \dots, u_n in $O(n^2\theta)$ -Zeit bewerkstelligt werden.

Behauptung: Für alle $k = 1, \dots, n - 1$ und alle $x, y \in U \setminus U_{k-1}$ mit $x \neq y$ und $w_k(\{x\}) \leq w_k(\{y\})$ folgt

$$w_k(\{x\}) = \min\{w_k(C) : x \in C \subseteq U \setminus \{y\}\}.$$

Wir beweisen diese Behauptung mittels Induktion über k . Für $k = 1$ ist die Aussage trivial, da $w_1(C) = \frac{1}{2}f(\emptyset)$ für alle $C \subseteq U$.

Sei nun $k > 1$ und $x, y \in U \setminus U_{k-1}$ mit $x \neq y$ und $w_k(\{x\}) \leq w_k(\{y\})$. Sei ferner $Z \subseteq U$ mit $u_{k-1} \notin Z$ und $z \in Z \setminus U_{k-1}$. Nach Wahl von u_{k-1} folgt $w_{k-1}(\{z\}) \leq w_{k-1}(\{u_{k-1}\})$; mit der Induktionsvoraussetzung erhalten wir dann $w_{k-1}(\{z\}) \leq w_{k-1}(Z)$. Ferner folgt aus der Submodularität von f :

$$\begin{aligned} & (w_k(Z) - w_{k-1}(Z)) - (w_k(\{z\}) - w_{k-1}(\{z\})) \\ &= \frac{1}{2}(f(Z \cup U_{k-2}) - f(Z \cup U_{k-1}) - f(U_{k-2}) + f(U_{k-1})) \\ &\quad - \frac{1}{2}(f(\{z\} \cup U_{k-2}) - f(\{z\} \cup U_{k-1}) - f(U_{k-2}) + f(U_{k-1})) \\ &= \frac{1}{2}(f(Z \cup U_{k-2}) + f(\{z\} \cup U_{k-1}) - f(Z \cup U_{k-1}) - f(\{z\} \cup U_{k-2})) \\ &\geq 0. \end{aligned}$$

Somit haben wir $w_k(Z) - w_k(\{z\}) \geq w_{k-1}(Z) - w_{k-1}(\{z\}) \geq 0$.

Für den letzten Schritt des Beweises der Behauptung, sei $C \subseteq U$ mit $x \in C$ und $y \notin C$. Wir haben folgende Fallunterscheidung:

Fall 1: $u_{k-1} \notin C$. Obiges Resultat angewendet auf $Z = C$ und $z = x$ liefert dann $w_k(C) \geq w_k(\{x\})$, wie erwünscht.

Fall 2: $u_{k-1} \in C$. Obiges Resultat angewendet auf $Z = U \setminus C$ und $z = y$ liefert dann $w_k(C) = w_k(U \setminus C) \geq w_k(\{y\}) \geq w_k(\{x\})$.

Damit ist die Behauptung bewiesen. Wenden wir sie auf $k = n - 1$, $x = u_n$ und $y = u_{n-1}$ an, so erhalten wir $w_{n-1}(\{u_n\}) = \min\{w_{n-1}(C) : u_n \in C \subseteq U \setminus \{u_{n-1}\}\}$.

Da $w_{n-1}(C) = f(C) - \frac{1}{2}(f(\{u_n\}) + f(U \setminus \{u_{n-1}\}) - f(U_{n-2}))$ für alle $C \subseteq U$ mit $u_n \in C$ und $u_{n-1} \notin C$, folgt das Lemma (setze $x := u_n$ und $y := u_{n-1}$). \square

Der obige Beweis stammt von Fujishige [1998]. Auf analoge Weise wenden wir uns nun dem Beweis von Satz 8.42 zu:

Satz 14.25. (Queyranne [1998]) Für eine gegebene symmetrische submodulare Funktion $f : 2^U \rightarrow \mathbb{R}$ kann man eine nichtleere echte Teilmenge A von U mit minimalem $f(A)$ in $O(n^3\theta)$ -Zeit bestimmen, wobei θ die Zeitschranke des Orakels für f ist.

Beweis: Das Problem ist trivial, falls $|U| = 1$. Andernfalls wenden wir Lemma 14.24 an und erhalten zwei Elemente $x, y \in U$ mit $f(\{x\}) = \min\{f(X) : x \in X \subseteq U \setminus \{y\}\}$ in $O(n^2\theta)$ -Zeit. Als Nächstes bestimmen wir rekursiv eine nichtleere echte Teilmenge von $U \setminus \{x\}$, welche die wie folgt definierte Funktion $f' : 2^{U \setminus \{x\}} \rightarrow \mathbb{R}$ minimiert: $f'(X) := f(X)$ für $y \notin X$ und $f'(X) := f(X \cup \{x\})$ für $y \in X$. Man erkennt leicht, dass f' symmetrisch und submodular ist.

Sei $\emptyset \neq Y \subset U \setminus \{x\}$ eine f' minimierende Menge; o. B. d. A. können wir annehmen, dass $y \in Y$ (da f' symmetrisch ist). Wir behaupten nun: Entweder $\{x\}$ oder $Y \cup \{x\}$ minimiert f (über allen nichtleeren echten Teilmengen von U). Um dies zu sehen, sei $C \subset U$ mit $x \in C$. Ist $y \notin C$, so folgt $f(\{x\}) \leq f(C)$ nach Wahl von x und y . Ist $y \in C$, so folgt $f(C) = f'(C \setminus \{x\}) \geq f'(Y) = f(Y \cup \{x\})$. Somit ist $f(C) \geq \min\{f(\{x\}), f(Y \cup \{x\})\}$ für alle nichtleeren echten Teilmengen C von U .

Um die angegebene Laufzeit zu erreichen, können wir f' natürlich nicht explizit berechnen. Stattdessen speichern wir eine Partition von U , die am Anfang aus lauter einteiligen Knotenmengen besteht. Bei jedem Schritt der Rekursion bilden wir die Vereinigung derjenigen beiden Mengen der Partition, die x und y enthalten. Auf diese Weise kann f' effizient berechnet werden (unter Verwendung des Orakels für f). \square

Dieses Resultat ist von Nagamochi und Ibaraki [1998] und von Rizzi [2000] weiter verallgemeinert worden.

14.6 Die Maximierung submodularer Funktionen

Das Problem der Maximierung einer submodularen Funktion (wiederum gegeben durch ein Orakel) kann nicht mit einer polynomiellen Anzahl von Orakelaufrufern gelöst werden, wie Feige, Mirrokni und Vondrák [2011] gezeigt haben; siehe Aufgabe 18. Es gibt jedoch effiziente Approximationsalgorithmen für das folgende Problem: Für eine gegebene nichtnegative submodulare Funktion $f : 2^U \rightarrow \mathbb{R}_+$ finde man eine Menge $A \subseteq U$ mit $f(A) \geq \frac{1}{k} \max\{f(W) : W \subseteq U\}$, d. h. eine k -Approximation. Bereits der Erwartungswert einer zufällig gleichmäßig gewählten Menge ist eine 4-Approximation (siehe Aufgabe 17). Es geht aber besser. Man betrachte den folgenden einfachen randomisierten Algorithmus. Er beginnt mit

$A = \emptyset$ und $B = U$ und entscheidet in Iteration i , ob er i dazunimmt (und es zu A hinzufügt) oder nicht (und es aus B entfernt). Am Ende stimmen A und B überein.

EINFACHER MAXIMIERUNGSALGORITHMUS FÜR SUBMODULARE FUNKTIONEN

Input: Eine endliche Menge $U = \{1, \dots, n\}$. Eine submodulare Funktion $f : 2^U \rightarrow \mathbb{R}_+$ (gegeben durch ein Orakel).

Output: Eine Teilmenge $A \subseteq U$.

① Setze $A := \emptyset$ und $B := U$.

② For $i := 1$ to n do:

$$\Delta_A := f(A \cup \{i\}) - f(A).$$

$$\Delta_B := f(B \setminus \{i\}) - f(B).$$

Setze $A := A \cup \{i\}$ mit Wahrscheinlichkeit x und $B := B \setminus \{i\}$ sonst, wobei

$$x = \begin{cases} 1 & \text{falls } \Delta_B \leq 0 \\ 0 & \text{falls } \Delta_A < 0 \\ \frac{\Delta_A}{\Delta_A + \Delta_B} & \text{falls } \Delta_A \geq 0 \text{ und } \Delta_B > 0 \end{cases} \quad (14.3)$$

③ Output A .

Satz 14.26. (Buchbinder et al. [2015]) *Der Erwartungswert von $f(A)$, wobei A der Output des EINFACHEN MAXIMIERUNGSALGORITHMUS FÜR SUBMODULARE FUNKTIONEN ist, beträgt mindestens $\frac{1}{2} \max\{f(W) : W \subseteq U\}$.*

Beweis: Man beachte zunächst, dass $\Delta_A + \Delta_B = f(A \cup \{i\}) - f(A) + f(B \setminus \{i\}) - f(B) \geq 0$ in jeder Iteration gilt, da $A = B \cap \{1, \dots, i-1\}$, $B = A \cup \{i, \dots, n\}$ und f submodular ist.

Sei $A_0 := \emptyset$, $B_0 := U$ und seien A_i, B_i die Mengen A, B nach i Iterationen. Sei $O_0 \subseteq U$ eine Menge mit $f(O_0) = \max\{f(W) : W \subseteq U\}$. Ferner definiere man $O_i \subseteq U$ mittels $O_i \cap \{i+1, \dots, n\} = O_0 \cap \{i+1, \dots, n\}$ und $O_i \cap \{1, \dots, i\} = A_i = B_i \cap \{1, \dots, i\}$. Mit anderen Worten, O_i stimmt mit den Entscheidungen in den ersten i Iterationen überein und auch mit der optimalen Lösung O_0 auf den Elementen $i+1, \dots, n$. Man betrachte nun die Änderung in Iteration i . Wir behaupten:

$$\text{Exp}[f(O_{i-1}) - f(O_i)] \leq \max\{x \Delta_B, (1-x) \Delta_A\}, \quad (14.4)$$

wobei x die Wahrscheinlichkeit der Hinzufügung von i zu A ist, unabhängig davon, wie x im Algorithmus gewählt wurde.

Um (14.4) zu beweisen, betrachten wir zwei Fälle. Ist $i \notin O_0$, dann ist $\text{Exp}[f(O_{i-1}) - f(O_i)] = x(f(O_{i-1}) - f(O_{i-1} \cup \{i\})) + (1-x)(f(O_{i-1}) - f(O_{i-1})) = x(f(O_{i-1}) - f(O_{i-1} \cup \{i\})) \leq x(f(B_{i-1} \setminus \{i\}) - f(B_{i-1})) = x \Delta_B$, wobei die Ungleichung aus der Submodularität und aus $O_{i-1} \subseteq B_{i-1} \setminus \{i\}$ folgt.

Ist $i \in O_0$, dann ist $\text{Exp}[f(O_{i-1}) - f(O_i)] = x(f(O_{i-1}) - f(O_{i-1})) + (1-x)(f(O_{i-1}) - f(O_{i-1} \setminus \{i\})) = (1-x)(f(O_{i-1}) - f(O_{i-1} \setminus \{i\})) \leq$

$(1 - x)(f(A_{i-1} \cup \{i\}) - f(A_{i-1})) = (1 - x)\Delta_A$, wobei die Ungleichung aus der Submodularität und aus $O_{i-1} \setminus \{i\} \supseteq A_{i-1}$ folgt.

Als Nächstes zeigen wir: Unsere Wahl von x bedeutet, dass

$$\max\{x\Delta_B, (1-x)\Delta_A\} \leq \frac{1}{2}(\text{Exp}[f(A_i) - f(A_{i-1})] + \text{Exp}[f(B_i) - f(B_{i-1})]). \quad (14.5)$$

In der Tat, gilt $\Delta_A < 0$ oder $\Delta_B \leq 0$, so verschwindet die linke Seite der Ungleichung und $f(A_i) - f(A_{i-1}) + f(B_i) - f(B_{i-1}) = \max\{\Delta_A, \Delta_B\} \geq 0$. Haben wir $\Delta_A \geq 0$ und $\Delta_B > 0$, so folgt die Ungleichung aus den folgenden einfachen Schritten:

$$\begin{aligned} \max\{x\Delta_B, (1-x)\Delta_A\} &= \frac{\Delta_A \Delta_B}{\Delta_A + \Delta_B} \\ &\leq \frac{1}{2} \frac{\Delta_A^2 + \Delta_B^2}{\Delta_A + \Delta_B} \\ &= \frac{1}{2} \left(\frac{\Delta_A}{\Delta_A + \Delta_B} \Delta_A + \frac{\Delta_B}{\Delta_A + \Delta_B} \Delta_B \right) \\ &= \frac{1}{2} (\text{Exp}[f(A_i) - f(A_{i-1})] + \text{Exp}[f(B_i) - f(B_{i-1})]). \end{aligned}$$

Aus den beiden Ungleichungen (14.4) und (14.5) folgt nun

$$\text{Exp}[f(O_{i-1}) - f(O_i)] \leq \frac{1}{2} \text{Exp}[f(A_i) - f(A_{i-1})] + \frac{1}{2} \text{Exp}[f(B_i) - f(B_{i-1})].$$

Addieren wir diese Ungleichung für $i = 1, \dots, n$, so bekommen wir $\text{Exp}[f(O_0)] - \text{Exp}[f(O_n)] \leq \frac{1}{2}(\text{Exp}[f(A_n)] - \text{Exp}[f(A_0)] + \text{Exp}[f(B_n) - \text{Exp}[f(B_0)]] = \text{Exp}[f(A_n)] - \frac{1}{2}f(\emptyset) - \frac{1}{2}f(U) \leq \text{Exp}[f(A_n)]$. Zusammen mit $O_n = A_n$ haben wir dann $2\text{Exp}[f(A_n)] \geq f(O_0)$. \square

Dieser Algorithmus kann folgendermaßen derandomisiert werden.

Satz 14.27. (Buchbinder und Feldman [2016]) *Es gibt einen deterministischen polynomiellen Algorithmus, der eine 2-Approximation für das Problem der Maximierung einer nichtnegativen submodularen Funktion berechnet.*

Beweis: Anstatt zufällige Wahlen zu treffen, behalten wir eine explizite Wahrscheinlichkeitsverteilung der Paare (A, B) . Das einzige Problem besteht darin, zu vermeiden, dass der Träger dieser Verteilung exponentiell wächst. Dazu gehen wir folgendermaßen vor. Sei T_{i-1} eine Menge von Tripeln (p, A, B) mit $p > 0$ und $A \subseteq B \subseteq U$ mit $A = A \cap \{1, \dots, i-1\} = B \cap \{1, \dots, i-1\}$ und $\{i, \dots, n\} \subseteq B$ und $\sum_{(p, A, B) \in T_{i-1}} p = 1$ am Anfang von Iteration i (zu Beginn ist $T_0 = \{(1, \emptyset, U)\}$). Dann betrachten wir das Polytop bestehend aus allen $x \in [0, 1]^{T_{i-1}}$, für die

$$\begin{aligned} \sum_{t=(p, A, B) \in T_{i-1}} p(x_t \Delta_A + (1-x_t) \Delta_B) &\geq 2 \sum_{t=(p, A, B) \in T_{i-1}} p x_t \Delta_B, \\ \sum_{t=(p, A, B) \in T_{i-1}} p(x_t \Delta_A + (1-x_t) \Delta_B) &\geq 2 \sum_{t=(p, A, B) \in T_{i-1}} p(1-x_t) \Delta_A, \end{aligned}$$

gilt, wobei $\Delta_A := f(A \cup \{i\}) - f(A)$ und $\Delta_B := f(B \setminus \{i\}) - f(B)$, wie bereits oben definiert. Dieses Polytop ist nicht leer, da wir für $t = (p, A, B)$ ein x_t wie in (14.3) wählen können; siehe (14.5) im vorherigen Beweis.

Sei x eine Ecke dieses Polytops. Nach Satz 4.18 können wir eine solche Ecke in polynomieller Zeit bestimmen. Sie erfüllt $|T_{i-1}|$ linear unabhängige Ungleichungen mit Gleichheit (siehe Proposition 3.9), somit sind alle Variablen bis auf höchstens zwei gleich 0 oder 1. Jetzt können wir mit der nächsten Iteration und der folgenden Menge fortfahren:

$$T_i := \{(px_t, A \cup \{i\}, B) : t = (p, A, B) \in T_{i-1}, x_t > 0\} \cup \\ \{(p(1-x_t), A, B \setminus \{i\}) : t = (p, A, B) \in T_{i-1}, x_t < 1\}.$$

Also wächst die Anzahl der Tripel um höchstens zwei in jeder Iteration.

Nehmen wir nun (A_i, B_i) als (A, B) mit Wahrscheinlichkeit p für jedes $(p, A, B) \in T_i$, so folgt

$$\begin{aligned} & \frac{1}{2}(\text{Exp}[f(A_i)] - \text{Exp}[f(A_{i-1})] + \text{Exp}[f(B_i) - \text{Exp}[f(B_{i-1})]]) \\ &= \frac{1}{2} \sum_{t=(p,A,B) \in T_{i-1}} p(x_t(f(A \cup \{i\}) - f(A)) + (1-x_t)(f(B \setminus \{i\}) - f(B))) \\ &= \frac{1}{2} \sum_{t=(p,A,B) \in T_{i-1}} p(x_t \Delta_A + (1-x_t) \Delta_B) \\ &\geq \sum_{t=(p,A,B) \in T_{i-1}} p \max\{x_t \Delta_B, (1-x_t) \Delta_A\}, \end{aligned}$$

wobei die letzte Ungleichung gilt, da x in dem obigen Polytop ist. Zusammen mit (14.4) haben wir also

$$\text{Exp}[f(O_{i-1}) - f(O_i)] \leq \frac{1}{2} \text{Exp}[f(A_i) - f(A_{i-1})] + \frac{1}{2} \text{Exp}[f(B_i) - f(B_{i-1})].$$

Somit gilt $2 \text{Exp}[f(O_n)] \geq O_0$, wie im vorherigen Beweis. \square

Es ist in der Tat nicht notwendig, die Ellipsoidmethode zur Bestimmung einer Ecke zu benutzen; stattdessen ist dies in linearer Zeit möglich; siehe Aufgabe 20.

Aufgaben

- Sei G ein ungerichteter Graph und M ein kardinalitätsmaximales Matching in G . Sei \mathcal{F} die Familie derjenigen Teilmengen $X \subseteq E(G)$, für die es einen speziellen Blütenwald F bezüglich M mit $E(F) \setminus M = X$ gibt. Man beweise, dass $(E(G) \setminus M, \mathcal{F})$ ein Greedoid ist.
Hinweis: Man benutze Aufgabe 25, Kapitel 10.
- Sei (E, \mathcal{F}) ein Greedoid und $c' : E \rightarrow \mathbb{R}_+$. Man betrachte die Bottleneck-Funktion $c(F) := \min\{c'(e) : e \in F\}$ für $F \subseteq E$. Man zeige, dass die Anwendung des GREEDY-ALGORITHMUS FÜR GREEDOIDE auf (E, \mathcal{F}) und c ein inklusionsmaximales Element F in \mathcal{F} mit maximalem $c(F)$ bestimmt.

3. Diese Aufgabe erläutert, wie man Greedoide auch als Sprache (siehe Definition 15.1) definieren kann. Sei E eine endliche Menge. Eine Sprache L über dem Alphabet E heißt Greedoidsprache, falls die folgenden vier Bedingungen erfüllt sind:

- (a) L enthält den leeren String;
- (b) $x_i \neq x_j$ für alle $(x_1, \dots, x_n) \in L$ und $1 \leq i < j \leq n$;
- (c) $(x_1, \dots, x_{n-1}) \in L$ für alle $(x_1, \dots, x_n) \in L$;
- (d) Sind $(x_1, \dots, x_n), (y_1, \dots, y_m) \in L$ mit $m < n$, so gibt es ein $i \in \{1, \dots, n\}$ mit $(y_1, \dots, y_m, x_i) \in L$.

Es heißt L Antimatroidsprache, falls die Bedingungen (a), (b), (c) und die folgende erfüllt sind:

- (d') Sind $(x_1, \dots, x_n), (y_1, \dots, y_m) \in L$ mit $\{x_1, \dots, x_n\} \not\subseteq \{y_1, \dots, y_m\}$, so gibt es ein $i \in \{1, \dots, n\}$ mit $(y_1, \dots, y_m, x_i) \in L$.

Man beweise: Eine Sprache L über dem Alphabet E ist genau dann eine Greedoidsprache (Antimatroidsprache), wenn das Mengensystem (E, \mathcal{F}) ein Greedoid (Antimatroid) ist, wobei $\mathcal{F} := \{\{x_1, \dots, x_n\} : (x_1, \dots, x_n) \in L\}$.

4. Sei U eine endliche Menge und $f : 2^U \rightarrow \mathbb{R}$. Man beweise, dass f genau dann submodular ist, wenn $f(X \cup \{y, z\}) - f(X \cup \{y\}) \leq f(X \cup \{z\}) - f(X)$ für alle $X \subseteq U$ und $y, z \in U$ mit $y \neq z$.

5. Sei (G, u, s, t) ein Netzwerk und $U := \delta^+(s)$. Sei $P := \{x \in \mathbb{R}_+^U : \text{es gibt einen } s\text{-}t\text{-Fluss } f \text{ in } (G, u) \text{ mit } f(e) = x_e \text{ für alle } e \in U\}$. Man beweise, dass P ein Polymatroid ist.

6. Sei P ein Polymatroid. Man zeige, dass es eine monotone submodulare Funktion f mit $f(\emptyset) = 0$ und $P = P(f)$ gibt.

* 7. Man beweise: Eine nichtleere kompakte Menge $P \subseteq \mathbb{R}_+^n$ ist genau dann ein Polymatroid, wenn P die folgenden beiden Eigenschaften hat:

- (a) Für alle $0 \leq x \leq y \in P$ gilt $x \in P$.
- (b) Für alle $x \in \mathbb{R}_+^n$ und alle $y, z \leq x$ mit $y, z \in P$, die maximal mit dieser Eigenschaft sind (d. h. aus $y \leq w \leq x$ und $w \in P$ folgt $w = y$, und aus $z \leq w \leq x$ und $w \in P$ folgt $w = z$), gilt $\mathbf{1}y = \mathbf{1}z$.

Bemerkung: Dies ist die ursprüngliche, von Edmonds [1970] stammende Definition.

8. Man beweise, dass der auf den Vektor $c \in \mathbb{R}^E$ mit $c(e) > 0$ für alle $e \in E$ und eine submodulare aber nicht notwendigerweise monotone Funktion $f : 2^E \rightarrow \mathbb{R}$ mit $f(\emptyset) \geq 0$ angewendete POLYMATROID-GREEDY-ALGORITHMUS das folgende LP löst:

$$\max \left\{ cx : \sum_{e \in A} x_e \leq f(A) \text{ für alle } A \subseteq E \right\}.$$

9. Man beweise Satz 14.12 im Spezialfall, dass f und g die Rangfunktionen zweier Matroide sind, indem man eine ganzzahlige optimale duale Lösung aus den mit dem GEWICHTETEN MATROID-INTERSEKTIONS-ALGORITHMUS erzeugten c_1 und c_2 bildet.

(Frank [1981])

- * 10. Sei S eine endliche Menge und $f : 2^S \rightarrow \mathbb{R}$. Man definiere $f' : \mathbb{R}_+^S \rightarrow \mathbb{R}$ wie folgt. Für jedes $x \in \mathbb{R}_+^S$ gibt es eindeutig bestimmte $k \in \mathbb{Z}_+$, $\lambda_1, \dots, \lambda_k > 0$ und $\emptyset \subset T_1 \subset T_2 \subset \dots \subset T_k \subseteq S$ mit $x = \sum_{i=1}^k \lambda_i \chi^{T_i}$, wobei χ^{T_i} der Inzidenzvektor von T_i ist. Dann setze man $f'(x) := \sum_{i=1}^k \lambda_i f(T_i)$. Man beweise, dass f genau dann submodular ist, wenn f' konvex ist. (Lovász [1983])
11. Sei E eine endliche Menge und $f : 2^E \rightarrow \mathbb{R}_+$ eine submodulare Funktion mit $f(\{e\}) \leq 2$ für alle $e \in E$. (Das Paar (E, f) wird gelegentlich ein 2-Polymatroid genannt.) Das POLYMATROID-MATCHING-PROBLEM fragt nach einer kardinalitätsmaximalen Menge $X \subseteq E$ mit $f(X) = 2|X|$. (Die Funktion f ist natürlich durch ein Orakel gegeben.)
Seien E_1, \dots, E_k paarweise disjunkte nicht-geordnete Paare und (E, \mathcal{F}) ein Matroid (gegeben durch ein Unabhängigkeits-Orakel), wobei $E = E_1 \cup \dots \cup E_k$. Das MATROID-PARITÄTS-PROBLEM fragt nach einer kardinalitätsmaximalen Menge $I \subseteq \{1, \dots, k\}$ mit $\bigcup_{i \in I} E_i \in \mathcal{F}$.
- (a) Man zeige, dass das MATROID-PARITÄTS-PROBLEM polynomiell auf das POLYMATROID-MATCHING-PROBLEM reduziert werden kann.
 - * (b) Man zeige, dass das POLYMATROID-MATCHING-PROBLEM polynomiell auf das MATROID-PARITÄTS-PROBLEM reduziert werden kann.
Hinweis: Man verwende einen Algorithmus für das MINIMIERUNGSPROBLEM SUBMODULARER FUNKTIONEN.
 - * (c) Man zeige, dass es keinen Algorithmus für das POLYMATROID-MATCHING-PROBLEM mit in $|E|$ polynomieller Laufzeit gibt.
(Jensen und Korte [1982], Lovász [1981])
- (Ein Problem lässt sich polynomiell auf ein anderes reduzieren, wenn das erste Problem mit einem polynomiellen Orakelalgorithmus, der ein Orakel für das zweite benutzt, gelöst werden kann; siehe Kapitel 15.)
- Bemerkung:* Ein polynomieller Algorithmus für einen wichtigen Spezialfall (repräsentierbare Matroide) wurde von Lovász [1980,1981] beschrieben und von Iwata und Kobayashi [2017] und Pap [2017+] auf die gewichtete Verallgemeinerung erweitert.
12. Eine Funktion $f : 2^S \rightarrow \mathbb{R} \cup \{\infty\}$ heißt kreuzend-submodular, falls $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$ für je zwei Mengen $X, Y \subseteq S$ mit $X \cap Y \neq \emptyset$ und $X \cup Y = S$.
Das SUBMODULARER-FLUSS-PROBLEM lautet: Gegeben sei ein Digraph G , Funktionen $l : E(G) \rightarrow \mathbb{R} \cup \{-\infty\}$, $u : E(G) \rightarrow \mathbb{R} \cup \{\infty\}$, $c : E(G) \rightarrow \mathbb{R}$ und eine kreuzend-submodulare Funktion $b : 2^{V(G)} \rightarrow \mathbb{R} \cup \{\infty\}$. Ein zulässiger submodularer Fluss ist eine Funktion $f : E(G) \rightarrow \mathbb{R}$ mit $l(e) \leq f(e) \leq u(e)$ für alle $e \in E(G)$ und

$$\sum_{e \in \delta^-(X)} f(e) - \sum_{e \in \delta^+(X)} f(e) \leq b(X)$$

für alle $X \subseteq V(G)$. Aufgabe ist es zu entscheiden, ob es einen zulässigen Fluss gibt und falls ja, einen solchen mit minimalen Kosten $\sum_{e \in E(G)} c(e)f(e)$ zu bestimmen.

Man zeige, dass dieses Problem das MINIMUM-COST-FLOW-PROBLEM und auch das Problem der Optimierung einer linearen Funktion auf dem Schnitt zweier Polymatroiden verallgemeinert.

Bemerkung: Das von Edmonds und Giles [1977] eingeführte SUBMODULARER-FLUSS-PROBLEM kann in streng polynomieller Zeit gelöst werden; siehe Fujishige, Röck und Zimmermann [1989]. Siehe auch Fleischer und Iwata [2000].

- * 13. Man zeige, dass das einen zulässigen submodularen Fluss beschreibende Ungleichungssystem (Aufgabe 12) TDI ist. Man zeige, dass daraus die Sätze 14.12 und 19.17 folgen.
(Edmonds und Giles [1977])
- 14. Man beweise, dass die Eckenmenge des Basispolyeders einer submodularen Funktion f mit $f(\emptyset) = 0$ genau die Menge der Vektoren b^\prec für alle vollständigen Ordnungen \prec von U ist, wobei für alle $u \in U$

$$b^\prec(u) := f(\{v \in U : v \preceq u\}) - f(\{v \in U : v \prec u\}).$$

Hinweis: Siehe den Beweis von Satz 14.11.

- 15. Sei $f : 2^U \rightarrow \mathbb{R}$ eine submodulare Funktion mit $f(\emptyset) = 0$ und Basispolyeder $B(f)$. Man beweise, dass $\min\{f(X) : X \subseteq U\} = \max\{\sum_{u \in U} z(u) : z(A) \leq \min\{0, f(A)\} \text{ für alle } A \subseteq U\} = \max\{\sum_{u \in U} \min\{0, y(u)\} : y \in B(f)\}$.

Hinweis: Man benutze Korollar 14.14 oder die Korrektheit von SCHRIJVERS ALGORITHMUS.

- 16. Man zeige, dass Lemma 8.41 ein Spezialfall von Lemma 14.24 ist.
- 17. Sei $f : 2^U \rightarrow \mathbb{R}$ eine submodulare Funktion. Sei ferner R eine zufällig gewählte Teilmenge von U , wobei jedes Element von R unabhängig mit der Wahrscheinlichkeit $\frac{1}{2}$ gewählt worden ist. Man beweise:
 - $\text{Exp}(f(R)) \geq \frac{1}{2}(f(\emptyset) + f(U))$.
 - Für jedes $A \subseteq U$ gilt $\text{Exp}(f(R)) \geq \frac{1}{4}(f(\emptyset) + f(A) + f(U \setminus A) + f(U))$.

Hinweis: Man wende (a) zweimal an.

- (c) Ist f nichtnegativ, so gilt $\text{Exp}(f(R)) \geq \frac{1}{4} \max_{A \subseteq U} f(A)$.

Bemerkung: Aus (c) folgt die Existenz eines randomisierten 4-Approximationsalgorithmus für die Maximierung (nichtnegativer) submodularer Funktionen. Dieses Problem kann nicht mit einer polynomiellen Anzahl von Orakelafrufen optimal gelöst werden.

(Feige, Mirrokni und Vondrák [2011])

- 18. Gegeben seien ein $0 < \epsilon < 1$ und ein gerades $n \in \mathbb{N}$ mit $\epsilon n \in \mathbb{N}$. Sei $U = \{1, \dots, n\}$. Betrachte für ein $C \subset U$ mit $2|C| = |U|$ die folgendermaßen definierten Funktionen $g, f_C : 2^U \rightarrow \mathbb{Z}_+$. Für $S \subseteq U$ seien $k := |S \cap C|$ und $l := |S \setminus C|$, und es seien $g(S) := |S||U \setminus S|$ und $f_C(S) := g(S)$, falls $|k - l| \leq \epsilon n$, und $f_C(S) := n|S| - 4kl + \epsilon^2 n^2 - 2\epsilon n|k - l|$, falls $|k - l| \geq \epsilon n$.

- (a) Man zeige, dass die zwei Definitionen von $f_C(S)$ übereinstimmen, wenn $|k - l| = \epsilon n$.
- (b) Man zeige, dass g und f_C submodular sind (*Hinweis*: Man benutze Aufgabe 4).
- * (c) Man beachte, dass ein Algorithmus wahrscheinlich exponentiell viele Orakelaufrufe benötigen wird, um festzustellen, welche dieser Funktionen (g oder f_C für irgendein C) der Input ist.
- (d) Man zeige, dass die maximalen Werte von f und irgendeinem g_C um einen Faktor größer als $2(1 - \epsilon)$ differieren.
 (Feige, Mirrokni und Vondrák [2011])
19. Man zeige: Ersetzt man den randomisierten Schritt im EINFACHEN MAXIMIERUNGSALGORITHMUS FÜR SUBMODULARE FUNKTIONEN, indem man $A := A \cup \{i\}$, falls $\Delta_A \geq \Delta_B$, und $B := B \setminus \{i\}$ sonst setzt, so bekommt man eine 3-Approximation (deterministisch und in linearer Zeit).
20. Man betrachte das Problem der Bestimmung einer Ecke des im Beweis von Satz 14.27 definierten Polytops.
- (a) Man ersetze eine der beiden nicht-trivialen Nebenbedingungen durch eine Zielfunktion und benutze dies, um zu zeigen, dass es eine Ecke gibt, in der höchstens eine Variable einen gebrochenen Wert annimmt.
- (b) Man zeige, wie man das aus (a) resultierende LP in $O(n \log n)$ Laufzeit lösen kann (man beachte, dass $|T_{i-1}| = O(n)$).
Bemerkung: Dieses LP kann sogar in linearer Zeit gelöst werden, indem man die Resultate von Abschnitt 17.1 heranzieht.
 (Buchbinder und Feldman [2016])

Literatur

Allgemeine Literatur:

- Bixby, R.E., und Cunningham, W.H. [1995]: Matroid optimization and algorithms. In: Handbook of Combinatorics; Vol. 1 (R.L. Graham, M. Grötschel, L. Lovász, Hrsg.), Elsevier, Amsterdam, 1995
- Björner, A., und Ziegler, G.M. [1992]: Introduction to greedoids. In: Matroid Applications (N. White, Hrsg.), Cambridge University Press, Cambridge 1992
- Frank, A. [2011]: Connections in Combinatorial Optimization. Oxford University Press, Oxford 2011
- Fujishige, S. [2005]: Submodular Functions and Optimization. 2. Aufl. Elsevier, Amsterdam 2005
- Iwata, S. [2008]: Submodular function minimization. Mathematical Programming B 112 (2008), 45–64
- Korte, B., Lovász, L., und Schrader, R. [1991]: Greedoids. Springer, Berlin 1991
- McCormick, S.T. [2004]: Submodular function minimization. In: Discrete Optimization (K. Aardal, G.L. Nemhauser, R. Weismantel, Hrsg.), Elsevier, Amsterdam 2005
- Schrijver, A. [2003]: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin 2003, Kapitel 44–49

Zitierte Literatur:

- Buchbinder, N., und Feldman, M. [2016]: Deterministic algorithms for submodular maximization problems. Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (2016), 392–403
- Buchbinder, N., Feldman, M., Naor, J., und Schwartz, R. [2015]: A tight linear time $(1/2)$ -approximation for unconstrained submodular maximization. SIAM Journal on Computing 44 (2015), 1384–1402
- Cunningham, W.H. [1984]: Testing membership in matroid polyhedra. Journal of Combinatorial Theory B 36 (1984), 161–188
- Edmonds, J. [1970]: Submodular functions, matroids and certain polyhedra. In: Combinatorial Structures and Their Applications; Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications 1969 (R. Guy, H. Hanani, N. Sauer, J. Schönheim, Hrsg.), Gordon and Breach, New York 1970, pp. 69–87
- Edmonds, J. [1979]: Matroid intersection. In: Discrete Optimization I; Annals of Discrete Mathematics 4 (P.L. Hammer, E.L. Johnson, B.H. Korte, Hrsg.), North-Holland, Amsterdam 1979, pp. 39–49
- Edmonds, J., und Giles, R. [1977]: A min-max relation for submodular functions on graphs. In: Studies in Integer Programming; Annals of Discrete Mathematics 1 (P.L. Hammer, E.L. Johnson, B.H. Korte, G.L. Nemhauser, Hrsg.), North-Holland, Amsterdam 1977, pp. 185–204
- Feige, U., Mirrokni, V.S., und Vondrák, J. [2011]: Maximizing non-monotone submodular functions. SIAM Journal on Computing 40 (2011), 1133–1153
- Fleischer, L., und Iwata, S. [2000]: Improved algorithms for submodular function minimization and submodular flow. Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (2000), 107–116
- Frank, A. [1981]: A weighted matroid intersection algorithm. Journal of Algorithms 2 (1981), 328–336
- Frank, A. [1982]: An algorithm for submodular functions on graphs. In: Bonn Workshop on Combinatorial Optimization; Annals of Discrete Mathematics 16 (A. Bachem, M. Grötschel, B. Korte, Hrsg.), North-Holland, Amsterdam 1982, pp. 97–120
- Fujishige, S. [1998]: Another simple proof of the validity of Nagamochi und Ibaraki's min-cut algorithm und Queyranne's extension to symmetric submodular function minimization. Journal of the Operations Research Society of Japan 41 (1998), 626–628
- Fujishige, S., Röck, H., und Zimmermann, U. [1989]: A strongly polynomial algorithm for minimum cost submodular flow problems. Mathematics of Operations Research 14 (1989), 60–69
- Grötschel, M., Lovász, L., und Schrijver, A. [1981]: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1 (1981), 169–197
- Grötschel, M., Lovász, L., und Schrijver, A. [1988]: Geometric Algorithms and Combinatorial Optimization. Springer, Berlin 1988
- Iwata, S. [2002]: A fully combinatorial algorithm for submodular function minimization. Journal of Combinatorial Theory B 84 (2002), 203–212
- Iwata, S. [2003]: A faster scaling algorithm for minimizing submodular functions. SIAM Journal on Computing 32 (2003), 833–840
- Iwata, S., Fleischer, L., und Fujishige, S. [2001]: A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. Journal of the ACM 48 (2001), 761–777

- Iwata, S., und Kobayashi, Y. [2017]: A weighted linear matroid parity algorithm. Proceedings of the 49th Annual ACM Symposium on Theory of Computing (2017), 264–276
- Jensen, P.M., und Korte, B. [1982]: Complexity of matroid property algorithms. SIAM Journal on Computing 11 (1982), 184–190
- Lee, Y.T., Sidford, A., und Wong, S.C. [2015]: A faster cutting plane method and its implications for combinatorial and convex optimization. Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (2015), 1049–1065
- Lovász, L. [1980]: Matroid matching and some applications. Journal of Combinatorial Theory B 28 (1980), 208–236
- Lovász, L. [1981]: The matroid matching problem. In: Algebraic Methods in Graph Theory; Vol. II (L. Lovász, V.T. Sós, Hrsg.), North-Holland, Amsterdam 1981, 495–517
- Lovász, L. [1983]: Submodular functions and convexity. In: Mathematical Programming: The State of the Art – Bonn 1982 (A. Bachem, M. Grötschel, B. Korte, Hrsg.), Springer, Berlin 1983
- Nagamochi, H., und Ibaraki, T. [1998]: A note on minimizing submodular functions. Information Processing Letters 67 (1998), 239–244
- Orlin, J.B. [2007]: A faster strongly polynomial time algorithm for submodular function minimization. Mathematical Programming 118 (2009), 237–251
- Pap, G. [2017+]. Weighted linear matroid matching. Erscheint demnächst. Abstract in: Oberwolfach Report 53 (2011), 3046–3049
- Queyranne, M. [1998]: Minimizing symmetric submodular functions. Mathematical Programming B 82 (1998), 3–12
- Rizzi, R. [2000]: On minimizing symmetric set functions. Combinatorica 20 (2000), 445–450
- Schrijver, A. [2000]: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. Journal of Combinatorial Theory B 80 (2000), 346–355
- Vygen, J. [2003]: A note on Schrijver’s submodular function minimization algorithm. Journal of Combinatorial Theory B 88 (2003), 399–402



15 NP-Vollständigkeit

Für viele kombinatorische Optimierungsprobleme gibt es polynomielle Algorithmen; die wichtigsten unter ihnen werden in diesem Buch besprochen. Es gibt jedoch auch viele wichtige Probleme, für die kein polynomieller Algorithmus bekannt ist. Obwohl wir nicht beweisen können, dass es diese nicht gibt, können wir aber zeigen, dass aus der Existenz eines polynomiellen Algorithmus für ein „schweres“ (genauer: *NP*-schweres) Problem die Existenz eines polynomiellen Algorithmus für fast alle in diesem Buch besprochenen Probleme (genauer: alle *NP*-leichten Probleme) folgt.

Um diese Begriffe formal einzuführen und die obige Aussage zu beweisen, brauchen wir ein Maschinenmodell, d. h. eine präzise Definition eines polynomiellen Algorithmus. Zu diesem Zwecke besprechen wir Turingmaschinen in Abschnitt 15.1. Mit diesem theoretischen Modell können keine komplizierten Algorithmen beschrieben werden. Wir werden jedoch darlegen, dass dieses Modell mit unserer intuitiven Auffassung von Algorithmen äquivalent ist: Jeder in diesem Buch besprochene Algorithmus kann theoretisch als eine Turing-Maschine dargestellt werden, mit einem polynomiell beschränkten Verlust an Effizienz. Dies steht in Abschnitt 15.2.

In Abschnitt 15.3 führen wir Entscheidungsprobleme ein, und insbesondere die Klassen *P* und *NP*. Während *NP* den Großteil der Entscheidungsprobleme in diesem Buch enthält, enthält *P* nur diejenigen, für die es einen polynomiellen Algorithmus gibt. Die Frage, ob $P = NP$ gilt, ist weiterhin offen. Obwohl wir viele Probleme in *NP* besprechen werden, für die kein polynomieller Algorithmus bekannt ist, hat bisher niemand in irgendeinem dieser Fälle beweisen können, dass es keinen solchen gibt. Wir werden angeben, was mit der Aussage, dass ein Problem auf ein anderes zurückgeführt oder reduziert werden kann, gemeint ist, und mit der Aussage, dass ein Problem mindestens so schwer ist wie ein anderes. In diesem Kontext sind die schwersten Probleme in *NP* die *NP*-vollständigen; diese können genau dann in polynomieller Zeit gelöst werden, wenn $P = NP$.

In Abschnitt 15.4 präsentieren wir das erste *NP*-vollständige Problem, nämlich SATISFIABILITY. In Abschnitt 15.5 beweisen wir die *NP*-Vollständigkeit einiger weiterer, enger mit der kombinatorischen Optimierung verbundener Entscheidungsprobleme. In den Abschnitten 15.6 und 15.7 werden wir weitere verwandte Begriffe einführen, die auch Optimierungsprobleme betreffen.

15.1 Turingmaschinen

In diesem Abschnitt stellen wir ein sehr einfaches Berechnungsmodell vor: die Turingmaschine. Sie kann als eine Folge von Befehlen einfacher Operationen auf Strings aufgefasst werden. Sowohl der Input als auch der Output besteht aus einem binären String.

Definition 15.1. Ein **Alphabet** ist eine endliche Menge mit mindestens zwei Elementen, aber ohne das besondere Element \sqcup , welches für Leerstellen reserviert ist. Ein **String** über einem Alphabet A ist eine endliche Folge von Elementen aus A . Mit A^n bezeichnen wir die Menge der Strings der Länge n und mit $A^* := \bigcup_{n \in \mathbb{Z}_+} A^n$ die Menge aller Strings über A . Wir benutzen die Konvention, dass A^0 genau ein Element enthält, nämlich den **leeren String**. Eine **Sprache** über A ist eine Teilmenge von A^* . Die Elemente einer Sprache werden oft die **Wörter** der Sprache genannt. Für $x \in A^n$ schreiben wir $\text{size}(x) := n$ für die **Länge** des Strings.

Wir werden viel mit dem Alphabet $A = \{0, 1\}$ und der Menge $\{0, 1\}^*$ aller **0-1-Strings** (oder **binären Strings**) arbeiten. Die Komponenten eines 0-1-Strings werden gelegentlich als seine **Bits** bezeichnet. Es gibt also genau einen 0-1-String der Länge Null, nämlich den leeren String.

Der Input einer Turingmaschine besteht aus einem String $x \in A^*$ für ein festes Alphabet A . Dieser Input wird durch Leerstellen (Elemente \sqcup) zu einem in beiden Richtungen unendlichen String $s \in (A \cup \{\sqcup\})^{\mathbb{Z}}$ erweitert. Der String s kann als ein Band mit Schreib-Lese-Kopf aufgefasst werden; bei jedem Schritt kann nur eine Stelle gelesen und verändert werden und der Schreib-Lese-Kopf um nur eine Stelle bewegt werden.

Eine Turingmaschine besteht aus einer Menge von $N + 1$ mit $0, \dots, N$ nummerierten Befehlen. Am Anfang wird Befehl 0 ausgeführt und der String steht auf Stelle 1. Nun hat jeder Befehl die folgende Form. Lese das Bit an der aktuellen Stelle und führe Folgendes aus, je nach Wert des Bits: Überschreibe das aktuelle Bit mit einem Element aus $A \cup \{\sqcup\}$, führe eventuell eine Bewegung um eine Stelle nach rechts oder links aus, und gehe zu einem als nächster auszuführenden Befehl.

Es gibt einen besonderen Befehl, der mit -1 bezeichnet wird und das Ende der Berechnung angibt. Die Komponenten unseres unendlichen Strings s mit den Indizes $1, 2, 3, \dots$ bis zum ersten \sqcup ergeben dann den Output-String. Die formale Definition einer Turingmaschine lautet wie folgt:

Definition 15.2. (Turing [1936]) Sei A ein Alphabet und $\bar{A} := A \cup \{\sqcup\}$. Eine **Turingmaschine** (mit Alphabet A) ist eine Funktion

$$\Phi : \{0, \dots, N\} \times \bar{A} \rightarrow \{-1, \dots, N\} \times \bar{A} \times \{-1, 0, 1\}$$

für ein $N \in \mathbb{Z}_+$. Die **Berechnung** von Φ für den Input x mit $x \in A^*$ ist die endliche oder unendliche Folge von Tripeln $(n^{(i)}, s^{(i)}, \pi^{(i)})$ mit $n^{(i)} \in \{-1, \dots, N\}$, $s^{(i)} \in \bar{A}^{\mathbb{Z}}$ und $\pi^{(i)} \in \mathbb{Z}$ ($i = 0, 1, 2, \dots$), die wie folgt rekursiv definiert wird (es bezeichne $n^{(i)}$ den aktuellen Befehl, $s^{(i)}$ den String und $\pi^{(i)}$ die aktuelle Stelle):

- $n^{(0)} := 0$. $s_j^{(0)} := x_j$ für $1 \leq j \leq \text{size}(x)$ und $s_j^{(0)} := \sqcup$ für alle $j \leq 0$ und $j > \text{size}(x)$. $\pi^{(0)} := 1$.

Ist $(n^{(i)}, s^{(i)}, \pi^{(i)})$ bereits definiert, so gibt es zwei Fälle:

- Ist $n^{(i)} \neq -1$, so sei $(m, \sigma, \delta) := \Phi(n^{(i)}, s_{\pi^{(i)}}^{(i)})$ und setze $n^{(i+1)} := m$, $s_{\pi^{(i)}}^{(i+1)} := \sigma$, $s_j^{(i+1)} := s_j^{(i)}$ für $j \in \mathbb{Z} \setminus \{\pi^{(i)}\}$ und $\pi^{(i+1)} := \pi^{(i)} + \delta$.
- Ist $n^{(i)} = -1$, so ist hier das Ende der Folge. Dann definieren wir $\text{time}(\Phi, x) := i$ und $\text{output}(\Phi, x) \in A^k$ mit $k := \min\{j \in \mathbb{N} : s_j^{(i)} = \sqcup\} - 1$ durch $\text{output}(\Phi, x)_j := s_j^{(i)}$ für $j = 1, \dots, k$.

Ist diese Folge unendlich (d. h. $n^{(i)} \neq -1$ für alle i), so setzen wir $\text{time}(\Phi, x) := \infty$. In diesem Fall ist $\text{output}(\Phi, x)$ undefiniert.

Natürlich interessieren uns hauptsächlich Turingmaschinen, deren Berechnung endlich oder sogar polynomiell beschränkt ist:

Definition 15.3. Sei A ein Alphabet. Ein **Berechnungsproblem** besteht aus einem Paar (X, R) , wobei $X \subseteq A^*$ eine Sprache ist und $R \subseteq X \times A^*$ eine Relation mit der Eigenschaft: Für jedes $x \in X$ gibt es ein $y \in A^*$ mit $(x, y) \in R$. Sei ferner Φ eine Turingmaschine mit Alphabet A , so dass $\text{time}(\Phi, x) < \infty$ und $(x, \text{output}(\Phi, x)) \in R$ für jedes $x \in X$. Dann sagen wir: Φ berechnet (X, R) . Gibt es ein Polynom p , so dass $\text{time}(\Phi, x) \leq p(\text{size}(x))$ für alle $x \in X$, so heißt Φ eine **polynomielle Turingmaschine** (für (X, R)).

Gilt $|\{y \in A^* : (x, y) \in R\}| = 1$ für alle $x \in X$, so können wir $f : X \rightarrow A^*$ durch $(x, f(x)) \in R$ definieren und sagen: Φ berechnet f . Ist insbesondere $X = A^*$ und $f : X \rightarrow \{0, 1\}$, so sagen wir: Φ entscheidet die Sprache $L := \{x \in X : f(x) = 1\}$. Gibt es eine polynomielle Turingmaschine, die eine Funktion f berechnet (bzw. eine Sprache L entscheidet), so sagen wir: f ist in **polynomieller Zeit berechenbar** (bzw. L ist in **polynomieller Zeit entscheidbar**).

Zum besseren Verständnis dieser Definitionen geben wir nun ein Beispiel. Die folgende Turingmaschine $\Phi : \{0, \dots, 3\} \times \{0, 1, \sqcup\} \rightarrow \{-1, \dots, 3\} \times \{0, 1, \sqcup\} \times \{-1, 0, 1\}$ berechnet die Nachfolger-Funktion $n \mapsto n + 1$ ($n \in \mathbb{N}$), wobei Zahlen wie üblich binär kodiert werden.

$\Phi(0, 0) = (0, 0, 1)$	① While $s_\pi \neq \sqcup$ do $\pi := \pi + 1$.
$\Phi(0, 1) = (0, 1, 1)$	
$\Phi(0, \sqcup) = (1, \sqcup, -1)$	Setze $\pi := \pi - 1$.
$\Phi(1, 1) = (1, 0, -1)$	① While $s_\pi = 1$ do $s_\pi := 0$ und $\pi := \pi - 1$.
$\Phi(1, 0) = (-1, 1, 0)$	If $s_\pi = 0$ then $s_\pi := 1$ und stop .
$\Phi(1, \sqcup) = (2, \sqcup, 1)$	Setze $\pi := \pi + 1$.
$\Phi(2, 0) = (3, 1, 1)$	② Setze $s_\pi = 1$ und $\pi := \pi + 1$.
$\Phi(3, 0) = (3, 0, 1)$	③ While $s_\pi = 0$ do $\pi := \pi + 1$.
$\Phi(3, \sqcup) = (-1, 0, 0)$	Setze $s_\pi := 0$ und stop .

Beachte, dass einige Φ -Werte nicht angegeben wurden, da sie nirgends vorkommen. Die Kommentare auf der rechten Seite erläutern die Berechnungen. Die Befehle ② und ③ werden nur dann gebraucht, wenn der Input aus lauter Einsen besteht, d.h. $n = 2^k - 1$ für ein $k \in \mathbb{N}$. Es gilt $\text{time}(\Phi, x) \leq 3 \text{size}(x) + 3$ für alle Inputs x , somit ist Φ eine polynomielle Turingmaschine.

Im folgenden Abschnitt werden wir zeigen, dass die obige Definition mit unserer in Abschnitt 1.2 angegebenen informellen Definition eines polynomiellen Algorithmus konsistent ist: Jeder in diesem Buch auftretende polynomielle Algorithmus kann mittels einer polynomiellen Turingmaschine simuliert werden.

15.2 Die Church'sche These

Die Turingmaschine ist das am meisten verwendete theoretische Modell für Algorithmen. Obwohl sie den Eindruck erweckt, nur von sehr beschränktem Nutzen zu sein, ist sie so mächtig wie jedes andere vernünftige Modell: Sowohl die Menge der **berechenbaren Funktionen** als auch die Menge der polynomiell berechenbaren Funktionen ist immer die gleiche. Diese Aussage, bekannt als die Church'sche These, ist offensichtlich zu ungenau, als dass man sie beweisen könnte. Es gibt jedoch starke Resultate, die diese These untermauern. Es kann z.B. jedes in einer bekannten Programmiersprache wie C geschriebene Programm als eine Turingmaschine modelliert werden. Insbesondere können alle in diesem Buch vorkommenden Algorithmen als Turingmaschinen umgeschrieben werden. Das ist normalerweise sehr umständlich (weswegen wir es auch nicht tun), es ist aber theoretisch möglich. Zudem ist auch jede mittels eines C-Programms in polynomieller Zeit berechenbare Funktion mittels einer Turingmaschine in polynomieller Zeit berechenbar (und umgekehrt).

Da es keineswegs eine triviale Aufgabe ist, kompliziertere Programme mittels einer Turingmaschine zu implementieren, betrachten wir nun als Zwischenstufe eine Turingmaschine mit zwei Strings (Bändern) und zwei unabhängigen Schreib-Lese-Köpfen, jeweils einen für die beiden Bänder:

Definition 15.4. Sei A ein Alphabet und $\bar{A} := A \cup \{\sqcup\}$. Eine **2-Band-Turingmaschine** ist eine Funktion

$$\Phi : \{0, \dots, N\} \times \bar{A}^2 \rightarrow \{-1, \dots, N\} \times \bar{A}^2 \times \{-1, 0, 1\}^2$$

für ein $N \in \mathbb{Z}_+$. Die **Berechnung** von Φ für den Input x mit $x \in A^*$ ist die endliche oder unendliche Folge von 5-Tupeln $(n^{(i)}, s^{(i)}, t^{(i)}, \pi^{(i)}, \rho^{(i)})$ mit $n^{(i)} \in \{-1, \dots, N\}$, $s^{(i)}, t^{(i)} \in \bar{A}^\mathbb{Z}$ und $\pi^{(i)}, \rho^{(i)} \in \mathbb{Z}$ ($i = 0, 1, 2, \dots$), die wie folgt rekursiv definiert wird:

- $n^{(0)} := 0$. $s_j^{(0)} := x_j$ für $1 \leq j \leq \text{size}(x)$ und $s_j^{(0)} := \sqcup$ für alle $j \leq 0$ und $j > \text{size}(x)$. $t_j^{(0)} := \sqcup$ für alle $j \in \mathbb{Z}$. $\pi^{(0)} := 1$ und $\rho^{(0)} := 1$.

Ist $(n^{(i)}, s^{(i)}, t^{(i)}, \pi^{(i)}, \rho^{(i)})$ bereits definiert, so gibt es zwei Fälle:

- Ist $n^{(i)} \neq -1$, so sei $(m, \sigma, \tau, \delta, \epsilon) := \Phi(n^{(i)}, s_{\pi^{(i)}}^{(i)}, t_{\rho^{(i)}}^{(i)})$ und setze $n^{(i+1)} := m$, $s_{\pi^{(i)}}^{(i+1)} := \sigma$, $s_j^{(i+1)} := s_j^{(i)}$ für $j \in \mathbb{Z} \setminus \{\pi^{(i)}\}$, $t_{\rho^{(i)}}^{(i+1)} := \tau$, $t_j^{(i+1)} := t_j^{(i)}$ für $j \in \mathbb{Z} \setminus \{\rho^{(i)}\}$, $\pi^{(i+1)} := \pi^{(i)} + \delta$ und $\rho^{(i+1)} := \rho^{(i)} + \epsilon$.
- Ist $n^{(i)} = -1$, so ist hier das Ende der Folge. Es werden $\text{time}(\Phi, x)$ und $\text{output}(\Phi, x)$ wie bei der 1-Band-Turingmaschine definiert.

Turingmaschinen mit mehr als zwei Bändern können analog definiert werden, wir benötigen diese aber nicht. Bevor wir zeigen, wie man Standard-Operationen mit einer 2-Band-Turingmaschine ausführt, möchten wir darauf hinweisen, dass eine 2-Band-Turingmaschine durch eine gewöhnliche (1-Band-) Turingmaschine simuliert werden kann.

Satz 15.5. Sei A ein Alphabet und

$$\Phi : \{0, \dots, N\} \times (A \cup \{\sqcup\})^2 \rightarrow \{-1, \dots, N\} \times (A \cup \{\sqcup\})^2 \times \{-1, 0, 1\}^2$$

eine 2-Band-Turingmaschine. Dann gibt es ein Alphabet $B \supseteq A$ und eine (1-Band-) Turingmaschine

$$\Phi' : \{0, \dots, N'\} \times (B \cup \{\sqcup\}) \rightarrow \{-1, \dots, N'\} \times (B \cup \{\sqcup\}) \times \{-1, 0, 1\}$$

mit $\text{output}(\Phi', x) = \text{output}(\Phi, x)$ und $\text{time}(\Phi', x) = O((\text{time}(\Phi, x))^2)$ für $x \in A^*$.

Beweis: Wir bezeichnen mit s und t die beiden Strings von Φ und mit π und ρ die von den Schreib-Lese-Köpfen eingenommenen Stellen, entsprechend Definition 15.4. Den String von Φ' bezeichnen wir mit u und die von seinem Schreib-Lese-Kopf eingenommene Stelle mit ψ .

Wir müssen beide Strings s, t und beide Stellen π, ρ in einem String u kodieren. Um dies zu ermöglichen, schreiben wir jedes Symbol u_j von u als 4-Tupel (s_j, p_j, t_j, r_j) , wobei s_j und t_j die entsprechenden Symbole von s und t sind und $p_j, r_j \in \{0, 1\}$ anzeigen, ob der Schreib-Lese-Kopf des ersten bzw. des zweiten Strings gerade die Stelle j scannt, d.h. es gilt $p_j = 1$ genau dann, wenn $\pi = j$, und $r_j = 1$ genau dann, wenn $\rho = j$.

Somit definieren wir $\bar{B} := (\bar{A} \times \{0, 1\} \times \bar{A} \times \{0, 1\})$; dann identifizieren wir $a \in \bar{A}$ mit $(a, 0, \sqcup, 0)$, um Inputs aus A^* zu ermöglichen. Der erste Schritt von Φ' besteht darin, die Marker p_1 und r_1 auf 1 zu setzen:

$$\Phi'(0, (., 0, ., 0)) = (1, (., 1, ., 1), 0) \quad \text{① Setze } \pi := \psi \text{ und } \rho := \psi.$$

Ein Punkt bezeichnet hier einen beliebigen Wert, der jedoch nicht verändert wird.

Nun zeigen wir, wie man einen allgemeinen Befehl $\Phi(m, \sigma, \tau) = (m', \sigma', \tau', \delta, \epsilon)$ implementiert. Zunächst müssen wir die Stellen π und ρ bestimmen. Hier ist es hilfreich, wenn wir annehmen, dass sich unser Einzel-Schreib-Lese-Kopf ψ bereits an der linken der beiden Stellen π und ρ befindet, d.h. $\psi = \min\{\pi, \rho\}$. Die andere Stelle finden wir, indem wir den String u nach rechts scannen; dann müssen wir prüfen, ob $s_\pi = \sigma$ und $t_\rho = \tau$ und falls ja, die verlangte Operation ausführen (neue Symbole für s und t schreiben, π und ρ bewegen und zum nächsten Befehl springen).

Der unten angegebene Block implementiert einen Befehl $\Phi(m, \sigma, \tau) = (m', \sigma', \tau', \delta, \epsilon)$ für $m = 0$; für jedes m haben wir $|\bar{A}|^2$ solche Blöcke, einen für jede Wahl von σ und τ . Der zweite Block für $m = 0$ beginnt mit ⑬ und der erste Block für m' mit ⑭, wobei $M := 12|\bar{A}|^2 m' + 1$. Insgesamt haben wir $N' = 12(N + 1)|\bar{A}|^2$.

Ein Punkt bezeichnet wiederum einen beliebigen Wert, der nicht verändert wird. Analog bezeichnet ζ bzw. ξ einen beliebigen Wert aus $\bar{A} \setminus \{\sigma\}$ bzw. $\bar{A} \setminus \{\tau\}$. Wir nehmen an, dass zu Beginn $\psi = \min\{\pi, \rho\}$ ist; beachte, dass ⑩, ⑪ und ⑫ gewährleisten, dass dies auch am Ende gilt.

$\Phi'(1, (\zeta, 1, ., .))$	$= (13, (\zeta, 1, ., .), 0)$	① If $\psi = \pi$ und $s_\psi \neq \sigma$ then go to ⑬.
$\Phi'(1, (., ., \zeta, 1))$	$= (13, (., ., \zeta, 1), 0)$	If $\psi = \rho$ und $t_\psi \neq \tau$ then go to ⑬.
$\Phi'(1, (\sigma, 1, \tau, 1))$	$= (2, (\sigma, 1, \tau, 1), 0)$	If $\psi = \pi$ then go to ②.
$\Phi'(1, (\sigma, 1, ., 0))$	$= (2, (\sigma, 1, ., 0), 0)$	If $\psi = \rho$ then go to ⑥.
$\Phi'(1, (., 0, \tau, 1))$	$= (6, (., 0, \tau, 1), 0)$	② While $\psi \neq \rho$ do $\psi := \psi + 1$.
$\Phi'(2, (., ., ., 0))$	$= (2, (., ., ., 0), 1)$	If $t_\psi \neq \tau$ then setze $\psi := \psi - 1$
$\Phi'(2, (., ., \xi, 1))$	$= (12, (., ., \xi, 1), -1)$	und go to ⑫.
$\Phi'(2, (., ., \tau, 1))$	$= (3, (., ., \tau', 0), \epsilon)$	Setze $t_\psi := \tau'$ und $\psi := \psi + \epsilon$.
$\Phi'(3, (., ., ., 0))$	$= (4, (., ., ., 1), 1)$	③ Setze $\rho := \psi$ und $\psi := \psi + 1$.
$\Phi'(4, (., 0, ., .))$	$= (4, (., 0, ., .), -1)$	④ While $\psi \neq \pi$ do $\psi := \psi - 1$.
$\Phi'(4, (\sigma, 1, ., .))$	$= (5, (\sigma', 0, ., .), \delta)$	Setze $s_\psi := \sigma'$ und $\psi := \psi + \delta$.
$\Phi'(5, (., 0, ., .))$	$= (10, (., 1, ., .), -1)$	⑤ Setze $\pi := \psi$ und $\psi := \psi - 1$.
		Go to ⑩.
$\Phi'(6, (., 0, ., .))$	$= (6, (., 0, ., .), 1)$	⑥ While $\psi \neq \pi$ do $\psi := \psi + 1$.
$\Phi'(6, (\zeta, 1, ., .))$	$= (12, (\zeta, 1, ., .), -1)$	If $s_\psi \neq \sigma$ then setze $\psi := \psi - 1$
		und go to ⑫.
$\Phi'(6, (\sigma, 1, ., .))$	$= (7, (\sigma', 0, ., .), \delta)$	Setze $s_\psi := \sigma'$ und $\psi := \psi + \delta$.
$\Phi'(7, (., 0, ., .))$	$= (8, (., 1, ., .), 1)$	⑦ Setze $\pi := \psi$ und $\psi := \psi + 1$.
$\Phi'(8, (., ., ., 0))$	$= (8, (., ., ., 0), -1)$	⑧ While $\psi \neq \rho$ do $\psi := \psi - 1$.
$\Phi'(8, (., ., \tau, 1))$	$= (9, (., ., \tau', 0), \epsilon)$	Setze $t_\psi := \tau'$ und $\psi := \psi + \epsilon$.
$\Phi'(9, (., ., ., 0))$	$= (10, (., ., ., 1), -1)$	⑨ Setze $\rho := \psi$ und $\psi := \psi - 1$.
$\Phi'(10, (., ., ., .))$	$= (11, (., ., ., .), -1)$	⑩ Setze $\psi := \psi - 1$.
$\Phi'(11, (., 0, ., 0))$	$= (11, (., 0, ., 0), 1)$	⑪ While $\psi \notin \{\pi, \rho\}$ do $\psi := \psi + 1$.
$\Phi'(11, (., 1, ., .))$	$= (M, (., 1, ., .), 0)$	Go to ⑭.
$\Phi'(11, (., 0, ., 1))$	$= (M, (., 0, ., 1), 0)$	
$\Phi'(12, (., 0, ., 0))$	$= (12, (., 0, ., 0), -1)$	⑫ While $\psi \notin \{\pi, \rho\}$ do $\psi := \psi - 1$.
$\Phi'(12, (., 1, ., .))$	$= (13, (., 1, ., .), 0)$	
$\Phi'(12, (., ., ., 1))$	$= (13, (., ., ., 1), 0)$	

Jede Berechnung von Φ' passiert höchstens $|\bar{A}|^2$ solche Blöcke für jeden Berechnungsschritt von Φ . Die Anzahl der Berechnungsschritte in jedem Block ist höchstens gleich $2|\pi - \rho| + 10$. Da $|\bar{A}|$ eine Konstante ist und $|\pi - \rho|$ durch $\text{time}(\Phi, x)$ beschränkt ist, folgt, dass die gesamte Berechnung von Φ durch Φ' mit $O((\text{time}(\Phi, x))^2)$ Schritten simuliert wird.

Abschließend müssen wir noch den Output umformen: Ersetze jedes Symbol $(\sigma, ., ., .)$ durch $(\sigma, 0, \sqcup, 0)$. Offensichtlich wird dadurch die Gesamtanzahl der Schritte höchstens verdoppelt. \square

Mit einer 2-Band-Turingmaschine ist es nicht allzu schwer, kompliziertere Befehle und somit auch allgemeine Algorithmen, zu implementieren:

Wir benutzen das Alphabet $A = \{0, 1, \#\}$ und modellieren eine beliebige Anzahl von Variablen mittels des Strings

$$x_0\#\#1\#x_1\#\#10\#x_2\#\#11\#x_3\#\#100\#x_4\#\#101\#x_5\#\#\dots \quad (15.1)$$

den wir auf dem ersten Band speichern. Jede Gruppe (außer der ersten) enthält eine binäre Darstellung des Indexes i , gefolgt von dem Wert von x_i , von dem wir annehmen, dass er ein binärer String ist. Die erste Variable x_0 und das zweite Band werden nur als Register für Zwischenergebnisse der Berechnungsschritte benutzt.

Der wahlfreie Zugriff auf Variablen in konstanter Zeit ist mit einer Turingmaschine nicht möglich, auch nicht mit beliebig vielen Bändern. Simuliert man einen beliebigen Algorithmus mit einer 2-Band-Turingmaschine, so müssen wir das erste Band recht oft scannen. Wird auch die Länge des Strings in einer Variable verändert, so muss der rechte Teilstring verschoben werden. Trotzdem kann jede Standard-Operation (d. h. jeder elementare Schritt eines Algorithmus) mit $O(l^2)$ Berechnungsschritten einer 2-Band-Turingmaschine simuliert werden, wobei l die aktuelle Länge des Strings (15.1) ist.

Wir werden diesen Sachverhalt an einem Beispiel erläutern. Betrachte den folgenden Befehl: Addiere den Wert der Variable, deren Index durch x_2 gegeben ist, zu x_5 .

Um den Wert von x_5 zu bestimmen, scannen wir das erste Band nach dem Teilstring $\#\#101\#$. Dann kopieren wir den darauf folgenden Teilstring bis $\#$, aber nicht inklusive diesem $\#$, auf das zweite Band. Dies ist einfach, da wir zwei separate Schreib-Lese-Köpfe haben. Dann kopieren wir den String vom zweiten Band auf x_0 . Ist der neue Wert von x_0 kürzer oder länger als der alte, so müssen wir den restlichen Teil des Strings (15.1) nach links bzw. nach rechts bewegen.

Als Nächstes müssen wir den durch x_2 angegebenen Variablenindex suchen. Um dies zu erreichen, kopieren wir x_2 auf das zweite Band. Dann können wir das erste Band scannen und jeden Variablenindex prüfen (indem wir ihn bitweise mit dem String auf dem zweiten Band vergleichen). Haben wir den korrekten Variablenindex gefunden, so kopieren wir den Wert dieser Variable auf das zweite Band.

Nun addieren wir die in x_0 gespeicherte Zahl zu derjenigen auf dem zweiten Band. Eine Turingmaschine hierfür lässt sich ohne weiteres mit den Standardmethoden entwerfen. Die Zahl auf dem zweiten Band können wir mit dem Resultat

überschreiben, während wir dieses berechnen. Schließlich haben wir das Resultat auf dem zweiten Band und kopieren es dann zurück auf x_5 . Eventuell ist noch eine Bewegung des Teilstrings rechts von x_5 notwendig.

Alles obige kann von einer 2-Band-Turingmaschine in $O(l^2)$ Berechnungsschritten bewerkstelligt werden (in der Tat kann alles, bis auf die Bewegung des Strings (15.1), in $O(l)$ Schritten erreicht werden). Es sollte klar sein, dass dasselbe für alle anderen Standard-Operationen gilt, auch für Multiplikation und Division.

Nach Definition 1.4 hat ein Algorithmus eine polynomielle Laufzeit, wenn es ein $k \in \mathbb{N}$ gibt, so dass die Anzahl der elementaren Schritte durch $O(n^k)$ beschränkt ist und jede in Zwischenberechnungen vorkommende Zahl mit $O(n^k)$ Bits gespeichert werden kann, wobei n die Inputgröße ist. Ferner speichern wir höchstens $O(n^k)$ Zahlen zu irgendeinem Zeitpunkt. Somit können wir die Länge eines jeden der beiden Strings in einer 2-Band-Turingmaschine, die einen solchen Algorithmus simuliert, durch $l = O(n^k \cdot n^k) = O(n^{2k})$ beschränken. Also wird die Gesamlaufzeit durch $O(n^k(n^{2k})^2) = O(n^{5k})$ beschränkt. Dies ist immer noch polynomiell in der Inputgröße.

Mit Satz 15.5 folgt sodann, dass es für eine Funktion f genau dann einen f berechnenden polynomiellen Algorithmus gibt, wenn es eine f berechnende polynomielle Turingmaschine gibt. Deswegen werden wir in diesem Kapitel fortan die Begriffe Algorithmus und Turingmaschine als synonym betrachten.

Hopcroft und Ullman [1979], Lewis und Papadimitriou [1981] und van Emde Boas [1990] haben die Äquivalenz verschiedener Maschinenmodelle im Detail untersucht. Ein anderes bekanntes Modell (welches unserem informellen Modell in Abschnitt 1.2 ähnlich ist) ist die RAM-Maschine (siehe Aufgabe 3), welche arithmetische Operationen mit ganzen Zahlen in konstanter Zeit ermöglicht. Weitere Modelle erlauben nur Operationen mit Bits (oder ganzen Zahlen einer festen Länge), was im Umgang mit großen Zahlen realistischer ist. Offensichtlich kann man Addition und Vergleich ganzer Zahlen mit n Bits mit $O(n)$ Bit-Operationen bewerkstelligen. Für Multiplikation (und Division) braucht die nahe liegende Methode $O(n^2)$ Bit-Operationen, der Algorithmus von Schönhage und Strassen [1971] aber nur $O(n \log n \log \log n)$ Bit-Operationen für die Multiplikation zweier ganzer Zahlen mit n Bits. Dies wurde von Fürer [2009] und Harvey, van der Hoeven und Lecerf [2016] noch verbessert. Natürlich folgt hiermit auch die Existenz von Algorithmen mit derselben Zeitkomplexität für Addition und Vergleich rationaler Zahlen. Was die Berechenbarkeit in polynomieller Zeit betrifft, so sind alle Modelle äquivalent, aber die Laufzeiten sind natürlich gänzlich verschieden.

Die Modellierung, welche die Kodierung des gesamten Inputs mittels 0-1-Strings (oder Strings über einem festen Alphabet) vorsieht, schließt die Kodierung bestimmter Zahlentypen im Prinzip nicht von vornherein aus, z. B. algebraische Zahlen (ist $x \in \mathbb{R}$ die k -kleinste Wurzel eines Polynoms p , so kann man x kodieren, indem man k und den Grad und die Koeffizienten von p auflistet). Es gibt jedoch keine Möglichkeit, eine beliebige reelle Zahl in einem digitalen Rechner darzustellen, da es überabzählbar viele reelle Zahlen aber nur abzählbar viele 0-

1-Strings gibt. Hier schließen wir uns dem klassischen Weg an und beschränken uns in diesem Kapitel auf rationale Inputs.

Wir schließen diesen Abschnitt mit einer formalen, auf 2-Band-Turingmaschinen basierenden Definition von Orakel-Algorithmen. Ein Orakel können wir zu jedem Zeitpunkt der Berechnung aufrufen; wir verwenden das zweite Band zum Schreiben des Orakel-Inputs und zum Lesen des Orakel-Outputs. Ferner führen wir einen weiteren Befehl, nämlich -2 , für Orakel-Aufrufe ein.

Definition 15.6. Sei A ein Alphabet und $\bar{A} := A \cup \{\square\}$. Sei (X, R) ein Berechnungsproblem mit $X \subseteq A^*$. Eine (X, R) benutzende **Orakel-Turingmaschine** ist eine Funktion

$$\Phi : \{0, \dots, N\} \times \bar{A}^2 \rightarrow \{-2, \dots, N\} \times \bar{A}^2 \times \{-1, 0, 1\}^2$$

für ein $N \in \mathbb{Z}_+$. Die Berechnung von Φ wird wie für eine 2-Band-Turingmaschine definiert, bis auf die folgenden Unterschiede: Am Anfang setzen wir $\text{time}^{(0)} := 0$. Gilt für einen Berechnungsschritt i , dass $\Phi(n^{(i)}, s_{\pi^{(i)}}^{(i)}, t_{\rho^{(i)}}^{(i)}) = (-2, \sigma, \tau, \delta, \epsilon)$ für irgendwelche $\sigma, \tau, \delta, \epsilon$, dann betrachte den auf dem zweiten Band liegenden String $x \in A^k$ mit $k := \min \{j \in \mathbb{N} : t_j^{(i)} = \square\} - 1$, gegeben durch $x_j := t_j^{(i)}$ für $j = 1, \dots, k$. Ist $x \in X$, so überschreibe das zweite Band mit $t_j^{(i+1)} = y_j$ für $j = 1, \dots, \text{size}(y)$ und $t_{\text{size}(y)+1}^{(i+1)} = \square$ für ein $y \in A^*$ mit $(x, y) \in R$, setze $\text{time}^{(i+1)} := \text{time}^{(i)} + 1 + \text{size}(y)$, und setze die Berechnung mit $n^{(i+1)} := n^{(i)} + 1$ fort. Der Rest bleibt unverändert, und in allen anderen Fällen setzen wir $\text{time}^{(i+1)} := \text{time}^{(i)} + 1$. Wie üblich endet die Berechnung, wenn $n^{(i)} = -1$; Dann setzen wir $\text{time}(\Phi, x) := \text{time}^{(i)}$. Der Output wird wie bei der 2-Band-Turingmaschine definiert.

Alle Definitionen bezüglich Turingmaschinen können auf Orakel-Turingmaschinen erweitert werden. Ein Orakel-Output ist nicht notwendigerweise eindeutig bestimmt; somit gibt es möglicherweise verschiedene Berechnungen für denselben Input. Bei dem Beweis der Korrektheit oder der Laufzeitabschätzung eines Orakel-Algorithmus müssen wir alle möglichen Berechnungen berücksichtigen, d. h. alle Orakel-Outputs.

Die Ergebnisse dieses Abschnitts zeigen: Die Existenz eines polynomiellen (Orakel)-Algorithmus ist äquivalent mit der Existenz einer polynomiellen (Orakel)-Turingmaschine.

15.3 P und NP

Die Komplexitätstheorie basiert größtenteils auf Entscheidungsproblemen. Diese sind besondere Berechnungsprobleme. Jede Sprache $L \subseteq \{0, 1\}^*$ kann als Entscheidungsproblem aufgefasst werden: Entscheide für einen gegebenen 0-1-String, ob er aus L ist. Wir sind jedoch mehr an anderen Problemen interessiert, z. B. an folgendem:

HAMILTON-KREIS

Instanz: Ein ungerichteter Graph G .

Frage: Besitzt G einen Hamilton-Kreis?

Wir werden immer eine feste effiziente Kodierung des Inputs als binären String voraussetzen; gelegentlich erweitern wir unser Alphabet um einige Symbole. Z. B. gehen wir davon aus, dass ein Graph mittels einer Adjazenzliste gegeben wird, und eine solche Liste kann als binärer String der Länge $O(n \log m + m \log n)$ kodiert werden, wobei n bzw. m die Anzahl der Knoten bzw. Kanten ist. Wir setzen immer eine effiziente Kodierung voraus, d. h. als binären String, dessen Länge polynomiell durch die kleinstmögliche Kodierungslänge beschränkt ist.

Nicht alle binären Strings sind Instanzen von HAMILTON-KREIS, sondern nur diejenigen, welche ungerichtete Graphen darstellen. Für die meisten interessanten Entscheidungsprobleme bilden die Instanzen eine echte Teilmenge der Menge der 0-1-Strings. Wir verlangen, dass wir in polynomieller Zeit entscheiden können, ob ein beliebiger String eine Instanz ist:

Definition 15.7. Ein Entscheidungsproblem ist ein Paar $\mathcal{P} = (X, Y)$, wobei X eine in polynomieller Zeit entscheidbare Sprache und $Y \subseteq X$ ist. Die Elemente von X heißen **Instanzen** von \mathcal{P} ; die Elemente von Y sind die **Ja-Instanzen**, diejenigen von $X \setminus Y$ die **Nein-Instanzen**.

Ein Entscheidungsproblem (X, Y) kann als das Berechnungsproblem $(X, \{(x, 1) : x \in Y\} \cup \{(x, 0) : x \in X \setminus Y\})$ aufgefasst werden. Somit ist ein Algorithmus für ein Entscheidungsproblem (X, Y) ein Algorithmus, welcher die Funktion $f : X \rightarrow \{0, 1\}$, gegeben durch $f(x) = 1$ für $x \in Y$ und $f(x) = 0$ für $x \in X \setminus Y$, berechnet.

Es folgen zwei weitere Beispiele, nämlich die Entscheidungsprobleme, die den Problemen LINEARE OPTIMIERUNG und GANZZAHLIGE OPTIMIERUNG entsprechen:

LINEARE UNGLEICHUNGEN

Instanz: Eine Matrix $A \in \mathbb{Z}^{m \times n}$ und ein Vektor $b \in \mathbb{Z}^m$.

Frage: Gibt es einen Vektor $x \in \mathbb{Q}^n$ mit $Ax \leq b$?

GANZZAHLIGE LINEARE UNGLEICHUNGEN

Instanz: Eine Matrix $A \in \mathbb{Z}^{m \times n}$ und ein Vektor $b \in \mathbb{Z}^m$.

Frage: Gibt es einen Vektor $x \in \mathbb{Z}^n$ mit $Ax \leq b$?

Definition 15.8. Die Klasse aller Entscheidungsprobleme, für die es einen polynomiellen Algorithmus gibt, wird mit P bezeichnet.

Mit anderen Worten: In P liegen die Paare (X, Y) mit $Y \subseteq X \subseteq \{0, 1\}^*$, wobei X und Y zwei in polynomieller Zeit entscheidbare Sprachen sind. Um zu beweisen,

dass ein Problem in P ist, gibt man gewöhnlich einen polynomiellen Algorithmus an. Nach den Resultaten in Abschnitt 15.2 gibt es eine polynomielle Turingmaschine für jedes Problem in P . Nach dem Satz von Khachiyan (Satz 4.18) folgt, dass LINEARE UNGLEICHUNGEN in P ist. Ob GANZZAHLIGE LINEARE UNGLEICHUNGEN oder HAMILTON-KREIS in P sind, ist unbekannt. Wir werden nun eine weitere Klasse einführen, nämlich NP , die sowohl diese beiden Probleme, als auch in der Tat die meisten der in diesem Buch besprochenen Entscheidungsprobleme enthält.

Wir werden nicht auf einem polynomiellen Algorithmus bestehen, fordern aber für jede Ja-Instanz ein in polynomieller Zeit prüftbares Zertifikat. Für das HAMILTON-KREIS-Problem beispielsweise ist ein mögliches Zertifikat einfach ein Hamilton-Kreis. Man prüft leicht, ob ein gegebener String die binäre Kodierung eines Hamilton-Kreises ist. Beachte, dass wir kein Zertifikat für Nein-Instanzen fordern. Wir definieren nun formal:

Definition 15.9. Ein Entscheidungsproblem $\mathcal{P} = (X, Y)$ ist in NP , falls es ein Polynom p und ein Entscheidungsproblem $\mathcal{P}' = (X', Y')$ in P gibt, wobei

$$X' := \left\{ x\#c : x \in X, c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor} \right\},$$

so dass

$$Y = \left\{ y \in X : \text{es gibt einen String } c \in \{0, 1\}^{\lfloor p(\text{size}(y)) \rfloor} \text{ mit } y\#c \in Y' \right\}.$$

Hier bezeichnet $x\#c$ die Verkettung des Strings x , des Symbols $\#$ und des Strings c . Ein String c mit $y\#c \in Y'$ heißt ein **Zertifikat** für y (da c beweist, dass $y \in Y$). Ein Algorithmus für \mathcal{P}' heißt ein **Zertifikat-Prüfalgorithmus**.

Proposition 15.10. $P \subseteq NP$.

Beweis: Man kann p identisch Null nehmen. Ein Algorithmus für \mathcal{P}' entfernt einfach das letzte Symbol des Inputs „ $x\#$ “ und wendet dann einen Algorithmus für \mathcal{P} an. \square

Ob $P = NP$ gilt, ist unbekannt. In der Tat ist dies das wichtigste offene Problem in der Komplexitätstheorie. Als Beispiel eines Problems in NP , von dem man nicht weiß, ob es in P ist, haben wir das folgende:

Proposition 15.11. HAMILTON-KREIS ist in NP .

Beweis: Für jede Ja-Instanz G nehmen wir irgendeinen Hamilton-Kreis in G als Zertifikat. Es ist klar, dass man in polynomieller Zeit prüfen kann, ob eine gegebene Kantenmenge tatsächlich einen Hamilton-Kreis in einem gegebenen Graphen bildet. \square

Proposition 15.12. GANZZAHLIGE LINEARE UNGLEICHUNGEN ist in NP .

Beweis: Als Zertifikat nehmen wir hier einfach einen Lösungsvektor. Gibt es eine Lösung, so gibt es nach Korollar 5.8 auch eine mit polynomieller Größe. \square

Die Abkürzung *NP* bedeutet „nichtdeterministisch polynomiell“. Um dies zu erklären, müssen wir den Begriff des nichtdeterministischen Algorithmus definieren. Das bietet eine gute Gelegenheit, allgemeine randomisierte Algorithmen zu definieren, einen Begriff, den wir schon früher angetroffen haben. Alle randomisierten Algorithmen haben eines gemeinsam: Ihre Berechnungen hängen nicht nur vom Input ab, sondern auch von einigen Zufalls-Bits.

Definition 15.13. Ein randomisierter Algorithmus zur Berechnung einer Funktion $f : S \rightarrow T$ ist ein Algorithmus, der eine Funktion $g : \{s\#r : s \in S, r \in \{0, 1\}^{k(s)}\} \rightarrow T$ berechnet, wobei $k : S \rightarrow \mathbb{Z}_+$. Für jede Instanz $s \in S$ kann der Algorithmus somit $k(s) \in \mathbb{Z}_+$ Zufalls-Bits benutzen. Wir messen nur die Abhängigkeit der Laufzeit von $\text{size}(s)$: polynomielle randomisierte Algorithmen können also nur eine polynomielle Anzahl von Zufalls-Bits lesen.

Natürlich interessieren uns solche randomisierten Algorithmen nur, wenn f und g funktionell verwandt sind. Der Idealfall ist gegeben durch $g(s\#r) = f(s)$ für alle $s \in S$ und $r \in \{0, 1\}^{k(s)}$; hier sprechen wir von einem **Las-Vegas-Algorithmus**. Ein Las-Vegas-Algorithmus berechnet immer das korrekte Ergebnis, aber die Laufzeit für verschiedene Läufe auf demselben Input s kann variieren. Gelegentlich sind auch weniger zuverlässige Algorithmen interessant: Besteht wenigstens eine instanz-unabhängige positive Wahrscheinlichkeit p für ein korrektes Ergebnis, d. h.

$$p := \inf_{s \in S} \frac{|\{r \in \{0, 1\}^{k(s)} : g(s\#r) = f(s)\}|}{2^{k(s)}} > 0,$$

so sprechen wir von einem **Monte-Carlo-Algorithmus**.

Sei $T = \{0, 1\}$. Gilt für jedes $s \in S$ mit $f(s) = 0$, dass $g(s\#r) = 0$ für alle $r \in \{0, 1\}^{k(s)}$, so sprechen wir von einem randomisierten Algorithmus mit **einseitigem Fehler**. Gilt auch noch für jedes $s \in S$ mit $f(s) = 1$, dass es mindestens ein $r \in \{0, 1\}^{k(s)}$ mit $g(s\#r) = 1$ gibt, so sprechen wir von einem **nichtdeterministischen Algorithmus**.

Ein randomisierter Algorithmus kann auch als ein Orakel-Algorithmus aufgefasst werden, für den das Orakel bei jedem Aufruf ein Zufalls-Bit (0 oder 1) liefert. Ein nichtdeterministischer Algorithmus für ein Entscheidungsproblem beantwortet jede Nein-Instanz mit „nein“, und bei jeder Ja-Instanz besteht die Möglichkeit, dass er mit „ja“ antwortet. Das folgende Resultat folgt leicht:

Proposition 15.14. Ein Entscheidungsproblem ist genau dann in *NP*, wenn es einen polynomiellen nichtdeterministischen Algorithmus hat.

Beweis: Sei $\mathcal{P} = (X, Y)$ ein Entscheidungsproblem in *NP*. Sei ferner $\mathcal{P}' = (X', Y')$ wie in Definition 15.9. Dann ist ein polynomieller Algorithmus für \mathcal{P}' in der Tat auch ein nichtdeterministischer Algorithmus für \mathcal{P} : Das unbekannte Zertifikat ersetzt man einfach durch Zufalls-Bits. Da die Anzahl der Zufalls-Bits

durch ein Polynom in $\text{size}(x)$, $x \in X$, beschränkt wird, gilt dies auch für die Laufzeit des Algorithmus.

Umgekehrt: Hat $\mathcal{P} = (X, Y)$ einen nichtdeterministischen polynomiellen Algorithmus, der $k(x)$ Zufalls-Bits für die Instanz x benutzt, so gibt es ein Polynom p mit $k(x) \leq p(\text{size}(x))$ für jede Instanz x . Nun setzen wir

$$X' := \left\{ x\#c : x \in X, c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor} \right\} \text{ und}$$

$$Y' := \left\{ x\#c \in X' : g(x\#r) = 1, r \text{ besteht aus den ersten } k(x) \text{ Bits von } c \right\}.$$

Dann folgt mit der Definition von nichtdeterministischen Algorithmen, dass $(X', Y') \in P$ und

$$Y = \left\{ y \in X : \text{es gibt einen String } c \in \{0, 1\}^{\lfloor p(\text{size}(y)) \rfloor} \text{ mit } y\#c \in Y' \right\}.$$

□

Die meisten in der kombinatorischen Optimierung auftretenden Entscheidungsprobleme sind in NP . Bei vielen dieser Probleme weiß man nicht, ob es für sie einen polynomiellen Algorithmus gibt. Man kann jedoch sagen, dass manche nicht leichter als andere sind. Um diese Idee zu präzisieren, führen wir den wichtigen Begriff der polynomiellen Reduktion ein, der für allgemeine Berechnungsprobleme gilt.

Definition 15.15. Seien \mathcal{P}_1 und \mathcal{P}_2 zwei Berechnungsprobleme. Wir sagen, dass sich \mathcal{P}_1 auf \mathcal{P}_2 **polynomiell reduziert**, falls es einen \mathcal{P}_2 benutzenden polynomiellen Orakel-Algorithmus für \mathcal{P}_1 gibt.

Das folgende Resultat ist der Hauptgrund für diesen Begriff:

Proposition 15.16. Angenommen, \mathcal{P}_1 reduziert sich polynomiell auf \mathcal{P}_2 und es gibt einen polynomiellen Algorithmus für \mathcal{P}_2 . Dann gibt es auch einen polynomiellen Algorithmus für \mathcal{P}_1 .

Beweis: Sei A_2 ein Algorithmus für \mathcal{P}_2 mit $\text{time}(A_2, y) \leq p_2(\text{size}(y))$ für alle Instanzen y von \mathcal{P}_2 . Sei A_1 ein \mathcal{P}_2 benutzender Orakel-Algorithmus für \mathcal{P}_1 mit $\text{time}(A_1, x) \leq p_1(\text{size}(x))$ für alle Instanzen x von \mathcal{P}_1 . Dann ergibt das Ersetzen der Orakelaufufe in A_1 durch Subroutinen, die mit A_2 äquivalent sind, einen Algorithmus A_3 für \mathcal{P}_1 . Für jede Instanz x von \mathcal{P}_1 mit $\text{size}(x) = n$ gilt $\text{time}(A_3, x) \leq p_1(n) \cdot p_2(p_1(n))$: Es gibt höchstens $p_1(n)$ Orakelaufufe in A_1 , und keine der durch A_1 produzierten Instanzen von \mathcal{P}_2 kann länger als $p_1(n)$ sein. Da wir für p_1 und p_2 Polynome wählen können, folgt, dass A_3 ein polynomieller Algorithmus ist. □

Die Theorie der NP -Vollständigkeit basiert auf einer besonderen Variante der polynomiellen Reduktion, die nur für Entscheidungsprobleme definiert wird:

Definition 15.17. Seien $\mathcal{P}_1 = (X_1, Y_1)$ und $\mathcal{P}_2 = (X_2, Y_2)$ Entscheidungsprobleme. Wir sagen, dass sich \mathcal{P}_1 in \mathcal{P}_2 **polynomiell transformiert**, falls es eine in polynomieller Zeit berechenbare Funktion $f : X_1 \rightarrow X_2$ gibt, so dass $f(x_1) \in Y_2$ für alle $x_1 \in Y_1$ und $f(x_1) \in X_2 \setminus Y_2$ für alle $x_1 \in X_1 \setminus Y_1$.

Mit anderen Worten, Ja-Instanzen werden in Ja-Instanzen transformiert und Nein-Instanzen in Nein-Instanzen. Offensichtlich gilt: Transformiert sich ein Problem \mathcal{P}_1 polynomiell in das Problem \mathcal{P}_2 , so reduziert sich \mathcal{P}_1 auch polynomiell auf \mathcal{P}_2 . Polynomielle Transformationen werden auch Karp-Reduktionen genannt, während allgemeine polynomielle Reduktionen auch Turing-Reduktionen genannt werden. Man sieht leicht, dass beide Operationen transitiv sind. (für polynomielle Reduktionen lässt sich dies wie im Beweis von Proposition 15.16 zeigen).

Definition 15.18. Ein Entscheidungsproblem $\mathcal{P} \in NP$ heißt **NP-vollständig**, falls sich alle anderen Probleme in NP polynomiell in \mathcal{P} transformieren.

Nach Proposition 15.16 wissen wir: Gibt es einen polynomiellen Algorithmus für irgendein NP-vollständiges Problem, so folgt $P = NP$.

Gäbe es gar keine NP-vollständigen Probleme, so wäre die obige Definition natürlich bedeutungslos. Das Ziel des nächsten Abschnitts ist es, zu beweisen, dass es tatsächlich ein NP-vollständiges Problem gibt.

15.4 Der Satz von Cook

In seiner bahnbrechenden Arbeit hat Cook [1971] bewiesen, dass ein bestimmtes Entscheidungsproblem, nämlich SATISFIABILITY, in der Tat NP-vollständig ist. Wir benötigen einige Definitionen:

Definition 15.19. Sei X eine endliche Menge boolescher Variablen. Eine **Wahrheitsbelegung** für X ist eine Funktion $T : X \rightarrow \{\text{true}, \text{false}\}$. Wir erweitern T folgendermaßen auf die Menge $L := X \dot{\cup} \{\bar{x} : x \in X\}$: Setze $T(\bar{x}) := \text{true}$, falls $T(x) = \text{false}$, und umgekehrt (\bar{x} kann als die Negation von x betrachtet werden). Die Elemente von L heißen **Literale über X** .

Eine **Klausel über X** ist eine Menge von Literalen über X . Eine Klausel repräsentiert die Disjunktion dieser Literalen und wird von einer Wahrheitsbelegung **erfüllt**, wenn mindestens eines ihrer Literalen true ist. Eine Familie von Klauseln über X ist **erfüllbar**, wenn es eine Wahrheitsbelegung gibt, die alle Klauseln gleichzeitig erfüllt.

Da wir die Konjunktion von Disjunktionen von Literalen betrachten, sprechen wir auch von booleschen Formeln (siehe Aufgabe 21) in konjunktiver Normalform. Der Familie $\{\{x_1, \bar{x_2}\}, \{\bar{x_2}, \bar{x_3}\}, \{x_1, x_2, \bar{x_3}\}, \{\bar{x_1}, x_3\}\}$ entspricht z. B. die boolesche Formel $(x_1 \vee \bar{x_2}) \wedge (\bar{x_2} \vee \bar{x_3}) \wedge (x_1 \vee x_2 \vee \bar{x_3}) \wedge (\bar{x_1} \vee x_3)$. Sie ist erfüllbar, wie die Wahrheitsbelegung $T(x_1) := \text{true}$, $T(x_2) := \text{false}$ und $T(x_3) := \text{true}$ zeigt. Wir können nun das SATISFIABILITY Problem angeben:

SATISFIABILITY

Instanz: Eine Menge X von Variablen und eine Familie \mathcal{Z} von Klauseln über X .

Frage: Ist \mathcal{Z} erfüllbar?

Satz 15.20. (Cook [1971]) SATISFIABILITY ist NP-vollständig.

Beweis: SATISFIABILITY ist in NP, weil eine erfüllende Wahrheitsbelegung als Zertifikat für jede Ja-Instanz genügt, und dies kann natürlich in polynomieller Zeit geprüft werden.

Sei nun $\mathcal{P} = (X, Y)$ ein beliebiges weiteres Problem in NP. Wir müssen zeigen, dass sich \mathcal{P} polynomiell in SATISFIABILITY transformiert.

Nach Definition 15.9 gibt es ein Polynom p und ein Entscheidungsproblem $\mathcal{P}' = (X', Y')$ in P , wobei $X' := \{x\#c : x \in X, c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor}\}$ und

$$Y = \left\{ y \in X : \text{es gibt einen String } c \in \{0, 1\}^{\lfloor p(\text{size}(y)) \rfloor} \text{ mit } y\#c \in Y' \right\}.$$

Sei

$$\Phi : \{0, \dots, N\} \times \bar{A} \rightarrow \{-1, \dots, N\} \times \bar{A} \times \{-1, 0, 1\}$$

eine polynomielle Turingmaschine für \mathcal{P}' mit Alphabet A , und setze $\bar{A} := A \cup \{\sqcup\}$. Sei q ein Polynom mit $\text{time}(\Phi, x\#c) \leq q(\text{size}(x\#c))$ für alle Instanzen $x\#c \in X'$. Beachte, dass $\text{size}(x\#c) = \text{size}(x) + 1 + \lfloor p(\text{size}(x)) \rfloor$.

Wir werden nun für jedes $x \in X$ eine Familie $\mathcal{Z}(x)$ von Klauseln über einer Menge $V(x)$ boolescher Variablen konstruieren, so dass $\mathcal{Z}(x)$ genau dann erfüllbar ist, wenn $x \in Y$.

Setze $Q := q(\text{size}(x) + 1 + \lfloor p(\text{size}(x)) \rfloor)$. Es ist Q eine obere Schranke für die Längen der Berechnungen von Φ für den Input $x\#c$ für alle $c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor}$. Es enthält $V(x)$ die folgenden booleschen Variablen:

- eine Variable $v_{ij\sigma}$ für alle $0 \leq i \leq Q, -Q \leq j \leq Q$ und $\sigma \in \bar{A}$;
- eine Variable w_{ijn} für alle $0 \leq i \leq Q, -Q \leq j \leq Q$ und $-1 \leq n \leq N$.

Diese Variablen sollen die folgende Bedeutung haben: $v_{ij\sigma}$ gibt an, ob zum Zeitpunkt i (d. h. nach i Schritten der Berechnung) die j -te Stelle des Strings das Symbol σ enthält. Die Variable w_{ijn} gibt an, ob zum Zeitpunkt i die j -te Stelle des Strings gescannt und der n -te Befehl ausgeführt werden wird.

Ist also $(n^{(i)}, s^{(i)}, \pi^{(i)})_{i=0,1,\dots}$ eine Berechnung von Φ , so wollen wir $v_{ij\sigma}$ genau dann auf true setzen, wenn $s_j^{(i)} = \sigma$, und w_{ijn} genau dann auf true, wenn $\pi^{(i)} = j$ und $n^{(i)} = n$.

Die Familie $\mathcal{Z}(x)$ von zu konstruierenden Klauseln wird genau dann erfüllbar sein, wenn es einen String c mit $\text{output}(\Phi, x\#c) = 1$ gibt.

Es enthält $\mathcal{Z}(x)$ die folgenden Klauseln zur Modellierung der folgenden Bedingungen:

Zu jedem Zeitpunkt enthält jede Stelle des Strings ein eindeutig bestimmtes Symbol:

- $\{v_{ij\sigma} : \sigma \in \bar{A}\}$ für $0 \leq i \leq Q$ und $-Q \leq j \leq Q$;
- $\{\overline{v_{ij\sigma}}, \overline{v_{ij\tau}}\}$ für $0 \leq i \leq Q$, $-Q \leq j \leq Q$ und $\sigma, \tau \in \bar{A}$ mit $\sigma \neq \tau$.

Zu jedem Zeitpunkt wird eine eindeutig bestimmte Stelle des Strings gescannt und ein einziger Befehl ausgeführt:

- $\{w_{ijn} : -Q \leq j \leq Q, -1 \leq n \leq N\}$ für $0 \leq i \leq Q$;
- $\{\overline{w_{ijn}}, \overline{w_{ij'n'}}\}$ für $0 \leq i \leq Q$, $-Q \leq j \leq Q$, $j' \leq Q$ und $-1 \leq n, n' \leq N$ mit $(j, n) \neq (j', n')$.

Der Algorithmus beginnt korrekt mit Input $x\#c$ für ein $c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor}$:

- $\{v_{0,j,x_j}\}$ für $1 \leq j \leq \text{size}(x)$;
- $\{v_{0,\text{size}(x)+1,\#}\}$;
- $\{v_{0,\text{size}(x)+1+j,0}, v_{0,\text{size}(x)+1+j,1}\}$ für $1 \leq j \leq \lfloor p(\text{size}(x)) \rfloor$;
- $\{v_{0,j,\sqcup}\}$ für $-Q \leq j \leq 0$ und $\text{size}(x) + 2 + \lfloor p(\text{size}(x)) \rfloor \leq j \leq Q$;
- $\{w_{010}\}$.

Der Algorithmus arbeitet korrekt:

- $\{\overline{v_{ij\sigma}}, \overline{w_{ijn}}, v_{i+1,j,\tau}\}, \{\overline{v_{ij\sigma}}, \overline{w_{ijn}}, w_{i+1,j+\delta,m}\}$ für $0 \leq i < Q$, $-Q \leq j \leq Q$, $\sigma \in \bar{A}$ und $0 \leq n \leq N$, wobei $\Phi(n, \sigma) = (m, \tau, \delta)$.

Erreicht der Algorithmus den Befehl -1 , so terminiert er:

- $\{\overline{w_{i,j,-1}}, w_{i+1,j,-1}\}, \{\overline{w_{i,j,-1}}, \overline{v_{i,j,\sigma}}, v_{i+1,j,\sigma}\}$ für $0 \leq i < Q$, $-Q \leq j \leq Q$ und $\sigma \in \bar{A}$.

Ungescannte Stellen bleiben unverändert:

- $\{\overline{v_{ij\sigma}}, \overline{w_{ij'n}}, v_{i+1,j,\sigma}\}$ für $0 \leq i < Q$, $\sigma \in \bar{A}$, $-1 \leq n \leq N$ und $-Q \leq j, j' \leq Q$ mit $j \neq j'$.

Der Output des Algorithmus ist 1:

- $\{v_{Q,1,1}\}, \{v_{Q,2,\sqcup}\}$.

Die Kodierungslänge von $\mathcal{Z}(x)$ ist $O(Q^3 \log Q)$: Literale kommen $O(Q^3)$ mal vor, und deren Indizes benötigen $O(\log Q)$ -Speicherraum. Da Q polynomiell von $\text{size}(x)$ abhängt, gibt es einen polynomiellen Algorithmus, der $\mathcal{Z}(x)$ für gegebenes x konstruiert. Beachte, dass p , Φ und q fest sind und nicht Teil des Inputs dieses Algorithmus sind.

Es bleibt zu zeigen, dass $\mathcal{Z}(x)$ genau dann erfüllbar ist, wenn $x \in Y$.

Sei $\mathcal{Z}(x)$ erfüllbar. Dann betrachten wir eine Wahrheitsbelegung T , die alle Klauseln erfüllt. Sei $c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor}$ mit $c_j = 1$ für alle j mit $T(v_{0,\text{size}(x)+1+j,1}) = \text{true}$, anderenfalls sei $c_j = 0$. Nach obiger Konstruktion folgt: Die Variablen geben die Berechnungen von Φ für den Input $x\#c$ an. Daraus folgern wir, dass $\text{output}(\Phi, x\#c) = 1$. Da Φ ein Zertifikat-Prüfalgorithmus ist, folgt, dass x eine Ja-Instanz ist.

Zur Umkehrung: Sei $x \in Y$ und c irgendein Zertifikat für x . Sei ferner $(n^{(i)}, s^{(i)}, \pi^{(i)})_{i=0,1,\dots,m}$ die Berechnung von Φ für den Input $x\#c$. Dann definieren wir $T(v_{i,j,\sigma}) := \text{true}$ genau dann, wenn $s_j^{(i)} = \sigma$, und $T(w_{i,j,n}) = \text{true}$

genau dann, wenn $\pi^{(i)} = j$ und $n^{(i)} = n$. Für $i := m + 1, \dots, Q$ setzen wir $T(v_{i,j,\sigma}) := T(v_{i-1,j,\sigma})$ und $T(w_{i,j,n}) := T(w_{i-1,j,n})$ für alle j, n und σ . Dann folgt, dass T eine $\mathcal{Z}(x)$ erfüllende Wahrheitsbelegung ist, womit der Beweis abgeschlossen ist. \square

SATISFIABILITY ist keineswegs das einzige NP -vollständige Problem; wir werden viele weitere in diesem Buch kennenlernen. Mit der Kenntnis eines NP -vollständigen Problems ist es nun viel einfacher, die NP -Vollständigkeit weiterer Probleme zu beweisen. Um zu zeigen, dass ein gewisses Entscheidungsproblem \mathcal{P} NP -vollständig ist, brauchen wir bloß zu zeigen, dass $\mathcal{P} \in NP$ und dass sich SATISFIABILITY (oder irgendein anderes bekanntlich NP -vollständiges Problem) polynomiell in \mathcal{P} transformiert. Das genügt, da polynomielle Transformierbarkeit transitiv ist.

Die folgende eingeschränkte Version von SATISFIABILITY wird sich bei einigen NP -Vollständigkeitsbeweisen als sehr nützlich erweisen:

3SAT

Instanz: Eine Menge X von Variablen und eine Familie \mathcal{Z} von Klauseln über

X , wobei jede dieser Klauseln genau drei Literale enthält.

Frage: Ist \mathcal{Z} erfüllbar?

Um zu zeigen, dass 3SAT NP -vollständig ist, bemerken wir, dass jede Klausel äquivalent durch eine Menge von 3SAT-Klauseln ersetzt werden kann:

Proposition 15.21. Sei X eine Menge von Variablen und Z eine Klausel über X mit k Literalen. Dann gibt es eine Menge Y von höchstens $\max\{k - 3, 2\}$ neuen Variablen und eine Familie \mathcal{Z}' von höchstens $\max\{k - 2, 4\}$ Klauseln über $X \cup Y$, so dass jedes Element von \mathcal{Z}' genau drei Literale enthält und für jede Familie \mathcal{W} von Klauseln über X gilt: $\mathcal{W} \cup \{Z\}$ ist genau dann erfüllbar, wenn $\mathcal{W} \cup \mathcal{Z}'$ erfüllbar ist. Ferner gilt: Eine solche Familie \mathcal{Z}' kann in $O(k)$ -Zeit berechnet werden.

Beweis: Hat Z drei Literale, so setzen wir $\mathcal{Z}' := \{Z\}$. Hat Z mehr als drei Literale, ist etwa $Z = \{\lambda_1, \dots, \lambda_k\}$ mit $k > 3$, so wählen wir eine Menge $Y = \{y_1, \dots, y_{k-3}\}$ von $k - 3$ neuen Variablen und setzen

$$\begin{aligned} \mathcal{Z}' := & \{\{\lambda_1, \lambda_2, y_1\}, \{\overline{y_1}, \lambda_3, y_2\}, \{\overline{y_2}, \lambda_4, y_3\}, \dots, \\ & \{\overline{y_{k-4}}, \lambda_{k-2}, y_{k-3}\}, \{\overline{y_{k-3}}, \lambda_{k-1}, \lambda_k\}\}. \end{aligned}$$

Ist $Z = \{\lambda_1, \lambda_2\}$, so wählen wir eine neue Variable y_1 ($Y := \{y_1\}$) und setzen

$$\mathcal{Z}' := \{\{\lambda_1, \lambda_2, y_1\}, \{\lambda_1, \lambda_2, \overline{y_1}\}\}.$$

Ist $Z = \{\lambda_1\}$, so wählen wir eine Menge $Y = \{y_1, y_2\}$ von zwei neuen Variablen und setzen

$$\mathcal{Z}' := \{\{\lambda_1, y_1, y_2\}, \{\lambda_1, y_1, \overline{y_2}\}, \{\lambda_1, \overline{y_1}, y_2\}, \{\lambda_1, \overline{y_1}, \overline{y_2}\}\}.$$

Beachte: In jeder Instanz von SATISFIABILITY kann Z in allen obigen Fällen äquivalent durch die Klauseln in \mathcal{Z}' ersetzt werden. \square

Satz 15.22. (Cook [1971]) 3SAT ist NP-vollständig.

Beweis: Als eingeschränkte Version von SATISFIABILITY ist 3SAT offensichtlich in NP. Wir zeigen nun, dass sich SATISFIABILITY polynomiell in 3SAT transformiert. Betrachte irgendeine Familie \mathcal{Z} von Klauseln Z_1, \dots, Z_m . Wir werden eine neue Familie \mathcal{Z}' von Klauseln mit drei Literalen pro Klausel konstruieren, so dass \mathcal{Z} genau dann erfüllbar ist, wenn \mathcal{Z}' erfüllbar ist.

Dies erreichen wir, indem wir jede Klausel Z_i durch eine äquivalente Menge von Klauseln ersetzen, jede mit genau drei Literalen. Dies ist nach Proposition 15.21 in linearer Zeit möglich. \square

Beschränken wir jede Klausel auf genau zwei Literale, so heißt das Problem 2SAT; es kann in linearer Zeit gelöst werden (Aufgabe 8).

15.5 Einige grundlegende NP-vollständige Probleme

Karp [1972] entdeckte, welch weitreichende Folgerungen für die kombinatorische Optimierung aus den Arbeiten von Cook resultieren. Zunächst betrachten wir das folgende Problem:

STABLE-SET

Instanz: Ein Graph G und eine ganze Zahl k .

Frage: Gibt es eine stabile Menge mit k Knoten?

Satz 15.23. (Karp [1972]) STABLE-SET ist NP-vollständig.

Beweis: Offensichtlich ist STABLE-SET in NP. Wir werden zeigen, dass sich SATISFIABILITY polynomiell in STABLE SET transformiert.

Sei \mathcal{Z} eine Familie von Klauseln Z_1, \dots, Z_m mit $Z_i = \{\lambda_{i1}, \dots, \lambda_{ik_i}\}$ ($i = 1, \dots, m$), wobei die λ_{ij} Literale über einer Menge X von Variablen sind.

Wir werden einen Graphen G konstruieren, mit der Eigenschaft, dass G genau dann eine stabile Menge der Größe m besitzt, wenn es eine alle m Klauseln erfüllende Wahrheitsbelegung gibt.

Für jede Klausel Z_i führen wir eine Clique mit k_i Knoten ein, entsprechend den Literalen dieser Klausel. Knoten, die verschiedenen Klauseln entsprechen, werden genau dann mit einer Kante verbunden, wenn die Literale einander widersprechen. Formal bedeutet dies: Sei $V(G) := \{v_{ij} : 1 \leq i \leq m, 1 \leq j \leq k_i\}$ und

$$\begin{aligned} E(G) := \{&\{v_{ij}, v_{kl}\} : (i = k \text{ und } j \neq l) \\ &\text{oder } (\lambda_{ij} = x \text{ und } \lambda_{kl} = \bar{x} \text{ für ein } x \in X)\}. \end{aligned}$$

Abbildung 15.1 zeigt ein Beispiel: $m = 4$, $Z_1 = \{\bar{x}_1, x_2, x_3\}$, $Z_2 = \{x_1, \bar{x}_3\}$, $Z_3 = \{x_2, \bar{x}_3\}$ und $Z_4 = \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}$.

Angenommen, G besitzt eine stabile Menge der Größe m . Dann ergeben ihre Knoten paarweise kompatible Literale aus verschiedenen Klauseln. Setzen wir all

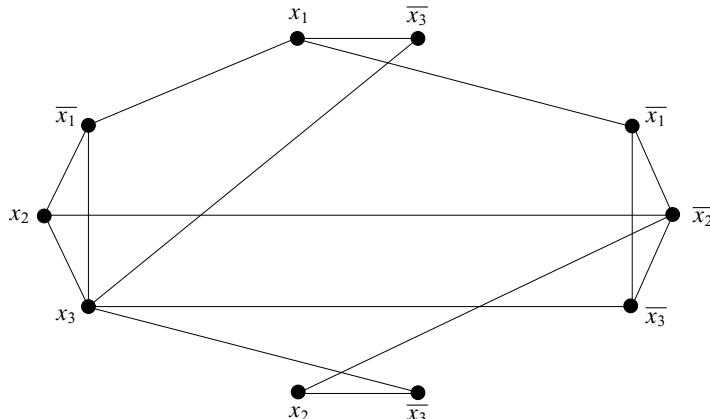


Abbildung 15.1.

diese Literale auf *true* (und hier nicht vorkommende Variablen beliebig), so erhalten wir eine alle m Klauseln erfüllende Wahrheitsbelegung.

Zur Umkehrung: Gegeben sei eine alle m Klauseln erfüllende Wahrheitsbelegung. Wir wählen aus jeder Klausel ein Literal, welches *true* ist. Die entsprechende Knotenmenge definiert dann eine stabile Menge der Größe m in G . \square

Es ist notwendig, dass k Teil des Inputs ist: Für jedes feste k kann man in $O(n^k)$ -Zeit entscheiden, ob ein gegebener Graph mit n Knoten eine stabile Menge der Größe k besitzt (indem man einfach alle k -elementigen Knotenmengen prüft). Es folgen zwei interessante verwandte Probleme:

VERTEX-COVER

Instanz: Ein Graph G und eine ganze Zahl k .

Frage: Gibt es eine Knotenüberdeckung der Kardinalität k ?

CLIQUE

Instanz: Ein Graph G und eine ganze Zahl k .

Frage: Besitzt G eine Clique der Kardinalität k ?

Korollar 15.24. (Karp [1972]) VERTEX-COVER und CLIQUE sind NP-vollständig.

Beweis: Nach Proposition 2.2 transformiert sich STABLE-SET polynomiell sowohl in VERTEX-COVER als auch in CLIQUE. \square

Wir wenden uns nun wieder dem berühmten (bereits in Abschnitt 15.3 definierten) Hamilton-Kreis-Problem zu.

Satz 15.25. (Karp [1972]) HAMILTON-KREIS ist NP-vollständig.

Beweis: Offensichtlich ist HAMILTON-KREIS in NP . Wir werden beweisen, dass sich 3SAT polynomiell in HAMILTON-KREIS transformiert. Gegeben sei eine Familie \mathcal{Z} von Klauseln Z_1, \dots, Z_m über $X = \{x_1, \dots, x_n\}$, wobei jede dieser Klauseln genau drei Literale enthält. Wir werden einen Graphen G konstruieren, mit der Eigenschaft, dass G genau dann hamiltonsch ist, wenn \mathcal{Z} erfüllbar ist.

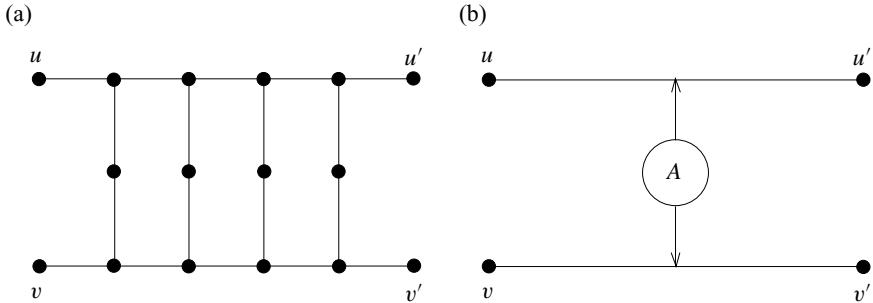


Abbildung 15.2.

Zunächst definieren wir zwei Hilfskonstruktionen, die mehrfach in G erscheinen werden. Betrachte den in Abb. 15.2(a) dargestellten Graphen, den wir mit A bezeichnen. Wir nehmen an, dass A ein Teilgraph von G ist und dass außer den Knoten u, u', v, v' kein Knoten von A mit einer anderen Kante von G inzident ist. Dann folgt: Jeder Hamilton-Kreis in G durchquert A auf eine der beiden in Abb. 15.3(a) und (b) dargestellten Weisen. Somit können wir A durch zwei Kanten ersetzen, unter der zusätzlichen Bedingung, dass jeder Hamilton-Kreis in G genau eine dieser beiden Kanten enthalten muss (Abb. 15.2(b)).

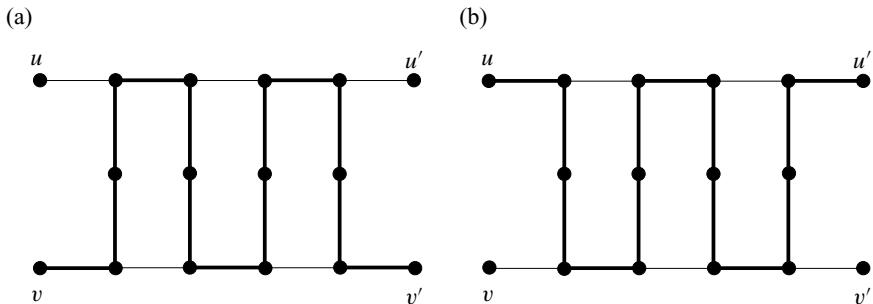


Abbildung 15.3.

Betrachte nun den in Abb. 15.4(a) dargestellten Graphen B . Wir nehmen an, dass B ein Teilgraph von G ist und dass außer den Knoten u und u' kein Knoten von B mit einer anderen Kante von G inzident ist. Dann durchläuft kein Hamilton-Kreis in G alle drei Kanten e_1, e_2, e_3 . Auch prüft man leicht, dass es für jedes

$S \subset \{e_1, e_2, e_3\}$ einen hamiltonschen Weg von u nach u' in B gibt, der S enthält, aber keine der Kanten aus $\{e_1, e_2, e_3\} \setminus S$. Wir stellen B wie in Abb. 15.4(b) dar.

Wir können nun den Graphen G konstruieren. Für jede Klausel nehmen wir eine Kopie von B und diese werden alle schrittweise verbunden. Zwischen der ersten und letzten Kopie von B fügen wir zwei Knoten für jede Variable ein und verbinden sie alle schrittweise. Dann verdoppeln wir noch die Kanten zwischen den Knotenpaaren für jede Variable x ; diese Kantenpaare entsprechen x bzw. \bar{x} .

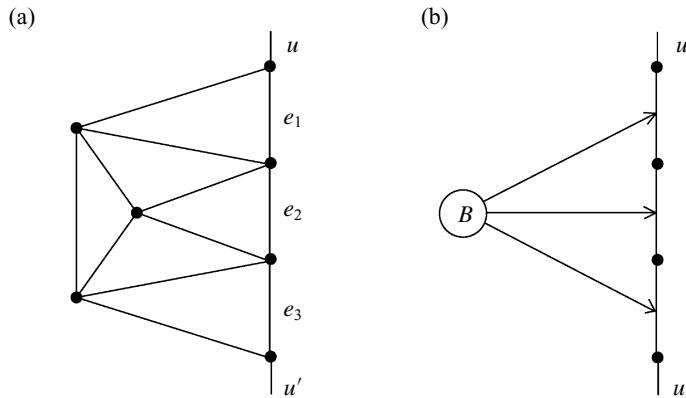


Abbildung 15.4.

In jeder Kopie von B verbinden wir nun die Kanten e_1 , e_2 und e_3 mittels einer Kopie von A mit denjenigen Kanten, die dem ersten, zweiten, bzw. dritten Literal der entsprechenden Klausel entsprechen. Diese Konstruktionen erfolgen schrittweise: Bei der Hinzunahme einer Kopie des Teilgraphen A an der Stelle einer Kante $e = \{u, v\}$, die einem Literal entspricht, spielt die mit u inzidente Kante in Abb. 15.2(a) die Rolle von e : Sie ist jetzt selbst die diesem Literal entsprechende Kante. Die Gesamtkonstruktion ist in Abb. 15.5 für das Beispiel $\{\{x_1, \bar{x}_2, \bar{x}_3\}, \{\bar{x}_1, x_2, \bar{x}_3\}, \{\bar{x}_1, \bar{x}_2, x_3\}\}$ abgebildet.

Nun behaupten wir, dass G genau dann hamiltonsch ist, wenn \mathcal{Z} erfüllbar ist. Sei C ein Hamilton-Kreis. Wir definieren eine Wahrheitsbelegung, indem wir ein Literal genau dann auf *true* setzen, wenn C die entsprechende Kante enthält. Mit den Eigenschaften der Hilfskonstruktionen A und B folgt dann, dass jede Klausel ein auf *true* gesetztes Literal enthält.

Zur Umkehrung: Jede erfüllende Wahrheitsbelegung definiert eine Kantenmenge, die den auf *true* gesetzten Literalen entspricht. Da jede Klausel ein auf *true* gesetztes Literal enthält, kann man diese Kantenmenge zu einer Tour in G erweitern. \square

Dieser Beweis stammt im Wesentlichen von Papadimitriou und Steiglitz [1982]. Das Entscheidungsproblem, ob ein gegebener Graph einen hamiltonschen Weg

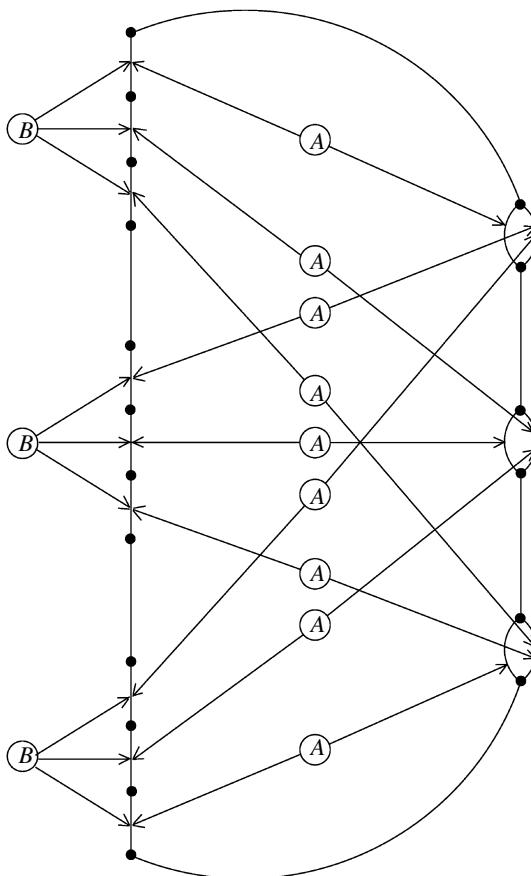


Abbildung 15.5.

enthält, ist auch *NP*-vollständig (Aufgabe 17(a)). Ferner kann man leicht die ungerichteten Versionen in das gerichtete Hamilton-Kreis-Problem oder das gerichtete Hamiltonscher-Weg-Problem umformen, indem man jede ungerichtete Kante durch zwei entgegengesetzt gerichtete Kanten ersetzt. Somit sind die gerichteten Versionen auch *NP*-vollständig.

Es gibt ein weiteres grundlegendes *NP*-vollständiges Problem:

3-DIMENSIONALES MATCHING (3DM)

Instanz: Drei paarweise disjunkte Mengen U, V, W gleicher Kardinalität und $T \subseteq U \times V \times W$.

Frage: Gibt es eine Teilmenge M von T mit $|M| = |U|$ und der Eigenschaft:
Für verschiedene $(u, v, w), (u', v', w') \in M$ gilt $u \neq u'$, $v \neq v'$ und $w \neq w'$?

Satz 15.26. (Karp [1972]) 3DM ist NP-vollständig.

Beweis: Offensichtlich ist 3DM in NP. Wir werden zeigen, dass sich SATISFIABILITY polynomiell in 3DM transformiert. Gegeben sei eine Familie \mathcal{Z} von Klauseln Z_1, \dots, Z_m über $X = \{x_1, \dots, x_n\}$. Wir werden eine Instanz (U, V, W, T) von 3DM konstruieren, die genau dann eine Ja-Instanz ist, wenn \mathcal{Z} erfüllbar ist.

Wir definieren:

$$\begin{aligned}
 U &:= \{x_i^j, \bar{x}_i^j : i = 1, \dots, n; j = 1, \dots, m\} \\
 V &:= \{a_i^j : i = 1, \dots, n; j = 1, \dots, m\} \cup \{v^j : j = 1, \dots, m\} \\
 &\quad \cup \{c_k^j : k = 1, \dots, n-1; j = 1, \dots, m\} \\
 W &:= \{b_i^j : i = 1, \dots, n; j = 1, \dots, m\} \cup \{w^j : j = 1, \dots, m\} \\
 &\quad \cup \{d_k^j : k = 1, \dots, n-1; j = 1, \dots, m\} \\
 T_1 &:= \{(x_i^j, a_i^j, b_i^j), (\bar{x}_i^j, a_i^{j+1}, b_i^j) : i = 1, \dots, n; j = 1, \dots, m\}, \\
 &\quad \text{wobei } a_i^{m+1} := a_i^1 \\
 T_2 &:= \{(x_i^j, v^j, w^j) : i = 1, \dots, n; j = 1, \dots, m; x_i \in Z_j\} \\
 &\quad \cup \{(\bar{x}_i^j, v^j, w^j) : i = 1, \dots, n; j = 1, \dots, m; \bar{x}_i \in Z_j\} \\
 T_3 &:= \{(x_i^j, c_k^j, d_k^j), (\bar{x}_i^j, c_k^j, d_k^j) : i = 1, \dots, n; j = 1, \dots, m; k = 1, \dots, n-1\} \\
 T &:= T_1 \cup T_2 \cup T_3.
 \end{aligned}$$

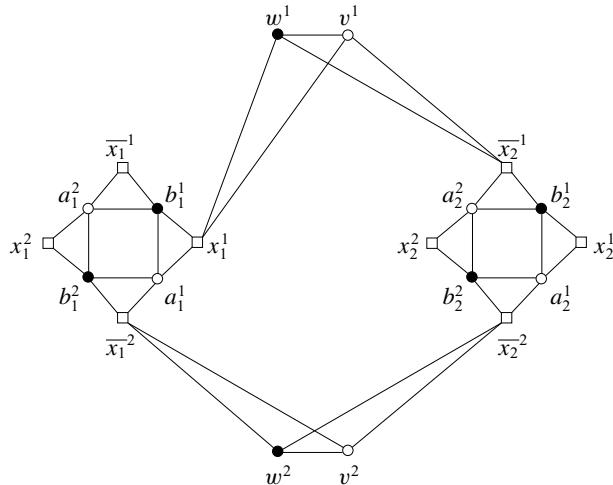


Abbildung 15.6.

Abbildung 15.6 zeigt diese Konstruktion mit dem Beispiel $m = 2$, $Z_1 = \{x_1, \bar{x}_2\}$, $Z_2 = \{\bar{x}_1, \bar{x}_2\}$. Jedes Dreieck entspricht einem Element von $T_1 \cup T_2$. Die Elemente c_k^j und d_k^j und die Tripel in T_3 sind nicht angegeben.

Angenommen, es ist (U, V, W, T) eine Ja-Instanz, und sei $M \subseteq T$ eine Lösung. Da die a_i^j und b_i^j nur in Elementen von T_1 vorkommen, haben wir für jedes i entweder $M \cap T_1 \supseteq \{(x_i^j, a_i^j, b_i^j) : j = 1, \dots, m\}$ oder $M \cap T_1 \supseteq \{(\bar{x}_i^j, a_i^{j+1}, b_i^j) : j = 1, \dots, m\}$. Im ersten Fall setzen wir x_i auf *false*, im letzteren auf *true*.

Ferner haben wir: Für jede Klausel Z_j ist $(\lambda^j, v^j, w^j) \in M$ für ein Literal $\lambda \in Z_j$. Da λ^j in keinem Element von $M \cap T_1$ vorkommt, ist dieses Literal auf *true* gesetzt. Somit haben wir eine erfüllende Wahrheitsbelegung.

Zur Umkehrung: Eine erfüllende Wahrheitsbelegung legt eine Menge $M_1 \subseteq T_1$ der Kardinalität nm und eine Menge $M_2 \subseteq T_2$ der Kardinalität m nahe, so dass für verschiedene $(u, v, w), (u', v', w') \in M_1 \cup M_2$ gilt: $u \neq u'$, $v \neq v'$ und $w \neq w'$. Nun lässt sich $M_1 \cup M_2$ leicht durch $(n-1)m$ Elemente von T_3 zu einer Lösung der 3DM-Instanz erweitern. \square

Ein einfach aussehendes Problem, von dem aber unbekannt ist, ob es in polynomieller Zeit lösbar ist, ist das folgende:

SUBSET-SUM

Instanz: Natürliche Zahlen c_1, \dots, c_n, K .

Frage: Gibt es eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} c_j = K$?

Korollar 15.27. (Karp [1972]) SUBSET-SUM ist NP-vollständig.

Beweis: Offensichtlich ist SUBSET-SUM in NP. Wir werden zeigen, dass sich 3DM polynomiell in SUBSET-SUM transformiert. Sei also (U, V, W, T) eine Instanz von 3DM. O. B. d. A. können wir annehmen, dass $U \cup V \cup W = \{u_1, \dots, u_{3m}\}$. Setze $S := \{(a, b, c) : (a, b, c) \in T\}$ und sei $S = \{s_1, \dots, s_n\}$.

Definiere

$$\begin{aligned} c_j &:= \sum_{u_i \in s_j} (n+1)^{i-1} \quad (j = 1, \dots, n), \text{ und} \\ K &:= \sum_{i=1}^{3m} (n+1)^{i-1}. \end{aligned}$$

Schreibt man die Zahl c_j in $(n+1)$ -ärer Form, so lässt sie sich als Inzidenzvektor von s_j ($j = 1, \dots, n$) auffassen und K besteht aus lauter Einsen. Somit entspricht jede Lösung der 3DM-Instanz einer Teilmenge R von S mit $\sum_{s_j \in R} c_j = K$ und umgekehrt. Auch gilt $\text{size}(c_j) \leq \text{size}(K) = O(m \log n)$, womit wir eine polynomielle Transformation haben. \square

Ein wichtiger Spezialfall ist das folgende Problem:

PARTITION

Instanz: Natürliche Zahlen c_1, \dots, c_n .

Frage: Gibt es eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$?

Korollar 15.28. (Karp [1972]) PARTITION ist NP-vollständig.

Beweis: Wir werden zeigen, dass sich SUBSET-SUM polynomiell in PARTITION transformiert. Sei also c_1, \dots, c_n, K eine Instanz von SUBSET-SUM. Indem wir ein Element $c_{n+1} := |\sum_{i=1}^n c_i - 2K|$ hinzufügen (sofern diese Zahl nicht gleich Null ist), erhalten wir eine Instanz c_1, \dots, c_{n+1} von PARTITION.

Fall 1: $2K \leq \sum_{i=1}^n c_i$. Dann gilt für jedes $I \subseteq \{1, \dots, n\}$:

$$\sum_{i \in I} c_i = K \quad \text{genau dann, wenn} \quad \sum_{i \in I \cup \{n+1\}} c_i = \sum_{i \in \{1, \dots, n\} \setminus I} c_i.$$

Fall 2: $2K > \sum_{i=1}^n c_i$. Dann gilt für jedes $I \subseteq \{1, \dots, n\}$:

$$\sum_{i \in I} c_i = K \quad \text{genau dann, wenn} \quad \sum_{i \in I} c_i = \sum_{i \in \{1, \dots, n+1\} \setminus I} c_i.$$

In beiden Fällen haben wir genau dann eine Ja-Instanz von PARTITION konstruiert, wenn die ursprüngliche Instanz von SUBSET-SUM eine Ja-Instanz ist. \square

Wir schließen mit dem folgenden Resultat:

Satz 15.29. GANZZAHLIGE LINEARE UNGLEICHUNGEN ist NP-vollständig.

Beweis: Nach Proposition 15.12 liegt GANZZAHLIGE LINEARE UNGLEICHUNGEN in NP. Jedes der obigen Probleme kann man leicht als Instanz von GANZZAHLIGE LINEARE UNGLEICHUNGEN formulieren. Z. B. ist eine PARTITION-Instanz c_1, \dots, c_n genau dann eine Ja-Instanz, wenn $\{x \in \mathbb{Z}^n : 0 \leq x \leq \mathbf{1}, 2c^\top x = c^\top \mathbf{1}\}$ nicht leer ist. \square

15.6 Die Klasse coNP

Die Definition von NP ist nicht symmetrisch bezüglich Ja- und Nein-Instanzen. Es ist z. B. eine offene Frage, ob das folgende Problem in NP ist: Ist es für einen gegebenen Graphen G wahr, dass G nicht hamiltonsch ist? Wir benötigen die folgenden weiteren Definitionen:

Definition 15.30. Für ein gegebenes Entscheidungsproblem $\mathcal{P} = (X, Y)$ ist $(X, X \setminus Y)$ das **komplementäre Entscheidungsproblem**. Die Klasse coNP beinhaltet alle Entscheidungsprobleme, deren komplementäre Entscheidungsprobleme in NP sind. Ein Entscheidungsproblem $\mathcal{P} \in \text{coNP}$ heißt **coNP-vollständig**, falls sich alle anderen Probleme in coNP polynomiell in \mathcal{P} transformieren.

Es ist trivial, dass das komplementäre Entscheidungsproblem eines Problems in P wieder in P liegt. Daraus folgt $P \subseteq coNP$, da $P \subseteq NP$. Es gibt die bekannte aber bisher unbewiesene Vermutung $NP \neq coNP$. Für diese Vermutung spielen die NP -vollständigen Probleme eine besondere Rolle:

Satz 15.31. *Ein Entscheidungsproblem ist genau dann $coNP$ -vollständig, wenn das komplementäre Entscheidungsproblem NP -vollständig ist. Gilt $NP \neq coNP$, so ist kein $coNP$ -vollständiges Problem in NP .*

Beweis: Die erste Aussage folgt direkt aus der Definition.

Angenommen, es gibt ein $coNP$ -vollständiges Problem $\mathcal{P} = (X, Y) \in NP$. Dann müssen wir zeigen, dass die Vermutung nicht gilt, dass also $NP = coNP$. Sei $\mathcal{Q} = (V, W)$ ein beliebiges Problem in $coNP$. Wir zeigen nun, dass $\mathcal{Q} \in NP$.

Da \mathcal{P} $coNP$ -vollständig ist, transformiert sich \mathcal{Q} polynomiell in \mathcal{P} . Somit gibt es einen polynomiellen Algorithmus, der jede Instanz v von \mathcal{Q} in eine Instanz $x = f(v)$ von \mathcal{P} transformiert, so dass $x \in Y$ genau dann, wenn $v \in W$. Beachte, dass $\text{size}(x) \leq p(\text{size}(v))$ für ein festes Polynom p .

Da $\mathcal{P} \in NP$, gibt es ein Polynom q und ein Entscheidungsproblem $\mathcal{P}' = (X', Y')$ in P , wobei $X' := \{x\#c : x \in X, c \in \{0, 1\}^{\lfloor q(\text{size}(x)) \rfloor}\}$, so dass

$$Y = \left\{ y \in X : \text{es gibt einen String } c \in \{0, 1\}^{\lfloor q(\text{size}(y)) \rfloor} \text{ mit } y\#c \in Y' \right\}$$

(siehe Definition 15.9). Wir definieren ein Entscheidungsproblem (V', W') folgendermaßen: $V' := \{v\#c : v \in V, c \in \{0, 1\}^{\lfloor q(p(\text{size}(v))) \rfloor}\}$, und $v\#c \in W'$ genau dann, wenn $f(v)\#c' \in Y'$, wobei c' aus den ersten $\lfloor q(\text{size}(f(v))) \rfloor$ Komponenten von c besteht.

Beachte, dass $(V', W') \in P$. Somit folgt $\mathcal{Q} \in NP$ nach Definition. Also haben wir $coNP \subseteq NP$ und mittels Symmetrie folgt $NP = coNP$. \square

Ist ein Problem bewiesenermaßen in $NP \cap coNP$, so sagt man, es hat eine **gute Charakterisierung** (Edmonds [1965]). Dies bedeutet, dass es sowohl für Ja- als auch für Nein-Instanzen Zertifikate gibt, die in polynomieller Zeit geprüft werden können. Satz 15.31 deutet an, dass ein Problem mit einer guten Charakterisierung wahrscheinlich nicht NP -vollständig ist.

Wir betrachten ein paar Beispiele. Proposition 2.9, Satz 2.24 und Proposition 2.27 liefern gute Charakterisierungen für die folgenden Probleme: Entscheide, ob ein gegebener Graph azyklisch ist, ob er einen eulerschen Spaziergang hat, bzw. ob er bipartit ist. Natürlich sind diese Probleme keine sonderlich interessanten Beispiele, da sie allesamt leicht polynomiell lösbar sind. Betrachte jedoch die Entscheidungsversion von LINEARE OPTIMIERUNG:

Satz 15.32. LINEARE UNGLEICHUNGEN ist in $NP \cap coNP$.

Beweis: Dies folgt sofort aus Satz 4.4 und Korollar 3.24. \square

Selbstverständlich folgt dieser Satz auch mittels jedes polynomiellen Algorithmus für LINEARE OPTIMIERUNG, z. B. Satz 4.18. Vor der Entdeckung der ELLIPSOIDMETHODE war Satz 15.32 jedoch der einzige theoretische Hinweis darauf,

dass LINEARE UNGLEICHUNGEN wahrscheinlich nicht NP -vollständig ist. Dies nährte die Hoffnung, einen polynomiellen Algorithmus für LINEARE OPTIMIERUNG zu finden (nach Proposition 4.16 reduziert sich LINEARE OPTIMIERUNG auf LINEARE UNGLEICHUNGEN); eine gerechtfertigte Hoffnung, wie wir heute wissen.

Das folgende berühmte Problem hat eine ähnliche Geschichte:

PRIM

Instanz: Eine Zahl $n \in \mathbb{N}$ (mit binärer Darstellung).

Frage: Ist n eine Primzahl?

Offensichtlich ist PRIM in $coNP$. Pratt [1975] hat bewiesen, dass PRIM auch in NP ist. Schließlich bewiesen Agrawal, Kayal und Saxena [2004], dass $PRIM \in P$, indem sie einen erstaunlich einfachen $O(\log^{7,5+\epsilon} n)$ -Algorithmus (für jedes $\epsilon > 0$) beschrieben. Davor war der beste bekannte deterministische Algorithmus für PRIM derjenige von Adleman, Pomerance und Rumely [1983], mit $O((\log n)^c \log \log \log n)$ Laufzeit für eine bestimmte Konstante c . Da die Inputgröße $O(\log n)$ ist, ist dies kein polynomieller Algorithmus.

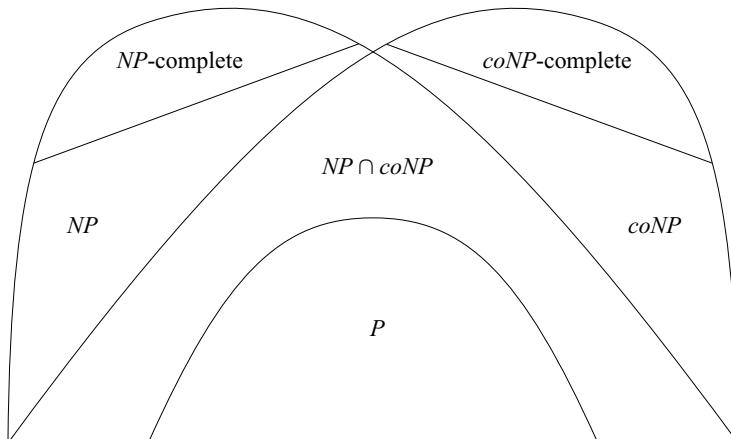


Abbildung 15.7.

Wir schließen diesen Abschnitt mit einer Abbildung der Inklusionsrelationen von NP und $coNP$ (Abb. 15.7). Ladner [1975] hat gezeigt: Gilt $P \neq NP$, so gibt es Probleme in $NP \setminus P$, die nicht NP -vollständig sind. Bis die Vermutung $P \neq NP$ entschieden ist, wissen wir jedoch nicht, ob nicht alle Gebiete in Abb. 15.7 zu einem Gebiet zusammenfallen.

15.7 NP-schwere Probleme

Nun werden wir die bisher gewonnenen Resultate auf allgemeine Berechnungsprobleme und insbesondere auf Optimierungsprobleme erweitern.

Definition 15.33. Ein Berechnungsproblem \mathcal{P} heißt **NP-schwer**, falls sich jedes Problem in NP polynomiell auf \mathcal{P} reduziert.

Beachte, dass die obige Definition auch für Entscheidungsprobleme gilt. Ferner ist sie symmetrisch (im Gegensatz zur NP -Vollständigkeit): Ein Entscheidungsproblem ist genau dann NP -schwer, wenn sein Komplement NP -schwer ist. NP -schwere Probleme sind mindestens so schwer wie die schwersten Probleme in NP . Manche können aber auch schwerer als jedes Problem in NP sein. Ein Problem, welches sich auf ein Problem in NP polynomiell reduziert, heißt **NP -leicht**. Ein Problem, welches sowohl NP -schwer wie auch NP -leicht ist, heißt **NP -äquivalent**. Mit anderen Worten, ein Problem ist genau dann NP -äquivalent, wenn es polynomiell äquivalent mit SATISFIABILITY ist, wobei zwei gegebene Probleme \mathcal{P} und \mathcal{Q} **polynomiell äquivalent** heißen, falls sich \mathcal{P} polynomiell auf \mathcal{Q} reduziert und \mathcal{Q} auf \mathcal{P} . Es gilt das folgende Resultat:

Proposition 15.34. Sei \mathcal{P} ein NP -äquivalentes Berechnungsproblem. Dann hat \mathcal{P} genau dann einen exakten polynomiellen Algorithmus, wenn $P = NP$. \square

Natürlich sind alle NP -vollständigen und alle $coNP$ -vollständigen Probleme NP -äquivalent. Fast alle in diesem Buch betrachteten Probleme sind NP -leicht, da sie sich polynomiell auf GANZZAHLIGE OPTIMIERUNG reduzieren; dies ist meistens so trivial, dass wir es gar nicht erwähnen.

Wir definieren nun formal diejenigen Typen von Optimierungsproblemen, die uns interessieren:

Definition 15.35. Ein **NP-Optimierungsproblem** ist ein Quadrupel $\mathcal{P} = (X, (S_x)_{x \in X}, c, \text{goal})$, wobei Folgendes gilt:

- X ist eine in polynomieller Zeit entscheidbare Sprache über $\{0, 1\}$;
- S_x ist eine nichtleere Teilmenge von $\{0, 1\}^*$ für jedes $x \in X$; es gibt ein Polynom p mit $\text{size}(y) \leq p(\text{size}(x))$ für alle $x \in X$ und $y \in S_x$; und die Sprache $\{(x, y) : x \in X, y \in S_x\}$ ist in polynomieller Zeit entscheidbar;
- $c : \{(x, y) : x \in X, y \in S_x\} \rightarrow \mathbb{Q}$ ist eine in polynomieller Zeit berechenbare Funktion; und
- $\text{goal} \in \{\max, \min\}$.

Die Elemente von X heißen **Instanzen** von \mathcal{P} . Für jede Instanz x heißen die Elemente von S_x **zulässige Lösungen** von x . Wir schreiben $\text{OPT}(x) := \text{goal}\{c(x, y) : y \in S_x\}$. Eine **optimale Lösung** von x ist eine zulässige Lösung y von x mit $c(x, y) = \text{OPT}(x)$.

Eine **Heuristik** für \mathcal{P} ist ein Algorithmus A , welcher für jeden Input $x \in X$ mit $S_x \neq \emptyset$ eine zulässige Lösung $y \in S_x$ berechnet. Gelegentlich schreiben wir $A(x) := c(x, y)$. Gilt $A(x) = \text{OPT}(x)$ für alle $x \in X$ mit $S_x \neq \emptyset$, so ist A ein **exakter Algorithmus** für \mathcal{P} .

In Abhängigkeit von dem Kontext nennt man $c(x, y)$ oft die Kosten, das Gewicht, den Gewinn oder die Länge von y . Ist c nichtnegativ, so sagen wir, das Optimierungsproblem habe nichtnegative Gewichte. Die Werte von c sind rationale Zahlen; wie üblich, setzen wir eine Kodierung als binäre Strings voraus.

Die meisten der interessanten Optimierungsprobleme liegen in dieser Klasse, es gibt aber Ausnahmen, (z. B. Aufgabe 24).

Ein Optimierungsproblem $(X, (S_x)_{x \in X}, c, \text{goal})$ kann als das Berechnungsproblem $(X, \{(x, y) : x \in X, y \in S_x, c(x, y) = \text{OPT}(x)\})$ betrachtet werden. Somit sind polynomielle Reduktionen auch auf Optimierungsprobleme anwendbar.

Satz 15.36. *Jedes NP-Optimierungsproblem ist NP-leicht.*

Beweis: Sei $\mathcal{P} = (X, (S_x)_{x \in X}, c, \text{goal})$ ein NP-Optimierungsproblem. Wir reduzieren \mathcal{P} polynomiell auf ein Entscheidungsproblem $\mathcal{Q} \in NP$. Wie üblich, nennen einen String $y \in \{0, 1\}^p$, $p \in \mathbb{Z}_+$, genau dann lexikographisch größer als einen String $s \in \{0, 1\}^q$, $q \in \mathbb{Z}_+$, wenn $y \neq s$ und $y_j > s_j$ für $j = \min\{i \in \mathbb{N} : y_i \neq s_i\}$, wobei $y_i := -1$ für $i > p$ und $s_i := -1$ für $i > q$.

Für $\text{goal} = \max$ definieren wir \mathcal{Q} wie folgt: Gibt es für gegebene $x \in X$, $\gamma \in \mathbb{Q}$ und $s \in \{0, 1\}^*$ ein $y \in S_x$, so dass $c(x, y) \geq \gamma$ und y gleich oder lexikographisch größer als s ist? Für $\text{goal} = \min$ wird $c(x, y) \geq \gamma$ durch $c(x, y) \leq \gamma$ ersetzt.

Beachte, dass \mathcal{Q} aus NP ist (y dient als Zertifikat). Wir reduzieren \mathcal{P} folgendermaßen polynomiell auf \mathcal{Q} .

Da c in polynomieller Laufzeit berechenbar ist, gibt es eine Konstante $d \in \mathbb{N}$, so dass $\text{size}(c(x, y)) \leq (\text{size}(x) + p(\text{size}(x)))^d =: k(x)$ für alle $x \in X$ und $y \in S_x$. Es folgt, dass $\text{OPT}(x) \in [-2^{k(x)}, 2^{k(x)}]$ und dass $|c(x, y) - c(x, y')|$ für alle $x \in X$ und $y, y' \in S_x$ ein ganzzahliges Vielfaches von $2^{-k(x)}$ ist.

Gegeben sei eine Instanz $x \in X$. Zunächst berechnen wir $k(x)$, und danach bestimmen wir $\text{OPT}(x)$ mittels binärer Suche. Wir beginnen mit $\alpha := -2^{k(x)}$ und $\beta := 2^{k(x)}$. In jeder Iteration wenden wir das Orakel auf (x, γ, s_0) an, wobei $\gamma = \frac{\alpha+\beta}{2}$ und s_0 der leere String ist. Ist die Antwort „ja“, so setzen wir $\alpha := \gamma$, sonst $\beta := \gamma$.

Nach $2k(x) + 2$ Iterationen haben wir $\beta - \alpha < 2^{-k(x)}$. Nun setzen wir $\gamma := \alpha$ und rufen das Orakel $2p(\text{size}(x))$ weitere Male auf, um eine Lösung $y \in S_x$ mit $c(x, y) \geq \alpha$ zu berechnen. Für $i := 1, \dots, p(\text{size}(x))$ wenden wir das Orakel auf (x, α, s_{i-1}^0) und (x, α, s_{i-1}^1) an, wobei s^j aus dem String s durch Anhängen des Symbols $j \in \{0, 1\}$ entsteht. Sind beide Antworten „ja“, so setzen wir $s_i := s_{i-1}^1$; ist nur die erste Antwort „ja“, so setzen wir $s_i := s_{i-1}^0$; sind beide Antworten „nein“, so setzen wir $s_i := s_{i-1}$. Hiermit folgt, dass $s_{p(\text{size}(x))}$ der lexikographisch maximale String y mit $y \in S_x$ und $c(x, y) = \text{OPT}(x)$ ist. \square

Die Mehrzahl der noch zu besprechenden Probleme sind auch NP-schwer, was wir meist dadurch beweisen, dass wir eine polynomielle Reduktion von einem NP -vollständigen Problem angeben. Als erstes Beispiel betrachten wir MAX-2SAT: Für eine gegebene Instanz von SATISFIABILITY mit genau zwei Literalen pro Klausel finde man eine die Anzahl der erfüllten Klauseln maximierende Wahrheitsbelegung.

Satz 15.37. (Garey, Johnson und Stockmeyer [1976]) MAX-2SAT ist NP-schwer.

Beweis: Der Beweis folgt durch Reduktion von 3SAT. Für eine gegebene Instanz I von 3SAT mit den Klauseln C_1, \dots, C_m , konstruieren wir eine Instanz I' von MAX-2SAT, indem wir neue Variablen $y_1, z_1, \dots, y_m, z_m$ hinzufügen und jede Klausel $C_i = \{\lambda_1, \lambda_2, \lambda_3\}$ durch die vierzehn folgenden Klauseln ersetzen:

$$\begin{aligned} & \{\lambda_1, z_i\}, \{\lambda_1, \bar{z}_i\}, \{\lambda_2, z_i\}, \{\lambda_2, \bar{z}_i\}, \{\lambda_3, z_i\}, \{\lambda_3, \bar{z}_i\}, \{y_i, z_i\}, \{y_i, \bar{z}_i\}, \\ & \{\lambda_1, \bar{y}_i\}, \{\lambda_2, \bar{y}_i\}, \{\lambda_3, \bar{y}_i\}, \{\bar{\lambda}_1, \bar{\lambda}_2\}, \{\bar{\lambda}_1, \bar{\lambda}_3\}, \{\bar{\lambda}_2, \bar{\lambda}_3\}. \end{aligned}$$

Beachte, dass keine Wahrheitsbelegung mehr als 11 dieser 14 Klauseln erfüllt. Beachte ferner: Sind 11 dieser Klauseln tatsächlich erfüllt, so ist mindestens eines von $\lambda_1, \lambda_2, \lambda_3$ true. Ist andererseits eines von $\lambda_1, \lambda_2, \lambda_3$ true, so können wir $y_i := \lambda_1 \wedge \lambda_2 \wedge \lambda_3$ und $z_i := \text{true}$ setzen, um 11 dieser Klauseln zu erfüllen.

Damit folgt, dass I genau dann eine alle m Klauseln erfüllende Wahrheitsbelegung hat, wenn I' eine $11m$ Klauseln erfüllende Wahrheitsbelegung hat. \square

Es ist eine offene Frage, ob jedes NP-schwere Entscheidungsproblem $\mathcal{P} \in NP$ NP-vollständig ist (beachte den Unterschied zwischen polynomieller Reduktion und polynomieller Transformation; Definitionen 15.15 und 15.17). In den Aufgaben 22 und 23 werden zwei NP-schwere Entscheidungsprobleme besprochen, die anscheinend nicht in NP sind. Siehe auch Aufgabe 2, Kapitel 19.

Gilt $P \neq NP$, so gibt es keinen exakten polynomiellen Algorithmus für irgendein NP-schweres Problem. Einen pseudopolynomiellen Algorithmus könnte es jedoch geben:

Definition 15.38. Sei \mathcal{P} ein Entscheidungs- oder ein Optimierungsproblem, für welches jede Instanz x aus einer Liste von nichtnegativen ganzen Zahlen besteht. Die größte dieser Zahlen bezeichnen wir mit $\text{largest}(x)$. Ein Algorithmus für \mathcal{P} heißt **pseudopolynomiell**, falls seine Laufzeit durch ein Polynom in $\text{size}(x)$ und $\text{largest}(x)$ beschränkt ist.

Z. B. gibt es einen trivialen pseudopolynomiellen Algorithmus für PRIM, der die auf die Primeigenschaft zu prüfende natürliche Zahl n durch jede natürliche Zahl von 2 bis $\lfloor \sqrt{n} \rfloor$ teilt. Ein weiteres Beispiel ist:

Satz 15.39. Es gibt einen pseudopolynomiellen Algorithmus für SUBSET-SUM.

Beweis: Für eine gegebene Instanz c_1, \dots, c_n, K von SUBSET-SUM konstruieren wir einen Digraphen G mit der Knotenmenge $\{0, \dots, n\} \times \{0, 1, 2, \dots, K\}$. Für jedes $j \in \{1, \dots, n\}$ fügen wir die Kanten $((j-1, i), (j, i))$ ($i = 0, 1, \dots, K$) und $((j-1, i), (j, i + c_j))$ ($i = 0, 1, \dots, K - c_j$) hinzu.

Beachte, dass jeder Weg von $(0, 0)$ nach (j, i) einer Teilmenge $S \subseteq \{1, \dots, j\}$ mit $\sum_{k \in S} c_k = i$ entspricht, und umgekehrt. Somit können wir unsere SUBSET-SUM-Instanz dadurch lösen, dass wir prüfen, ob G einen Weg von $(0, 0)$ nach (n, K) enthält. Mit dem GRAPH-SCANNING-ALGORITHMUS kann dies in $O(nK)$ -Zeit bewerkstelligt werden, womit wir einen pseudopolynomiellen Algorithmus haben. \square

Der obige Algorithmus ist auch ein pseudopolynomieller Algorithmus für PARTITION, da $\frac{1}{2} \sum_{i=1}^n c_i \leq \frac{n}{2} \text{largest}(c_1, \dots, c_n)$ gilt. In Abschnitt 17.2 werden wir eine Erweiterung dieses Algorithmus besprechen. Bei nicht allzu großen Zahlen kann ein pseudopolynomieller Algorithmus recht effizient sein. Aus diesem Grunde ist die folgende Definition nützlich:

Definition 15.40. Gegeben sei ein Entscheidungsproblem $\mathcal{P} = (X, Y)$ bzw. ein Optimierungsproblem $\mathcal{P} = (X, (S_x)_{x \in X}, c, \text{goal})$ und ferner eine Teilmenge $X' \subseteq X$ von Instanzen. Dann ist $\mathcal{P}' = (X', X' \cap Y)$ bzw. $\mathcal{P}' = (X', (S_x)_{x \in X'}, c, \text{goal})$ die **Restriktion** von \mathcal{P} auf X' .

Sei \mathcal{P} ein Entscheidungs- bzw. ein Optimierungsproblem, für welches jede Instanz aus einer Liste von Zahlen besteht. Sei \mathcal{P}_p für ein gegebenes Polynom p die Restriktion von \mathcal{P} auf Instanzen x , welche aus nichtnegativen ganzen Zahlen mit $\text{largest}(x) \leq p(\text{size}(x))$ bestehen. Dann heißt \mathcal{P} stark **NP-schwer**, falls es ein Polynom p gibt, so dass \mathcal{P}_p NP-schwer ist. \mathcal{P} heißt stark **NP-vollständig**, falls $\mathcal{P} \in \text{NP}$ und es ein Polynom p gibt, so dass \mathcal{P}_p NP-vollständig ist.

Proposition 15.41. Gilt $P \neq \text{NP}$, so gibt es keinen exakten pseudopolynomiellen Algorithmus für irgendein stark NP-schweres Problem. \square

Es folgen einige berühmte Beispiele:

Satz 15.42. GANZZAHLIGE OPTIMIERUNG ist stark NP-schwer.

Beweis: Für einen gegebenen ungerichteten Graphen G hat das ganzzahlige LP $\max\{\mathbb{1}x : x \in \mathbb{Z}^{V(G)}, 0 \leq x \leq \mathbb{1}, x_v + x_w \leq 1 \text{ für } \{v, w\} \in E(G)\}$ einen optimalen Zielfunktionswert von mindestens k genau dann, wenn G eine stabile Menge der Kardinalität k enthält. Da für alle nicht-trivialen Instanzen (G, k) von STABLE SET $k \leq |V(G)|$ gilt, folgt das Resultat mit Satz 15.23. \square

TRAVELING-SALESMAN-PROBLEM (TSP)

Instanz: Ein vollständiger Graph K_n ($n \geq 3$) und Gewichte $c : E(K_n) \rightarrow \mathbb{R}_+$.

Aufgabe: Bestimme einen Hamilton-Kreis T mit minimalem Gewicht $\sum_{e \in E(T)} c(e)$.

Die Knoten einer TSP-Instanz werden häufig Städte genannt und die Gewichte Entferungen.

Satz 15.43. Das TSP ist stark NP-schwer.

Beweis: Wir werden zeigen, dass das TSP NP-schwer ist, selbst wenn es auf Instanzen beschränkt wird, für die alle Entfernungen 1 oder 2 sind. Dazu geben wir eine polynomiale Reduktion von dem HAMILTON-KREIS-Problem an. Für einen gegebenen Graphen G mit $n \geq 3$ Knoten konstruieren wir die folgende Instanz von TSP: Jedem Knoten in G entspreche eine Stadt und für jede Kante aus $E(G)$ sei die Entfernung zwischen den zwei entsprechenden Städten gleich 1;

alle Entfernungen zwischen anderen Städtepaaren seien gleich 2. Dann folgt sofort: G ist hamiltonsch genau dann, wenn die Länge einer optimalen TSP-Tour n ist.

□

Dieser Beweis zeigt auch, dass das folgende Entscheidungsproblem nicht leichter als das TSP selbst ist: Gibt es für eine gegebene Instanz des TSP und eine ganze Zahl k eine Tour der Länge kleiner oder gleich k ? Eine ähnliche Aussage gilt für eine große Klasse von diskreten Optimierungsproblemen:

Proposition 15.44. *Seien \mathcal{F} und \mathcal{F}' (unendliche) Familien endlicher Mengen und sei \mathcal{P} das folgende Optimierungsproblem: Gegeben sei eine Menge $E \in \mathcal{F}$ und eine Funktion $c : E \rightarrow \mathbb{Z}$. Bestimme eine Menge $F \subseteq E$ mit $F \in \mathcal{F}'$ und $c(F)$ minimal (oder entscheide, dass es kein solches F gibt).*

Dann kann \mathcal{P} in polynomieller Zeit gelöst werden genau dann, wenn das folgende Entscheidungsproblem in polynomieller Zeit gelöst werden kann: Gilt für eine gegebene Instanz (E, c) von \mathcal{P} und eine ganze Zahl k , dass $\text{OPT}((E, c)) \leq k$? Ferner gilt: Ist das Optimierungsproblem \mathcal{P} NP-schwer, so auch dieses Entscheidungsproblem.

Beweis: Es genügt zu zeigen, dass es einen das Entscheidungsproblem benutzenden Orakel-Algorithmus für das Optimierungsproblem gibt (die Umkehrung ist trivial). Sei (E, c) eine Instanz von \mathcal{P} . Zunächst bestimmen wir $\text{OPT}((E, c))$ mittels binärer Suche. Da es höchstens $1 + \sum_{e \in E} |c(e)| \leq 2^{\text{size}(c)}$ mögliche Werte gibt, können wir dies mit $O(\text{size}(c))$ Iterationen, jede mit einem Orakelauftruf, erreichen.

Dann prüfen wir schrittweise für jedes Element von E , ob es eine optimale Lösung ohne dieses Element gibt. Dies kann dadurch bewerkstelligt werden, dass wir sein Gewicht erhöhen (etwa um eins) und dann prüfen, ob sich dadurch der Wert einer optimalen Lösung auch erhöht hat. Falls ja, nehmen wir die Erhöhung zurück, andernfalls bleibt es bei der Erhöhung. Nachdem alle Elemente von E geprüft worden sind, bilden diejenigen Elemente, deren Gewicht nicht erhöht wurde, eine optimale Lösung.

□

Beispiele, auf die das obige Resultat anwendbar ist, sind das TSP, das MAXIMUM-WEIGHT-CLIQUE-PROBLEM, das KÜRZESTE-WEGE-PROBLEM, das KNAPSACK-PROBLEM und viele weitere. Schulz [2009] und Orlin, Punnen und Schulz [2009] haben ähnliche Resultate für GANZZAHLIGE OPTIMIERUNG bewiesen.

Die folgenden Kapitel werden überwiegend NP-schwere kombinatorische Optimierungsprobleme behandeln. Da wir uns keinen exakten polynomiellen Algorithmus erhoffen können, gibt es im Wesentlichen zwei Alternativen: Approximationsalgorithmen, für die nicht garantiert werden kann, dass sie eine optimale Lösung finden (diese werden wir im nächsten Kapitel besprechen), und exakte Algorithmen, deren Worst-Case-Laufzeit nicht polynomiell beschränkt ist. Für Letzteres geben wir nun ein Beispiel.

Wir erinnern uns, dass VERTEX-COVER *NP*-vollständig ist. Eine Instanz ist ein Paar (G, k) , und es sei $n := |V(G)|$ und $m := |E(G)|$. Ein trivialer exakter Algorithmus mit Laufzeit $O(n^k m)$ besteht darin, einfach alle k -elementigen Knotenmengen zu überprüfen. Wenn also k fest und nicht Teil des Inputs ist, liefert das einen polynomiellen Algorithmus; ein solches Problem wird im Englischen oft „*fixed-parameter tractable*“ genannt. Es geht noch besser, wenn man vorher die folgende Präprozessierung anwendet, die auch *Kern-Reduktion* genannt wird:

Satz 15.45. *Für jede VERTEX-COVER-Instanz (G, k) kann man in linearer Zeit entweder entscheiden, ob G eine Knotenüberdeckung der Größe höchstens k hat, oder eine äquivalente Instanz (G', l) mit $l \leq k$ und $|V(G')| \leq |E(G')| \leq l^2$ berechnen.*

Beweis: Wir beschreiben einen rekursiven Algorithmus. Falls $k < 0$, geben wir „nein“ aus. Falls $k \geq 0$ und G keine Kanten enthält, geben wir „ja“ aus. Andernfalls löschen wir erst parallele Kanten und isolierte Knoten. Falls G dann einen Knoten v enthält, dessen Grad größer als k ist, wissen wir, dass v zu jeder Knotenüberdeckung der Größe höchstens k gehört; daher rufen wir den Algorithmus rekursiv mit dem Input $(G - v, k - 1)$ auf. Wenn andernfalls G einen Knoten v mit nur einem Nachbarn w enthält, dann gibt es eine kardinalitätsminimale Knotenüberdeckung, die w (und nicht v) enthält, also rufen wir den Algorithmus rekursiv mit dem Input $(G - w, k - 1)$ auf. Wenn andernfalls G keinen Knoten mit Grad größer k enthält und G mehr als k^2 Kanten hat, dann gibt es keine Knotenüberdeckung der Größe höchstens k ; in diesem Fall geben wir „nein“ aus. Wenn keiner dieser Reduktionsschritte ausgeführt werden kann, geben wir die Instanz aus, wie sie ist.

□

Für eine Verbesserung siehe Aufgabe 6 aus Kapitel 16. Satz 15.45 ergibt eine Gesamtaufzeit von $O(n + m + k^{2k})$. Wenn k klein ist, ist das auch für sehr große Graphen ein vernünftiger Algorithmus.

Aufgaben

1. Man beachte, dass es mehr Sprachen als Turingmaschinen gibt. Man leite hieraus ab, dass es Sprachen gibt, die nicht mit einer Turingmaschine entscheidbar sind.

Turingmaschinen können auch mittels binärer Strings kodiert werden. Man betrachte das berühmte HALTEPROBLEM: Gegeben seien zwei binäre Strings x und y , wobei x eine Turingmaschine Φ kodiert. Ist $\text{time}(\Phi, y) < \infty$?

Man beweise, dass das HALTEPROBLEM unentscheidbar ist (d. h. es gibt keinen Algorithmus für das HALTEPROBLEM).

Hinweis: Unter der Annahme, es gäbe einen solchen Algorithmus A , konstruiere man eine Turingmaschine, die für den Input x zunächst den Algorithmus A auf den Input (x, x) anwendet und genau dann terminiert, wenn $\text{output}(A, (x, x)) = 0$.

2. Man beschreibe eine Turingmaschine, die zwei Strings vergleicht: Der Input bestehe aus einem String $a\#b$ mit $a, b \in \{0, 1\}^*$, und der Output sei 1 für $a = b$ und 0 für $a \neq b$.
3. Ein bekanntes Maschinenmodell ist die **RAM-Maschine**. Sie arbeitet mit einer unendlichen Folge von Registern x_1, x_2, \dots und einem weiteren besonderen Register, nämlich dem Akkumulator Acc . Jedes Register kann eine beliebig große ganze Zahl speichern, die auch negativ sein darf. Ein RAM-Programm ist eine Folge von Befehlen. Es gibt zehn Befehlstypen (die Bedeutung wird rechts angegeben):

WRITE	k	$Acc := k.$
LOAD	k	$Acc := x_k.$
LOADI	k	$Acc := x_{x_k}.$
STORE	k	$x_k := Acc.$
STOREI	k	$x_{x_k} := Acc.$
ADD	k	$Acc := Acc + x_k.$
SUBTR	k	$Acc := Acc - x_k.$
HALF		$Acc := \lfloor Acc/2 \rfloor.$
IFPOS	i	If $Acc > 0$ then go to $\textcircled{i}.$
HALT		Stop.

Ein RAM-Programm ist eine Folge von m Befehlen; jeder ist einer der obigen, wobei $k \in \mathbb{Z}$ und $i \in \{1, \dots, m\}$. Die Berechnung beginnt mit Befehl 1; weiterhin verläuft sie erwartungsgemäß; wir werden hier keine formale Definition geben.

Die obige Befehlsliste kann erweitert werden. Wir sagen, ein Befehl kann mittels eines RAM-Programms in Zeit n simuliert werden, falls es durch RAM-Befehle so ersetzt werden kann, dass sich die Gesamtanzahl der Schritte in irgendeiner Berechnung um höchstens den Faktor n vergrößert.

- (a) Man zeige, dass die folgenden Befehle mittels kleiner RAM-Programme in konstanter Zeit simuliert werden können:

IFNEG	i	If $Acc < 0$ then go to $\textcircled{i}.$
IFZERO	i	If $Acc = 0$ then go to $\textcircled{i}.$

- * (b) Man zeige, dass die Befehle SUBTR bzw. HALF mittels RAM-Programmen, die nur die anderen acht Befehle benutzen, in $O(\text{size}(x_k))$ -Zeit bzw. $O(\text{size}(Acc))$ Zeit simuliert werden können.
- * (c) Man zeige, dass die folgenden Befehle mittels RAM-Programmen in $O(n)$ -Zeit simuliert werden können, wobei $n = \max\{\text{size}(x_k), \text{size}(Acc)\}$:

MULT	k	$Acc := Acc \cdot x_k.$
DIV	k	$Acc := \lfloor Acc/x_k \rfloor.$
MOD	k	$Acc := Acc \bmod x_k.$

- * 4. Sei $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ eine Abbildung. Man zeige: Gibt es eine f berechnende Turingmaschine Φ , so gibt es ein RAM-Programm (siehe Aufgabe 3) mit der Eigenschaft, dass die Berechnung für den Input x (in Acc) nach $O(\text{size}(x) + \text{time}(\Phi, x))$ Schritten mit $Acc = f(x)$ terminiert.
Man zeige ferner: Gibt es eine RAM-Maschine, die $f(x)$ in Acc für gegebenes x in Acc in höchstens $g(\text{size}(x))$ Schritten berechnet, so gibt es eine f berechnende Turingmaschine mit $\text{time}(\Phi, x) = O(g(\text{size}(x))^3)$.
- 5. Man beweise, dass die folgenden Entscheidungsprobleme in NP sind:
 - Gegeben seien zwei Graphen G und H . Ist G mit einem Teilgraphen von H isomorph?
 - Gegeben sei eine binär kodierte ganze Zahl n . Gibt es eine Primzahl p mit $n = p^p$?
 - Gegeben sei eine Matrix $A \in \mathbb{Z}^{m \times n}$ und ein Vektor $b \in \mathbb{Z}^m$. Ist das Polyeder $P = \{x : Ax \leq b\}$ beschränkt?
 - Gegeben sei eine Matrix $A \in \mathbb{Z}^{m \times n}$ und ein Vektor $b \in \mathbb{Z}^m$. Ist das Polyeder $P = \{x : Ax \leq b\}$ unbeschränkt?
- 6. Man beweise: Ist $\mathcal{P} \in NP$, so gibt es ein Polynom p , so dass \mathcal{P} mittels eines (deterministischen) Algorithmus der Zeitkomplexität $O(2^{p(n)})$ gelöst werden kann, wobei n die Inputgröße ist.
- 7. Man beweise, dass die Menge der Entscheidungsprobleme in NP abzählbar ist.
- 8. Sei \mathcal{Z} eine 2SAT-Instanz, d. h. eine Familie von Klauseln über X mit zwei Literalen pro Klausel. Man betrachte den folgenden Digraphen $G(\mathcal{Z})$: Es ist $V(G)$ die Menge der Literale über X . Es gibt die Kante $(\lambda_1, \lambda_2) \in E(G)$ genau dann, wenn die Klausel $\{\bar{\lambda}_1, \lambda_2\}$ ein Element von \mathcal{Z} ist.
 - Man zeige: Gibt es eine Variable x , so dass x und \bar{x} in derselben starken Zusammenhangskomponente von $G(\mathcal{Z})$ liegen, so ist \mathcal{Z} nicht erfüllbar.
 - Man beweise die Umkehrung von (a).
 - Man gebe einen Algorithmus mit linearer Laufzeit für 2SAT an.
- 9. Man beschreibe einen Algorithmus mit linearer Laufzeit, der für jede SATISFIABILITY-Instanz eine Wahrheitsbelegung bestimmt, die mindestens die Hälfte aller Klauseln erfüllt.
- 10. Man betrachte SATISFIABILITY-Instanzen, in denen jede Klausel von einer der Formen $\{x\}$, $\{\bar{x}\}$ oder $\{\bar{x}, y\}$ ist, wobei x und y Variablen sind. Gegeben sei eine solche Instanz und ein nichtnegatives Gewicht für jede Klausel. Man finde (in polynomieller Laufzeit) eine Wahrheitsbelegung, die das Gesamtgewicht der erfüllten Klauseln maximiert.
Hinweis: Man reduziere dies auf das MINIMUM-CAPACITY-CUT-PROBLEM.
- 11. Man betrachte 3-OCCURRENCE-SAT, d. h. SATISFIABILITY beschränkt auf Instanzen, wo jede Klausel höchstens drei Literale enthält und jede Variable in höchstens drei Klauseln vorkommt. Man beweise, dass sogar diese eingeschränkte Version von SATISFIABILITY NP -vollständig ist.

12. Sei $\kappa : \{0, 1\}^m \rightarrow \{0, 1\}^m$ mit $m \geq 2$ eine (nicht notwendigerweise bijektive) Abbildung. Für $x = (x_1, \dots, x_n) \in \{0, 1\}^m \times \dots \times \{0, 1\}^m = \{0, 1\}^{nm}$ sei $\kappa(x) := (\kappa(x_1), \dots, \kappa(x_n))$, und für ein Entscheidungsproblem $\mathcal{P} = (X, Y)$ mit $X \subseteq \bigcup_{n \in \mathbb{Z}_+} \{0, 1\}^{nm}$ sei $\kappa(\mathcal{P}) := (\{\kappa(x) : x \in X\}, \{\kappa(x) : x \in Y\})$. Man beweise:
- Für alle Kodierungen κ und alle $\mathcal{P} \in NP$ folgt, dass auch $\kappa(\mathcal{P}) \in NP$.
 - Ist $\kappa(\mathcal{P}) \in P$ für alle Kodierungen κ und alle $\mathcal{P} \in P$, so ist $P = NP$. (Papadimitriou [1994])
13. Man beweise, dass STABLE-SET NP -vollständig ist, selbst wenn es auf Graphen mit Grad höchstens gleich 4 beschränkt wird.
Hinweis: Man verwende Aufgabe 11.
14. Man beweise, dass das folgende als DOMINATING-SET bekannte Problem NP -vollständig ist. Gegeben sei ein ungerichteter Graph G und eine Zahl $k \in \mathbb{N}$. Gibt es eine Menge $X \subseteq V(G)$ mit $|X| \leq k$, so dass $X \cup \Gamma(X) = V(G)$?
Hinweis: Man betrachte eine Transformation von VERTEX-COVER.
15. Das Entscheidungsproblem CLIQUE ist NP -vollständig. Ist es weiterhin NP -vollständig (vorausgesetzt, dass $P \neq NP$), wenn es auf die folgenden Graphen beschränkt wird?
 - Bipartite Graphen,
 - planare Graphen,
 - 2-fach zusammenhängende Graphen.
16. Man zeige für jedes der folgenden Entscheidungsprobleme, dass es entweder in P ist, oder NP -vollständig ist. Sei G ein ungerichteter Graph. Enthält G
 - einen Kreis der Länge mindestens 17?
 - einen mindestens die Hälfte der Knoten enthaltenden Kreis?
 - einen Kreis mit ungerader Länge?
 - eine mindestens die Hälfte der Knoten enthaltende Clique?
 - zwei Cliques mit der Eigenschaft: Jeder Knoten liegt in wenigstens einer der beiden Cliques?
17. Man beweise die NP -Vollständigkeit der folgenden Probleme:
 - HAMILTONSCHER WEG und GERICHTETER HAMILTONSCHER WEG
Enthält ein gegebener (gerichteter oder ungerichteter) Graph G einen hamiltonschen Weg?
 - KÜRZESTER WEG
Gegeben sei ein Graph G (gerichtet oder ungerichtet), Gewichte $c : E(G) \rightarrow \mathbb{Z}$, zwei Knoten $s, t \in V(G)$ und eine ganze Zahl k . Gibt es einen $s-t$ -Weg mit Gewicht höchstens k ?
 - 3-MATROID-INTERSEKTION
Gegeben seien drei Matroide $(E, \mathcal{F}_1), (E, \mathcal{F}_2), (E, \mathcal{F}_3)$ (durch Unabhängigkeitsrakel) und eine Zahl $k \in \mathbb{N}$. Entscheide, ob es eine Menge $F \in \mathcal{F}_1 \cap \mathcal{F}_2 \cap \mathcal{F}_3$ mit $|F| \geq k$ gibt.
 - CHINESISCHES POSTBOTEN-PROBLEM
Gegeben seien ungerichtete Graphen G und H mit $V(G) = V(H)$, Gewichte $c : E(H) \rightarrow \mathbb{Z}_+$ und eine ganze Zahl k . Gibt es eine Teilmenge $F \subseteq E(H)$ mit $c(F) \leq k$, so dass der Graph $(V(G), E(G) \cup F)$ zusammenhängend und eulersch ist?

18. Entweder man bestimme einen polynomiellen Algorithmus für die folgenden Entscheidungsprobleme, oder man beweise ihre *NP*-Vollständigkeit:
- Gibt es für einen gegebenen ungerichteten Graphen G und eine gegebene Knotenmenge $T \subseteq V(G)$ einen aufspannenden Baum in G , so dass alle Knoten in T Blätter sind?
 - Gibt es für einen gegebenen ungerichteten Graphen G und eine gegebene Knotenmenge $T \subseteq V(G)$ einen aufspannenden Baum in G , so dass alle Blätter Elemente von T sind?
 - Gegeben sei ein Digraph G , Gewichte $c : E(G) \rightarrow \mathbb{R}$, eine Knotenmenge $T \subseteq V(G)$ und eine Zahl k . Gibt es ein Branching B mit $|\delta_B^+(x)| \leq 1$ für alle $x \in T$ und $c(B) \geq k$?
19. Man beweise, dass das folgende Entscheidungsproblem in *coNP* ist: Ist das Polyeder $\{x : Ax \leq b\}$ für eine gegebene Matrix $A \in \mathbb{Q}^{m \times n}$ und einen gegebenen Vektor $b \in \mathbb{Q}^n$ ganzzahlig?
- Hinweis:* Man verwende Proposition 3.9, Lemma 5.12 und Satz 5.14.
- Bemerkung:* Es ist nicht bekannt, ob dieses Problem in *NP* ist.
20. Man beweise, dass das folgende Problem in *coNP* ist: Man entscheide für eine gegebene Matrix $A \in \mathbb{Z}^{m \times n}$ und einen Vektor $b \in \mathbb{Z}^m$, ob das Polyeder $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ ganzzahlig ist.
- Bemerkung:* Dieses Problem ist in der Tat *coNP*-vollständig, wie von Papadimitriou und Yannakakis [1990] gezeigt wurde.
21. Man definiere *boolesche Formeln* folgendermaßen: Sei X eine Variablenmenge. Es seien *true* und *false* die booleschen Formeln über X der Länge Null, die Literale seien die booleschen Formeln über X der Länge Eins, und die booleschen Formeln über X der Länge $k \geq 2$ seien die Strings $(\psi \wedge \psi')$ und $(\psi \vee \psi')$ für alle booleschen Formeln ψ der Länge $l \in \mathbb{N}$ und ψ' der Länge $l' \in \mathbb{N}$ mit $l + l' = k$. Eine gegebene Wahrheitsbelegung $T : X \rightarrow \{\text{true}, \text{false}\}$ erweitere man auf die booleschen Formeln über X , indem man $T((\psi \wedge \psi')) := T(\psi) \wedge T(\psi')$ und $T((\psi \vee \psi')) := T(\psi) \vee T(\psi')$ setzt. Zwei boolesche Formeln ψ und ψ' über X heißen *äquivalent*, falls $T(\psi) = T(\psi')$ für alle Wahrheitsbelegungen $T : X \rightarrow \{\text{true}, \text{false}\}$.
- Man beweise, dass das folgende Problem, genannt **BOOLESCHE ÄQUIVALENZ**, *coNP*-vollständig ist: Sind zwei gegebene boolesche Formeln über einer Variablenmenge X äquivalent?
22. Man zeige, dass das folgende Problem *NP*-schwer ist (es ist nicht bekannt, ob dieses Problem in *NP* ist): Erfüllt für eine gegebene Instanz von **SATISFIABILITY** die Mehrzahl aller Wahrheitsbelegungen alle Klauseln?
23. Man zeige, dass sich **PARTITION** polynomiell in das folgende Problem transformiert (Letzteres ist somit *NP*-schwer; ob es in *NP* ist, ist nicht bekannt; siehe Haase und Kiefer [2016]):

K-SCHWERSTE TEILMENGE

Instanz: Ganze Zahlen c_1, \dots, c_n, K, L .

Frage: Gibt es K paarweise verschiedene Teilmengen $S_1, \dots, S_K \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S_i} c_j \geq L$ für $i = 1, \dots, K$?

24. Man beweise, dass das folgende Problem, genannt LOGIK-MINIMIERUNG, genau dann in polynomieller Zeit gelöst werden kann, wenn $P = NP$: Für eine gegebene Variablenmenge X und eine boolesche Formel über X bestimme man eine äquivalente boolesche Formel über X mit minimaler Länge.

Hinweis: Man benutze Aufgabe 21.

Literatur

Allgemeine Literatur:

- Aho, A.V., Hopcroft, J.E., und Ullman, J.D. [1974]: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading 1974
- Arora, S., und Barak, B. [2009]: Computational Complexity: A Modern Approach. Cambridge University Press, New York 2009
- Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., und Protasi, M. [1999]: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer, Berlin 1999
- Bovet, D.B., und Crescenzi, P. [1994]: Introduction to the Theory of Complexity. Prentice-Hall, New York 1994
- Garey, M.R., und Johnson, D.S. [1979]: Computers and Intractability: A Guide to the Theory of NP -Completeness. Freeman, San Francisco 1979, Kapitel 1–3, 5, und 7
- Goldreich, O. [2008]: Computational Complexity: A Conceptual Perspective. Cambridge University Press, New York 2008
- Horowitz, E., und Sahni, S. [1978]: Fundamentals of Computer Algorithms. Computer Science Press, Potomac 1978, Kapitel 11
- Johnson, D.S. [1981]: The NP -completeness column: an ongoing guide. Journal of Algorithms starting with Vol. 4 (1981)
- Karp, R.M. [1975]: On the complexity of combinatorial problems. Networks 5 (1975), 45–68
- Papadimitriou, C.H. [1994]: Computational Complexity. Addison-Wesley, Reading 1994
- Papadimitriou, C.H., und Steiglitz, K. [1982]: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, Kapitel 15 und 16
- Wegener, I. [2005]: Complexity Theory: Exploring the Limits of Efficient Algorithms. Springer, Berlin 2005

Zitierte Literatur:

- Adleman, L.M., Pomerance, C., und Rumely, R.S. [1983]: On distinguishing prime numbers from composite numbers. Annals of Mathematics 117 (1983), 173–206
- Agrawal, M., Kayal, N., und Saxena, N. [2004]: PRIMES is in P. Annals of Mathematics 160 (2004), 781–793
- Cook, S.A. [1971]: The complexity of theorem proving procedures. Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing (1971), 151–158
- Edmonds, J. [1965]: Minimum partition of a matroid into independent subsets. Journal of Research of the National Bureau of Standards B 69 (1965), 67–72
- van Emde Boas, P. [1990]: Machine models and simulations. In: Handbook of Theoretical Computer Science; Volume A; Algorithms and Complexity (J. van Leeuwen, Hrsg.), Elsevier, Amsterdam 1990, pp. 1–66

- Fürer, M. [2009]: Faster integer multiplication. *SIAM Journal on Computing* 39 (2009), 979–1005
- Garey, M.R., Johnson, D.S., und Stockmeyer, L. [1976]: Some simplified *NP*-complete graph problems. *Theoretical Computer Science* 1 (1976), 237–267
- Haase, C., und Kiefer, S. [2016]: The complexity of the K th largest subset problem and related problems. *Information Processing Letters* 116 (1016), 111–115
- Harvey, D., van der Hoeven, J., und Lecerf, G. [2016]: Even faster integer multiplication. *Journal of Complexity* 36 (2016), 1–30
- Hopcroft, J.E., und Ullman, J.D. [1979]: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading 1979
- Karp, R.M. [1972]: Reducibility among combinatorial problems. In: *Complexity of Computer Computations* (R.E. Miller, J.W. Thatcher, Hrsg.), Plenum Press, New York 1972, pp. 85–103
- Ladner, R.E. [1975]: On the structure of polynomial time reducibility. *Journal of the ACM* 22 (1975), 155–171
- Lewis, H.R., und Papadimitriou, C.H. [1981]: Elements of the Theory of Computation. Prentice-Hall, Englewood Cliffs 1981
- Orlin, J.B., Punnen, A.P., und Schulz, A.S. [2009]: Integer programming: optimization and evaluation are equivalent. In: *Algorithms and Data Structures – Proceedings of the 11th Algorithms and Data Structures Symposium; LNCS 5664* (F. Dehne, M. Gavrilova, J.-R. Sack, C.D. Tóth, Hrsg.), Springer, Berlin 2009, pp. 519–529
- Papadimitriou, C.H., und Yannakakis, M. [1990]: On recognizing integer polyhedra. *Combinatorica* 10 (1990), 107–109
- Pratt, V. [1975]: Every prime has a succinct certificate. *SIAM Journal on Computing* 4 (1975), 214–220
- Schönhage, A., und Strassen, V. [1971]: Schnelle Multiplikation großer Zahlen. *Computing* 7 (1971), 281–292
- Schulz, A.S. [2009]: On the relative complexity of 15 problems related to 0/1-integer programming. In: *Research Trends in Combinatorial Optimization* (W.J. Cook, L. Lovász, J. Vygen, Hrsg.), Springer, Berlin 2009, pp. 399–428
- Turing, A.M. [1936]: On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* (2) 42 (1936), 230–265 und 43 (1937), 544–546



16 Approximationssalgorithmen

In diesem Kapitel führen wir den sehr wichtigen Begriff des Approximationssalgorithmus ein. Bisher haben wir hauptsächlich polynomiell lösbarer Probleme betrachtet. In den verbleibenden Kapiteln werden wir einige Strategien zur Handhabung von *NP*-schweren kombinatorischen Optimierungsproblemen besprechen. An erster Stelle stehen hier die Approximationssalgorithmen.

Der Idealfall ist der, in dem es eine Garantie dafür gibt, dass die Lösung nur um einen konstanten Betrag vom Optimum abweicht:

Definition 16.1. Ein **absoluter Approximationssalgorithmus** für ein Optimierungsproblem \mathcal{P} ist ein polynomieller Algorithmus A für \mathcal{P} zusammen mit einer Konstante k , so dass für alle Instanzen I von \mathcal{P}

$$|A(I) - \text{OPT}(I)| \leq k.$$

Leider sind es bisher nur sehr wenige klassische *NP*-schwere Optimierungsprobleme, für die ein absoluter Approximationssalgorithmus bekannt ist. Hierzu werden wir in Abschnitt 16.3 zwei Hauptbeispiele besprechen, nämlich das KANTENFÄRBUNGS-PROBLEM und das KNOTENFÄRBUNGS-PROBLEM in planaren Graphen.

In den meisten Fällen müssen wir uns mit relativen Approximationsgüten begnügen. Hierbei müssen wir uns auf Probleme mit nichtnegativen Gewichten beschränken.

Definition 16.2. Sei \mathcal{P} ein Optimierungsproblem mit nichtnegativen Gewichten und $k \geq 1$. Ein **k -Approximationssalgorithmus** für \mathcal{P} ist ein polynomieller Algorithmus A für \mathcal{P} mit

$$\frac{1}{k} \text{OPT}(I) \leq A(I) \leq k \text{OPT}(I)$$

für alle Instanzen I von \mathcal{P} . Wir sagen auch, dass A die **Approximationsgüte k** hat.

Die erste der beiden Ungleichungen ist für Maximierungsprobleme, die zweite für Minimierungsprobleme. Beachte, dass wir für Instanzen I mit $\text{OPT}(I) = 0$ eine exakte Lösung benötigen. Beachte ferner, dass die 1-Approximationssalgorithmen gerade die exakten polynomiellen Algorithmen sind. Gelegentlich wird die obige Definition auf den Fall erweitert, dass k eine Funktion der Instanz I ist, anstatt einer Konstante. Ein Beispiel hierzu werden wir im nächsten Abschnitt kennenlernen.

In Abschnitt 13.4 haben wir gesehen, dass der BEST-IN-GREEDY-ALGORITHMUS für das MAXIMIERUNGSPROBLEM für ein Unabhängigkeitssystem (E, \mathcal{F}) die Approximationsgüte $\frac{1}{q(E, \mathcal{F})}$ hat (Satz 13.19). In Abschnitt 14.6 haben wir außerdem einen Approximationsalgorithmus für die Maximierung submodularer Funktionen gesehen. In den folgenden Abschnitten und den weiteren Kapiteln werden wir Beispiele zu den obigen Definitionen kennenlernen. Ferner werden wir die Approximierbarkeit verschiedener *NP*-schwerer Probleme analysieren. Wir beginnen mit Überdeckungsproblemen.

16.1 Das Set-Covering-Problem

In diesem Abschnitt behandeln wir das folgende recht allgemeine Problem:

MINIMUM-WEIGHT-SET-COVER-PROBLEM

Instanz: Ein Mengensystem (U, \mathcal{S}) mit $\bigcup_{S \in \mathcal{S}} S = U$, Gewichte $c : \mathcal{S} \rightarrow \mathbb{R}_+$.

Aufgabe: Bestimme eine **Überdeckung** von (U, \mathcal{S}) (d. h. eine Unterfamilie $\mathcal{R} \subseteq \mathcal{S}$ mit $\bigcup_{R \in \mathcal{R}} R = U$) mit minimalem Gewicht.

Ist $c \equiv 1$, so heißt das Problem MINIMUM-SET-COVER-PROBLEM. Ein weiterer interessanter Spezialfall wird durch $|\{S \in \mathcal{S} : x \in S\}| = 2$ für alle $x \in U$ gegeben; dies ist das MINIMUM-WEIGHT-VERTEX-COVER-PROBLEM: Für einen gegebenen Graphen G mit $c : V(G) \rightarrow \mathbb{R}_+$ wird die entsprechende Set-Covering-Instanz durch $U := E(G)$, $\mathcal{S} := \{\delta(v) : v \in V(G)\}$ und $c(\delta(v)) := c(v)$ für alle $v \in V(G)$ definiert. Da das MINIMUM-WEIGHT-VERTEX-COVER-PROBLEM sogar für den Spezialfall $c \equiv 1$ *NP*-schwer ist (Satz 15.24), ist auch das MINIMUM-SET-COVER-PROBLEM *NP*-schwer.

Johnson [1974] und Lovász [1975] haben einen einfachen Greedy-Algorithmus für das MINIMUM-SET-COVER-PROBLEM vorgeschlagen: In jeder Iteration wähle man eine Menge, die eine maximale Anzahl von noch nicht überdeckten Elementen überdeckt. Chvátal [1979] hat diesen Algorithmus auf den gewichteten Fall erweitert:

GREEDY-ALGORITHMUS FÜR SET-COVER

Input: Ein Mengensystem (U, \mathcal{S}) mit $\bigcup_{S \in \mathcal{S}} S = U$, Gewichte $c : \mathcal{S} \rightarrow \mathbb{R}_+$.

Output: Eine Überdeckung \mathcal{R} von (U, \mathcal{S}) .

- ① Setze $\mathcal{R} := \emptyset$ und $W := \emptyset$.
- ② **While** $W \neq U$ **do**:
Wähle eine Menge $R \in \mathcal{S} \setminus \mathcal{R}$ mit $R \setminus W \neq \emptyset$ und minimalem $\frac{c(R)}{|R \setminus W|}$.
Setze $\mathcal{R} := \mathcal{R} \cup \{R\}$ und $W := W \cup R$.

Dieser Algorithmus hat offensichtlich $O(|U||\mathcal{S}|)$ -Laufzeit. Er hat die folgende Approximationsgüte:

Satz 16.3. (Chvátal [1979]) *Für jede Instanz (U, \mathcal{S}, c) des MINIMUM-WEIGHT-SET-COVER-PROBLEMS bestimmt der GREEDY-ALGORITHMUS FÜR SET-COVER*

eine Überdeckung, deren Gewicht höchstens gleich $H(r) \text{OPT}(U, \mathcal{S}, c)$ ist, wobei $r := \max_{S \in \mathcal{S}} |S|$ und $H(r) = 1 + \frac{1}{2} + \dots + \frac{1}{r}$.

Beweis: Sei (U, \mathcal{S}, c) eine Instanz des MINIMUM-WEIGHT-SET-COVER-PROBLEMS und $\mathcal{R} = \{R_1, \dots, R_k\}$ die mit obigem Algorithmus gefundene Lösung, wobei R_i die in der i -ten Iteration gewählte Menge ist. Sei $W_j := \bigcup_{i=1}^j R_i$ für $j = 0, \dots, k$.

Für jedes $e \in U$ sei $j(e) := \min\{j \in \{1, \dots, k\} : e \in R_j\}$ diejenige Iteration, in der e überdeckt wird. Sei ferner

$$y(e) := \frac{c(R_{j(e)})}{|R_{j(e)} \setminus W_{j(e)-1}|}.$$

Sei $S \in \mathcal{S}$ fest und $k' := \max\{j(e) : e \in S\}$. Dann haben wir

$$\begin{aligned} \sum_{e \in S} y(e) &= \sum_{i=1}^{k'} \sum_{e \in S: j(e)=i} y(e) \\ &= \sum_{i=1}^{k'} \frac{c(R_i)}{|R_i \setminus W_{i-1}|} |S \cap (W_i \setminus W_{i-1})| \\ &= \sum_{i=1}^{k'} \frac{c(R_i)}{|R_i \setminus W_{i-1}|} (|S \setminus W_{i-1}| - |S \setminus W_i|) \\ &\leq \sum_{i=1}^{k'} \frac{c(S)}{|S \setminus W_{i-1}|} (|S \setminus W_{i-1}| - |S \setminus W_i|) \end{aligned}$$

wegen der Wahl von R_i in ② (beachte, dass $S \setminus W_{i-1} \neq \emptyset$ für $i = 1, \dots, k'$). Setzen wir $s_i := |S \setminus W_{i-1}|$, so folgt

$$\begin{aligned} \sum_{e \in S} y(e) &\leq c(S) \sum_{i=1}^{k'} \frac{s_i - s_{i+1}}{s_i} \\ &\leq c(S) \sum_{i=1}^{k'} \left(\frac{1}{s_i} + \frac{1}{s_i - 1} + \dots + \frac{1}{s_{i+1} + 1} \right) \\ &= c(S) \sum_{i=1}^{k'} (H(s_i) - H(s_{i+1})) \\ &= c(S)(H(s_1) - H(s_{k'+1})) \\ &\leq c(S)H(s_1). \end{aligned}$$

Da $s_1 = |S| \leq r$, folgt schließlich

$$\sum_{e \in S} y(e) \leq c(S)H(r).$$

Nun summieren wir über alle $S \in \mathcal{O}$ für eine optimale Überdeckung \mathcal{O} und bekommen somit

$$\begin{aligned}
c(\mathcal{O})H(r) &\geq \sum_{S \in \mathcal{O}} \sum_{e \in S} y(e) \\
&\geq \sum_{e \in U} y(e) \\
&= \sum_{i=1}^k \sum_{e \in U: j(e)=i} y(e) \\
&= \sum_{i=1}^k c(R_i) = c(\mathcal{R}). \quad \square
\end{aligned}$$

Siehe Slavík [1997] für eine etwas strengere Analyse des ungewichteten Falls. Raz und Safra [1997] haben entdeckt, dass es eine Konstante $c > 0$ mit folgender Eigenschaft gibt: Gilt $P \neq NP$, so ist eine Approximationsgüte von $c \ln |U|$ unerreichbar. In der Tat gilt: Gibt es bloß ein einziges Problem in NP , welches nicht in $O(n^{O(\log \log n)})$ -Zeit lösbar ist, so ist eine Approximationsgüte von $c \ln |U|$ für jedes $c < 1$ unerreichbar (Feige [1998]).

Das MINIMUM-WEIGHT-EDGE-COVER-PROBLEM ist offensichtlich ein Spezialfall des MINIMUM-WEIGHT-SET-COVER-PROBLEMS. Hier haben wir $r = 2$ in Satz 16.3, somit ist der obige Algorithmus ein $\frac{3}{2}$ -Approximationsalgorithmus in diesem Spezialfall. Dieses Problem kann jedoch auch in polynomieller Zeit optimal gelöst werden; siehe Aufgabe 13, Kapitel 11.

Für das MINIMUM-VERTEX-COVER-PROBLEM lautet der obige Algorithmus wie folgt:

GREEDY-ALGORITHMUS FÜR VERTEX-COVER

Input: Ein Graph G .

Output: Eine Knotenüberdeckung R von G .

① Setze $R := \emptyset$.

② **While** $E(G) \neq \emptyset$ **do**:

Wähle einen Knoten $v \in V(G) \setminus R$ mit maximalem Grad.

Setze $R := R \cup \{v\}$ und entferne alle mit v inzidenten Kanten.

Dieser Algorithmus sieht vernünftig aus und somit könnte man fragen, für welche k er ein k -Approximationsalgorithmus ist. Es ist vielleicht erstaunlich, dass es kein solches k gibt. Die in Satz 16.3 angegebene Schranke ist nämlich die fast bestmögliche:

Satz 16.4. (Johnson [1974], Papadimitriou und Steiglitz [1982]) *Für jedes $n \geq 3$ gibt es eine Instanz G des MINIMUM-VERTEX-COVER-PROBLEMS mit den*

Eigenschaften: $nH(n-1)+2 \leq |V(G)| \leq nH(n-1)+n$, der maximale Grad von G ist $n-1$, $\text{OPT}(G) = n$ und der obige Algorithmus kann eine Knotenüberdeckung finden, die alle außer n Knoten enthält.

Beweis: Für alle $n \geq 3$ und $i \leq n$ setzen wir $A_n^i := \sum_{j=2}^i \left\lfloor \frac{n}{j} \right\rfloor$ und

$$\begin{aligned} V(G_n) &:= \{a_1, \dots, a_{A_n^{n-1}}, b_1, \dots, b_n, c_1, \dots, c_n\}, \\ E(G_n) &:= \{\{b_i, c_i\} : i = 1, \dots, n\} \cup \\ &\quad \bigcup_{i=2}^{n-1} \bigcup_{j=A_n^{i-1}+1}^{A_n^i} \{\{a_j, b_k\} : (j - A_n^{i-1} - 1)i + 1 \leq k \leq (j - A_n^{i-1})i\}. \end{aligned}$$

Beachte, dass $|V(G_n)| = 2n + A_n^{n-1}$, $A_n^{n-1} \leq nH(n-1) - n$ und $A_n^{n-1} \geq nH(n-1) - n - (n-2)$. Abbildung 16.1 zeigt G_6 .

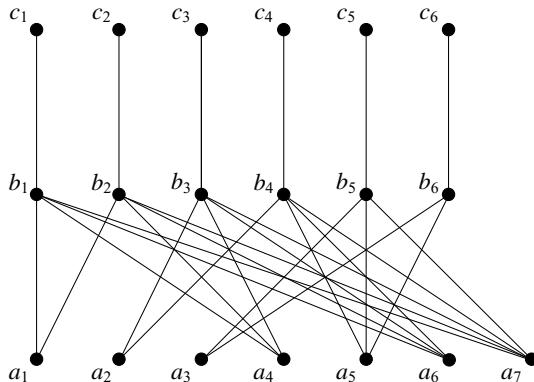


Abbildung 16.1.

Wenden wir unseren Algorithmus auf G_n an, so könnte er zunächst den Knoten $a_{A_n^{n-1}}$ wählen (da dieser maximalen Grad hat) und danach die Knoten $a_{A_n^{n-1}-1}, a_{A_n^{n-1}-2}, \dots, a_1$. Dann sind noch n paarweise disjunkte Kanten übrig, somit sind n weitere Knoten notwendig. Damit besteht die konstruierte Knotenüberdeckung aus $A_n^{n-1} + n$ Knoten, während die optimale Knotenüberdeckung $\{b_1, \dots, b_n\}$ die Größe n hat. \square

Es gibt jedoch 2-Approximationsalgorithmen für das MINIMUM-VERTEX-COVER-PROBLEM. Der einfachste stammt von Gavril (siehe Garey und Johnson [1979]): Bestimme ein inklusionsmaximales Matching M und nehme die Endknoten aller Kanten in M . Dies ist offensichtlich eine Knotenüberdeckung und enthält $2|M|$ Knoten. Da jede Knotenüberdeckung $|M|$ Knoten enthält (kein Knoten kann zwei Kanten in M überdecken), ist dies ein 2-Approximationsalgorithmus.

Diese Approximationsgüte ist die bestmögliche: Man braucht sich bloß einen aus vielen paarweise disjunkten Kanten bestehenden Graphen vorzustellen. Es mag überraschen, dass der obige Algorithmus der beste bekannte Approximationsalgorithmus für das MINIMUM-VERTEX-COVER-PROBLEM ist. Später werden wir zeigen: Gilt $P \neq NP$, so gibt es eine Zahl $k > 1$ mit der Eigenschaft, dass es keinen k -Approximationsalgorithmus gibt (Satz 16.46). Tatsächlich gibt es keinen 1,36-Approximationsalgorithmus, sofern $P \neq NP$ (Dinur und Safra [2002]). Siehe auch Khot und Regev [2008].

Wenigstens kann Gavril's Algorithmus auf den gewichteten Fall erweitert werden. Hier stellen wir den Algorithmus von Bar-Yehuda und Even [1981] vor, der auf das allgemeine MINIMUM-WEIGHT-SET-COVER-PROBLEM angewendet werden kann:

BAR-YEHUDA-EVEN-ALGORITHMUS

Input: Ein Mengensystem (U, \mathcal{S}) mit $\bigcup_{S \in \mathcal{S}} S = U$, Gewichte $c : \mathcal{S} \rightarrow \mathbb{R}_+$.
Output: Eine Überdeckung \mathcal{R} von (U, \mathcal{S}) .

-
- ① Setze $\mathcal{R} := \emptyset$ und $W := \emptyset$. Setze $y(e) := 0$ für alle $e \in U$.
 Setze $c'(S) := c(S)$ für alle $S \in \mathcal{S}$.
 - ② **While** $W \neq U$ **do**:
 - Wähle ein Element $e \in U \setminus W$.
 - Sei $R \in \mathcal{S}$ mit $e \in R$ und $c'(R)$ minimal. Setze $y(e) := c'(R)$.
 - Setze $c'(S) := c'(S) - y(e)$ für alle $S \in \mathcal{S}$ mit $e \in S$.
 - Setze $\mathcal{R} := \mathcal{R} \cup \{R\}$ und $W := W \cup R$.
-

Satz 16.5. (Bar-Yehuda und Even [1981]) *Für eine gegebene Instanz (U, \mathcal{S}, c) des MINIMUM-WEIGHT-SET-COVER-PROBLEMS findet der BAR-YEHUDA-EVEN-ALGORITHMUS eine Überdeckung, deren Gewicht höchstens gleich $p \text{OPT}(U, \mathcal{S}, c)$ ist, wobei $p := \max_{e \in U} |\{S \in \mathcal{S} : e \in S\}|$.*

Beweis: Das MINIMUM-WEIGHT-SET-COVER-PROBLEM kann als ganzzahliges LP geschrieben werden:

$$\min \left\{ cx : Ax \geq \mathbf{1}, x \in \{0, 1\}^{\mathcal{S}} \right\},$$

wobei die Zeilen der Matrix A den Elementen von U entsprechen und die Spalten von A die Inzidenzvektoren der Mengen in \mathcal{S} sind. Der optimale Zielfunktionswert der LP-Relaxierung

$$\min \{cx : Ax \geq \mathbf{1}, x \geq 0\}$$

ist eine untere Schranke für $\text{OPT}(U, \mathcal{S}, c)$ (das Weglassen der Nebenbedingungen $x \leq \mathbf{1}$ verändert nicht den optimalen Zielfunktionswert dieses LP). Nach Proposition 3.13 folgt somit, dass der optimale Zielfunktionswert des dualen LP

$$\max\{y\mathbf{1} : yA \leq c, y \geq 0\}$$

auch eine untere Schranke für $\text{OPT}(U, \mathcal{S}, c)$ ist.

Beachte nun, dass zu jedem Zeitpunkt des Algorithmus $c'(S) \geq 0$ für alle $S \in \mathcal{S}$. Sei \bar{y} der Vektor y bei Terminierung. Dann gilt $\bar{y} \geq 0$ und $\sum_{e \in S} \bar{y}(e) \leq c(S)$ für alle $S \in \mathcal{S}$, d. h. \bar{y} ist eine zulässige Lösung des dualen LP und

$$\bar{y} \mathbb{1} \leq \max\{y \mathbb{1} : yA \leq c, y \geq 0\} \leq \text{OPT}(U, \mathcal{S}, c).$$

Beachte schließlich, dass

$$\begin{aligned} c(\mathcal{R}) &= \sum_{R \in \mathcal{R}} c(R) \\ &= \sum_{R \in \mathcal{R}} \sum_{e \in R} \bar{y}(e) \\ &\leq \sum_{e \in U} p\bar{y}(e) \\ &= p\bar{y} \mathbb{1} \\ &\leq p \text{OPT}(U, \mathcal{S}, c). \end{aligned}$$

□

Da wir im Knotenüberdeckungsfall $p = 2$ haben, ist dies ein 2-Approximationsalgorithmus für das MINIMUM-WEIGHT-VERTEX-COVER-PROBLEM. Der erste 2-Approximationsalgorithmus stammt von Hochbaum [1982]. Sie hat vorgeschlagen, dass man im obigen Beweis eine optimale Lösung y des dualen LP finde und dann alle Mengen S mit $\sum_{e \in S} y(e) = c(S)$ nehme. Alternativ könnte man auch eine optimale Lösung x des primalen LP finden und dann alle Mengen S mit $x_S \geq \frac{1}{p}$ nehmen.

Der Vorteil des BAR-YEHUDA-EVEN-ALGORITHMUS ist, dass er keinen expliziten Gebrauch von der linearen Optimierung macht. In der Tat kann er leicht in $O(|\sum_{S \in \mathcal{S}} |S||)$ -Zeit implementiert werden. Er ist unser erstes Beispiel eines primal-dualen Approximationsalgorithmus; kompliziertere Beispiele werden wir in den Abschnitten 20.5 und 22.3 kennen lernen.

16.2 Das Max-Cut-Problem

In diesem Abschnitt betrachten wir ein weiteres grundlegendes Problem:

MAXIMUM-WEIGHT-CUT-PROBLEM

Instanz: Ein ungerichteter Graph G und Gewichte $c : E(G) \rightarrow \mathbb{R}_+$.

Aufgabe: Bestimme einen Schnitt in G mit maximalem Gesamtgewicht.

Dieses Problem wird oft einfach kurz MAX-CUT genannt. Im Gegensatz zu Schnitten minimalen Gewichtes, die wir in Abschnitt 8.7 besprochen haben, ist dies ein schwieriges Problem. Es ist stark NP -schwer; sogar der Spezialfall mit $c \equiv 1$ (das MAXIMUM-CUT-PROBLEM) ist NP -schwer:

Satz 16.6. (Garey, Johnson und Stockmeyer [1976]) Das MAXIMUM-CUT-PROBLEM ist NP-schwer.

Beweis: Der Beweis erfolgt mittels Reduktion von MAX-2SAT (siehe Satz 15.37). Für eine gegebene Instanz von MAX-2SAT mit n Variablen und m Klauseln konstruieren wir einen Graphen G , dessen Knoten die Literale und ein weiterer Knoten z sind. Für jede Variable x fügen wir $3m$ parallele Kanten zwischen x und \bar{x} hinzu. Für jede Klausel $\{\lambda, \lambda'\}$ fügen wir drei Kanten $\{\lambda, \lambda'\}$, $\{\lambda, z\}$ und $\{\lambda', z\}$ hinzu. Somit hat G $2n + 1$ Knoten und $3m(n + 1)$ Kanten.

Wir behaupten nun: Die maximale Kardinalität eines Schnittes in G ist $3mn + 2t$, wobei t die maximale Anzahl von Klauseln ist, die von irgendeiner Wahrheitsbelegung erfüllt werden. Zum Beweis betrachten wir eine t Klauseln erfüllende Wahrheitsbelegung, und es sei X die Menge der auf *true* gesetzten Literale. Dann gilt $|\delta_G(X)| = 3mn + 2t$. Umgekehrt: Gibt es eine Menge $X \subseteq V(G)$ mit $|\delta_G(X)| \geq 3mn + a$, so können wir o. B. d. A. annehmen, dass $z \notin X$ (anderenfalls ersetzen wir X durch $V(G) \setminus X$), und dass $|X \cap \{x, \bar{x}\}| = 1$ für jede Variable x (anderenfalls ersetzen wir X durch $X \triangle \{x\}$ und vergrößern den Schnitt). Somit können wir alle Literale in X auf *true* setzen und erhalten damit eine Wahrheitsbelegung, die mindestens $\frac{a}{2}$ Klauseln erfüllt. \square

Ein 2-Approximationsalgorithmus für das MAXIMUM-WEIGHT-CUT-PROBLEM lässt sich leicht finden: Ist $V(G) = \{v_1, \dots, v_n\}$, so beginne mit $X := \{v_1\}$ und füge v_i für $i = 3, \dots, n$ zu X hinzu, falls $\sum_{e \in E(v_i, \{v_1, \dots, v_{i-1}\} \cap X)} c(e) < \sum_{e \in E(v_i, \{v_1, \dots, v_{i-1}\} \setminus X)} c(e)$. (Die recht einfache Analyse dieses Algorithmus bildet den Inhalt von Aufgabe 9.)

Eine lange Zeit war kein besserer Approximationsalgorithmus bekannt. Dann fanden Goemans und Williamson [1995] einen viel besseren, der semidefinite Optimierung benutzt; der Rest dieses Abschnitts basiert auf deren Arbeit.

Sei G ein ungerichteter Graph und $c : E(G) \rightarrow \mathbb{R}_+$. O. B. d. A. können wir annehmen, dass $V(G) = \{1, \dots, n\}$. Für alle $1 \leq i, j \leq n$ sei $c_{ij} := c(\{i, j\})$ für $\{i, j\} \in E(G)$ und $c_{ij} := 0$ sonst. Dann lautet das MAXIMUM-WEIGHT-CUT-PROBLEM: Bestimme eine $\sum_{i \in S, j \in \{1, \dots, n\} \setminus S} c_{ij}$ maximierende Teilmenge $S \subseteq \{1, \dots, n\}$. Drückt man S mittels $y \in \{-1, 1\}^n$ aus, mit $y_i = 1$ genau dann, wenn $i \in S$, so können wir dieses Problem wie folgt formulieren:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{1 \leq i < j \leq n} c_{ij}(1 - y_i y_j) \\ \text{bzgl.} \quad & y_i \in \{-1, 1\} \quad (i = 1, \dots, n). \end{aligned}$$

Die Variablen y_i können als eindimensionale Vektoren der Norm 1 aufgefasst werden. Relaxieren wir sie zu mehrdimensionalen Vektoren mit euklidischer Norm 1, so erhalten wir ein äußerst interessantes Problem:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{1 \leq i < j \leq n} c_{ij}(1 - y_i^\top y_j) \\ \text{bzgl.} \quad & y_i \in \mathcal{S}_m \quad (i = 1, \dots, n), \end{aligned} \tag{16.1}$$

wobei $m \in \mathbb{N}$ und $\mathcal{S}_m = \{x \in \mathbb{R}^m : \|x\|_2 = 1\}$ die Einheitssphäre in \mathbb{R}^m ist. Für das Dreieck ($n = 3$, $c_{12} = c_{13} = c_{23} = 1$) z. B. wird der optimale Zielfunktionswert in den auf der Einheitssphäre in \mathbb{R}^2 liegenden drei Ecken eines gleichseitigen Dreiecks angenommen, z. B. den drei Punkten $y_1 = (0, -1)$, $y_2 = (-\frac{\sqrt{3}}{2}, \frac{1}{2})$ und $y_3 = (\frac{\sqrt{3}}{2}, \frac{1}{2})$. Dies ergibt den optimalen Zielfunktionswert $\frac{9}{4}$, im Gegensatz zu dem maximalen Gewicht eines Schnittes, welches 2 beträgt. Die interessante Tatsache ist jedoch, dass wir (16.1) in polynomieller Zeit fast optimal lösen können.

Dazu benutzen wir den Trick, nicht die Variablen y_i direkt zu betrachten, auch nicht deren Dimension. Stattdessen betrachten wir die $n \times n$ -Matrix $(y_i^\top y_j)_{i,j=1,\dots,n}$. Da eine Matrix X genau dann symmetrisch und positiv semidefinit ist, wenn es eine Matrix B gibt, so dass $X = B^\top B$, haben wir die äquivalente Formulierung:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{1 \leq i < j \leq n} c_{ij}(1 - x_{ij}) \\ \text{bzgl.} \quad & x_{ii} = 1 \quad (i = 1, \dots, n) \\ & X = (x_{ij})_{1 \leq i, j \leq n} \text{ symmetrisch und positiv semidefinit.} \end{aligned} \tag{16.2}$$

Aus einer Lösung von (16.2) können wir eine Lösung von (16.1) mit $m \leq n$ und fast demselben Zielfunktionswert mittels Cholesky-Faktorisierung in $O(n^3)$ -Zeit erhalten (wir müssen einen beliebig kleinen Rundungsfehler akzeptieren; siehe Aufgabe 6, Kapitel 4).

Problem (16.2) ist ein sogenanntes relaxiertes semidefinites Programm. Es kann in polynomieller Zeit approximativ gelöst werden, und zwar mit der ELLIPSOID-METHODE unter Anwendung von Satz 4.19, wie wir jetzt zeigen werden. Beachte zunächst, dass wir eine lineare Zielfunktion über der konvexen Menge

$$P := \left\{ X = (x_{ij})_{1 \leq i, j \leq n} \in \mathbb{R}^{n \times n} : X \text{ symmetrisch und positiv semidefinit, } x_{ii} = 1 \text{ (} i = 1, \dots, n \text{)} \right\}$$

optimieren. Die Projektion von P auf die $\frac{n^2-n}{2}$ freien Variablen ergibt

$$P' := \left\{ (x_{ij})_{1 \leq i < j \leq n} : (x_{ij})_{1 \leq i, j \leq n} \in P \text{ wobei } x_{ii} := 1 \text{ und } x_{ji} := x_{ij} \text{ für } i < j \right\}.$$

Beachte, dass weder P noch P' ein Polyeder ist. Es ist P' jedoch konvex, beschränkt und volldimensional:

Proposition 16.7. *Die Menge P' ist konvex. Ferner ist $B(0, \frac{1}{n}) \subseteq P' \subseteq B(0, n)$.*

Beweis: Die Konvexität folgt aus der einfachen Tatsache, dass Konvexitätskombinationen positiv semidefiniten Matrizen wieder positiv semidefinit sind.

Zum Beweis der ersten Inklusion: Beachte, dass für eine symmetrische $n \times n$ -Matrix X , deren Elemente auf der Hauptdiagonale alle gleich 1 und deren weitere Elemente alle im Betrage höchstens gleich $\frac{1}{n}$ sind, für jedes $d \in \mathbb{R}^n$ Folgendes gilt:

$$\begin{aligned}
d^\top X d &= \sum_{i,j=1}^n x_{ij} d_i d_j \\
&\geq \frac{1}{2n-2} \sum_{i \neq j} (x_{ii} d_i^2 + x_{jj} d_j^2 - (2n-2)|x_{ij}| |d_i d_j|) \\
&\geq \frac{1}{2n-2} \sum_{i \neq j} (d_i^2 + d_j^2 - 2|d_i d_j|) \\
&= \frac{1}{2n-2} \sum_{i \neq j} (|d_i| - |d_j|)^2 \\
&\geq 0,
\end{aligned}$$

d. h. X ist positiv semidefinit.

Zum Beweis der zweiten Inklusion: Beachte, dass alle nicht auf der Hauptdiagonale liegenden Elemente einer Matrix in P im Betrage höchstens gleich 1 sind. Somit ist die euklidische Norm des Vektors der Elemente der Dreiecksmatrix oberhalb der Hauptdiagonale höchstens gleich n . \square

Es bleibt zu zeigen, dass das SEPARATIONS-PROBLEM für P' in polynomieller Zeit gelöst werden kann. Dies erreicht man mittels GAUSS-ELIMINATION:

Satz 16.8. Für eine gegebene symmetrische Matrix $X \in \mathbb{Q}^{n \times n}$ kann man in polynomieller Zeit entscheiden, ob X positiv semidefinit ist, und einen Vektor $d \in \mathbb{Q}^n$ mit $d^\top X d < 0$ finden, falls es einen solchen gibt.

Beweis: Ist $x_{nn} < 0$, so setzen wir $d := (0, \dots, 0, 1)$, womit $d^\top X d < 0$ folgt. Ist $x_{nn} = 0$ und $x_{nj} \neq 0$ für irgendein $j < n$, so definieren wir d folgendermaßen: $d_j := -1$, $d_n := \frac{x_{jj}}{2x_{nj}} + x_{nj}$ und $d_i := 0$ für $i \in \{1, \dots, n-1\} \setminus \{j\}$. Dann folgt $d^\top X d = x_{jj} - 2x_{nj}(\frac{x_{jj}}{2x_{nj}} + x_{nj}) = -2(x_{nj})^2 < 0$, womit wiederum bewiesen ist, dass X nicht positiv semidefinit ist.

In den übrigen Fällen reduzieren wir die Dimension. Ist $x_{nj} = 0$ für alle j , so können wir die letzte Zeile und die letzte Spalte entfernen: Es ist X genau dann positiv semidefinit, wenn $X' := (x_{ij})_{i,j=1,\dots,n-1}$ positiv semidefinit ist. Ferner folgt: Gilt $c^\top X' c < 0$ für ein $c \in \mathbb{Q}^{n-1}$, so setzen wir $d := \begin{pmatrix} c \\ 0 \end{pmatrix}$ und haben $d^\top X d < 0$.

Also setzen wir nun voraus, dass $x_{nn} > 0$, und betrachten $X' := (x_{ij} - \frac{x_{ni}x_{nj}}{x_{nn}})_{i,j=1,\dots,n-1}$. Dieser Schritt entspricht einer Iteration der GAUSS-ELIMINATION. Beachte, dass X' genau dann positiv semidefinit ist, wenn X positiv semidefinit ist.

Für einen Vektor $c \in \mathbb{Q}^{n-1}$ mit $c^\top X' c < 0$ setzen wir $d := (-\frac{1}{x_{nn}} \sum_{i=1}^{n-1} c_i x_{ni})$. Dann folgt

$$\begin{aligned}
d^\top X d &= \sum_{i,j=1}^{n-1} d_i \left(x'_{ij} + \frac{x_{ni}}{x_{nn}} x_{nj} \right) d_j + 2 \sum_{j=1}^{n-1} d_n x_{nj} d_j + d_n^2 x_{nn} \\
&= c^\top X' c + \sum_{i,j=1}^{n-1} c_i \frac{x_{ni} x_{nj}}{x_{nn}} c_j (1 - 2 + 1) \\
&= c^\top X' c \\
&< 0.
\end{aligned}$$

Dies definiert einen polynomiellen Algorithmus. Um zu sehen, dass die bei der Berechnung von d auftretenden Zahlen nicht zu groß werden, seien $X^{(n)}, X^{(n-1)}, \dots, X^{(k)}$ mit $X^{(i)} \in \mathbb{Q}^{i \times i}$ die betrachteten Matrizen. Angenommen, in der $(n+1-k)$ -ten Iteration sehen wir, dass die Matrix $X^{(k)} = (y_{ij})_{i,j=1,\dots,k}$ nicht positiv semidefinit ist (d. h. $y_{kk} < 0$ oder $y_{kk} = 0$ und $y_{kj} \neq 0$ für ein $j < k$). Dann haben wir einen Vektor $c \in \mathbb{Q}^k$ mit $c^\top X^{(k)} c < 0$ und $\text{size}(c) \leq 2 \text{size}(X^{(k)})$. Nun können wir einen Vektor $d \in \mathbb{Q}^n$ mit $d^\top X d < 0$ konstruieren wie oben angegeben; beachte, dass d eine Lösung des linearen Gleichungssystems $Md = \begin{pmatrix} c \\ 0 \end{pmatrix}$ ist, wobei die j -te Zeile von M folgendermaßen lautet:

- der j -te Einheitsvektor, falls $j \leq k$,
- der j -te Einheitsvektor, falls $j > k$ und die j -te Zeile von $X^{(j)}$ gleich Null ist,
- die j -te Zeile von $X^{(j)}$ und danach Nullen, in den übrigen Fällen.

Nach Satz 4.4 haben wir sodann $\text{size}(d) \leq 4n(\text{size}(M) + \text{size}(c))$, und dies ist polynomiell nach Satz 4.10. \square

Korollar 16.9. Das SEPARATIONS-PROBLEM für P' kann in polynomieller Zeit gelöst werden.

Beweis: Sei $(y_{ij})_{1 \leq i < j \leq n}$ gegeben, und sei $Y = (y_{ij})_{1 \leq i,j \leq n}$ die symmetrische Matrix mit $y_{ii} = 1$ für alle i und $y_{ji} := y_{ij}$ für $i < j$. Nun wenden wir Satz 16.8 an. Ist Y positiv semidefinit, so sind wir fertig. Andernfalls bestimmen wir einen Vektor $d \in \mathbb{Q}^n$ mit $d^\top Y d < 0$. Dann gilt $-\sum_{i=1}^n d_i^2 > d^\top Y d - \sum_{i=1}^n d_i^2 = \sum_{1 \leq i < j \leq n} 2d_i d_j y_{ij}$. Aber $\sum_{1 \leq i < j \leq n} 2d_i d_j z_{ij} \geq -\sum_{i=1}^n d_i^2$ für alle $z \in P'$. Somit liefert $(d_i d_j)_{1 \leq i < j \leq n}$ eine trennende Hyperebene. \square

Wir haben nun das Resultat:

Satz 16.10. Für jede Instanz des MAXIMUM-WEIGHT-CUT-PROBLEMS können wir eine Matrix $Y = (y_{ij})_{1 \leq i,j \leq n} \in P$ mit

$$\sum_{1 \leq i < j \leq n} c_{ij} (1 - y_{ij}) \geq \max \left\{ \sum_{1 \leq i < j \leq n} c_{ij} (1 - x_{ij}) : (x_{ij})_{1 \leq i,j \leq n} \in P \right\} - \epsilon$$

in Zeit polynomiell in n , $\text{size}((c_{ij})_{1 \leq i < j \leq n})$ und $\text{size}(\epsilon)$ bestimmen.

Beweis: Wir wenden Satz 4.19 an, unter Benutzung von Proposition 16.7 und Korollar 16.9. \square

Semidefinite Programme wie (16.2) können auch mit Innere-Punkte-Algorithmen approximativ gelöst werden. Diese sind effizienter als die ELLIPSOIDMETHODE; Details hierzu findet man bei Alizadeh [1995]. Ein kombinatorischer Algorithmus, der noch schneller ist, wurde von Arora und Kale [2016] vorgeschlagen.

Aus einer fast optimalen Lösung für (16.2) können wir, wie bereits erwähnt wurde, mittels Cholesky-Faktorisierung eine Lösung für (16.1) mit fast demselben Zielfunktionswert ableiten. Diese Lösung besteht aus einer Menge von Vektoren $y_i \in \mathbb{R}^m$ ($i = 1, \dots, n$) für ein $m \leq n$. Da (16.1) eine Relaxierung unseres ursprünglichen Problems darstellt, ist der optimale Zielfunktionswert des ursprünglichen Problems höchstens gleich $\frac{1}{2} \sum_{1 \leq i < j \leq n} c_{ij}(1 - y_i^\top y_j) + \epsilon$.

Die Vektoren y_i liegen auf einer Einheitssphäre. Die Idee ist es nun, eine zufällig gewählte Hyperebene durch den Ursprung zu nehmen und S als die Menge derjenigen Indizes i zu definieren, für die die zugehörigen y_i alle auf einer Seite dieser Hyperebene liegen.

Eine zufällig gewählte Hyperebene durch den Ursprung wird durch einen zufällig gewählten Punkt auf der $(m - 1)$ -dimensionalen Sphäre gegeben. Dieser kann dadurch gewonnenen werden, dass man unabhängig m mal eine reelle Zahl aus den normalverteilten reellen Zahlen wählt. Dies kann wiederum durch Verwendung von unabhängigen gleichverteilten Zufallszahlen aus $[0, 1]$ erreicht werden. Details hierzu findet man bei Knuth [1969] (Abschnitt 3.4.1).

Der Algorithmus von Goemans und Williamson lautet nun wie folgt:

GOEMANS-WILLIAMSON-MAX-CUT-ALGORITHMUS

Input: Eine Zahl $n \in \mathbb{N}$, Zahlen $c_{ij} \geq 0$ für $1 \leq i < j \leq n$.

Output: Eine Menge $S \subseteq \{1, \dots, n\}$.

- ① Löse (16.2) approximativ; d. h. bestimme eine symmetrische positiv semidefinite Matrix $X = (x_{ij})_{1 \leq i, j \leq n}$ mit $x_{ii} = 1$ für $i = 1, \dots, n$, so dass $\sum_{1 \leq i < j \leq n} c_{ij}(1 - x_{ij}) \geq 0, 9995 \cdot \text{OPT}(16.2)$.
- ② Wende Cholesky-Faktorisierung auf X an, um Vektoren $y_1, \dots, y_n \in \mathbb{R}^m$ mit $m \leq n$ und $y_i^\top y_j \approx x_{ij}$ für alle $i, j \in \{1, \dots, n\}$ zu erhalten.
- ③ Wähle einen zufälligen Punkt a auf der Einheitssphäre $\{x \in \mathbb{R}^m : \|x\|_2 = 1\}$.
- ④ Setze $S := \{i \in \{1, \dots, n\} : a^\top y_i \geq 0\}$.

Satz 16.11. Der GOEMANS-WILLIAMSON-MAX-CUT-ALGORITHMUS läuft in polynomieller Zeit.

Beweis: Wir verweisen auf die vorangegangene Betrachtung. Der schwierigste Schritt, nämlich ①, kann nach Satz 16.10 in polynomieller Zeit gelöst werden. Hier können wir $\epsilon = 0,00025 \sum_{1 \leq i < j \leq n} c_{ij}$ wählen, denn $\frac{1}{2} \sum_{1 \leq i < j \leq n} c_{ij}$ ist eine untere Schranke für den optimalen Zielfunktionswert (gewonnen durch Zufallswahl von $S \subseteq \{1, \dots, n\}$) und somit auch für den optimalen Zielfunktionswert von (16.2). \square

Wir beweisen nun die Approximationsgüte:

Satz 16.12. (Goemans und Williamson [1995]) *Der GOEMANS-WILLIAMSON-MAX-CUT-ALGORITHMUS liefert als Output eine Menge S , für welche der Erwartungswert von $\sum_{i \in S, j \notin S} c_{ij}$ gleich mindestens $0,878$ mal dem größtmöglichen Wert ist.*

Beweis: Sei \mathcal{S}_m wiederum die Einheitssphäre in \mathbb{R}^m und $H(y) := \{x \in \mathcal{S}_m : x^\top y \geq 0\}$ die Hemisphäre mit dem Pol y für ein $y \in \mathcal{S}_m$. Für eine gegebene Teilmenge $A \subseteq \mathcal{S}_m$ sei $\mu(A) := \frac{\text{volume}(A)}{\text{volume}(\mathcal{S}_m)}$; damit wird ein Wahrscheinlichkeitsmaß auf \mathcal{S}_m definiert. Es ist $|S \cap \{i, j\}| = 1$ mit Wahrscheinlichkeit $\mu(H(y_i) \triangle H(y_j))$, wobei \triangle die symmetrische Differenz bedeutet. Beachte, dass $H(y_i) \triangle H(y_j)$ die Vereinigung zweier sphärischer Zweiecke ist, beide mit dem Winkel $\arccos(y_i^\top y_j)$. Da das Volumen zum Winkel proportional ist, haben wir $\mu(H(y_i) \triangle H(y_j)) = \frac{1}{\pi} \arccos(y_i^\top y_j)$.

Behauptung: $\frac{1}{\pi} \arccos \beta \geq 0,8785 \cdot \frac{1-\beta}{2}$ für alle $\beta \in [-1, 1]$.

Für $\beta = 1$ gilt Gleichheit. Mittels elementarer Differenzialrechnung gilt ferner

$$\min_{-1 \leq \beta < 1} \frac{\arccos \beta}{1 - \beta} = \min_{0 < \gamma \leq \pi} \frac{\gamma}{1 - \cos \gamma} = \frac{1}{\sin \gamma},$$

wobei γ' durch $\cos \gamma' + \gamma' \sin \gamma' = 1$ gegeben ist. Wir bekommen somit $2,3311 < \gamma' < 2,3312$ und $\frac{1}{\sin \gamma'} > \frac{1}{\sin 2,3311} > 1,38$. Da $\frac{1,38}{\pi} > \frac{0,8785}{2}$, folgt die Behauptung.

Für den Erwartungswert von $\sum_{i \in S, j \notin S} c_{ij}$ bekommen wir hiermit

$$\begin{aligned} \sum_{1 \leq i < j \leq n} c_{ij} \mu(H(y_i) \triangle H(y_j)) &= \sum_{1 \leq i < j \leq n} c_{ij} \frac{1}{\pi} \arccos(y_i^\top y_j) \\ &\geq 0,8785 \cdot \frac{1}{2} \sum_{1 \leq i < j \leq n} c_{ij} (1 - y_i^\top y_j) \\ &\approx 0,8785 \cdot \frac{1}{2} \sum_{1 \leq i < j \leq n} c_{ij} (1 - x_{ij}) \\ &\geq 0,8785 \cdot 0,9995 \cdot \text{OPT(16.2)} \\ &> 0,878 \cdot \text{OPT(16.2)} \\ &\geq 0,878 \cdot \max \left\{ \sum_{i \in S, j \notin S} c_{ij} : S \subseteq \{1, \dots, n\} \right\}. \end{aligned}$$

□

Folglich haben wir einen randomisierten Approximationsalgorithmus mit der Approximationsgüte $\frac{1}{0,878} < 1,139$. Mahajan und Ramesh [1999] haben gezeigt, wie man diesen Algorithmus derandomisieren kann und somit einen deterministischen 1,139-Approximationsalgorithmus erhält. Es gibt jedoch keinen 1,062-Approximationsalgorithmus, sofern $P \neq NP$ gilt (Håstad [2001], Papadimitriou und Yannakakis [1991]).

Für weitere interessante Zusammenhänge zwischen semidefiniter und kombinatorischer Optimierung siehe Lovász [2003].

16.3 Färbung

In diesem Abschnitt werden wir kurz auf zwei weitere bekannte Spezialfälle des MINIMUM-SET-COVER-PROBLEMS eingehen: die Partitionierung der Knotenmenge eines Graphen in stabile Mengen bzw. der Kantenmenge eines Graphen in Matchings.

Definition 16.13. Sei G ein ungerichteter Graph. Eine **Knotenfärbung** von G ist eine Abbildung $f : V(G) \rightarrow \mathbb{N}$ mit $f(v) \neq f(w)$ für alle $\{v, w\} \in E(G)$. Eine **Kantenfärbung** von G ist eine Abbildung $f : E(G) \rightarrow \mathbb{N}$ mit $f(e) \neq f(e')$ für alle $e, e' \in E(G)$ mit $e \neq e'$ und $e \cap e' \neq \emptyset$.

Die Zahl $f(v)$ bzw. $f(e)$ heißt die **Farbe** von v bzw. e . Anders ausgedrückt: Die Menge der gleichfarbigen (mit gleichem f -Wert) Knoten bzw. Kanten bildet eine stabile Menge bzw. ein Matching. Natürlich sind wir daran interessiert, möglichst wenige Farben zu benutzen:

KNOTENFÄRBUNGS-PROBLEM

Instanz: Ein ungerichteter Graph G .

Aufgabe: Bestimme eine Knotenfärbung $f : V(G) \rightarrow \{1, \dots, k\}$ von G mit minimalem k .

KANTENFÄRBUNGS-PROBLEM

Instanz: Ein ungerichteter Graph G .

Aufgabe: Bestimme eine Kantenfärbung $f : E(G) \rightarrow \{1, \dots, k\}$ von G mit minimalem k .

Die Reduzierung dieser Probleme auf das MINIMUM-SET-COVER-PROBLEM ist nicht sonderlich hilfreich: Für das KNOTENFÄRBUNGS-PROBLEM müssten wir alle inklusionsmaximalen stabilen Mengen auflisten und für das KANTENFÄRBUNGS-PROBLEM alle inklusionsmaximalen Matchings; in beiden Fällen müssten wir mit einer exponentiell wachsenden Anzahl rechnen.

Das Optimum des KNOTENFÄRBUNGS-PROBLEMS (d. h. die minimale Anzahl von Farben) heißt die **chromatische Zahl** des Graphen. Das Optimum des KANTENFÄRBUNGS-PROBLEMS heißt die **kantenchromatische Zahl** oder gelegentlich auch der **chromatische Index** des Graphen. Beide Färbungsprobleme sind *NP*-schwer:

Satz 16.14. Die folgenden Entscheidungsprobleme sind *NP*-vollständig:

- (a) (Holyer [1981]) Entscheide, ob ein gegebener einfacher Graph die kantenchromatische Zahl 3 hat.
- (b) (Stockmeyer [1973]) Entscheide, ob ein gegebener planarer Graph die chromatische Zahl 3 hat.

Diese Probleme bleiben sogar dann *NP*-schwer, wenn der maximale Grad des Graphen gleich 3 im Fall (a) und gleich 4 im Fall (b) ist.

Proposition 16.15. *Für einen gegebenen Graphen kann man in linearer Zeit entscheiden, ob die chromatische Zahl (bzw. die kantenchromatische Zahl) kleiner als 3 ist und falls ja, eine optimale Färbung finden.*

Beweis: Ein Graph hat chromatische Zahl 1 genau dann, wenn er keine Kanten besitzt. Definitionsgemäß sind die Graphen mit chromatischer Zahl höchstens gleich 2 gerade die bipartiten Graphen. Nach Proposition 2.27 können wir in linearer Zeit entscheiden, ob ein Graph bipartit ist und falls ja, eine Bipartition finden, d. h. eine Knotenfärbung mit zwei Farben.

Um zu entscheiden, ob die kantenchromatische Zahl eines Graphen G kleiner als 3 ist (und falls ja, eine optimale Kantenfärbung zu finden), betrachten wir einfach das KNOTENFÄRBUNGS-PROBLEM für den Kantengraphen von G . Dies ist offensichtlich äquivalent. \square

Für bipartite Graphen kann das KANTENFÄRBUNGS-PROBLEM auch gelöst werden:

Satz 16.16. (König [1916]) *Die kantenchromatische Zahl eines bipartiten Graphen G ist gleich dem maximalen Grad eines Knotens in G .*

Beweis: Der Beweis erfolgt mittels Induktion über $|E(G)|$. Sei G ein Graph mit maximalem Grad k und $e = \{v, w\}$ eine Kante. Nach Induktionsvoraussetzung hat $G - e$ eine Kantenfärbung f mit k Farben. Es gibt Farben $i, j \in \{1, \dots, k\}$ mit $f(e') \neq i$ für alle $e' \in \delta(v)$ und $f(e') \neq j$ für alle $e' \in \delta(w)$. Ist $i = j$, so sind wir fertig, da wir f auf G erweitern können, indem wir e die Farbe i geben.

Der Graph $H = (V(G), \{e' \in E(G) \setminus \{e\} : f(e') \in \{i, j\}\})$ hat maximalen Grad 2 und v hat höchstens Grad 1 in H . Betrachte den maximalen Weg P in H mit Endknoten v . Die Farben alternieren auf P , also ist der andere Endknoten von P auf keinen Fall w . Nun vertauschen wir die Farben i und j entlang P und erweitern die Kantenfärbung auf G , indem wir e die Farbe j geben. \square

Der maximale Grad eines Knotens ist offensichtlich eine untere Schranke für die kantenchromatische Zahl eines Graphen. Wie das Gegenbeispiel des Dreiecks K_3 zeigt, wird diese Schranke aber nicht immer erreicht. Der folgende Satz zeigt, wie man für einen gegebenen einfachen Graphen eine Kantenfärbung findet, die höchstens eine Farbe mehr als notwendig benutzt:

Satz 16.17. (Vizing [1964]) *Sei G ein ungerichteter einfacher Graph mit maximalem Grad k . Dann besitzt G eine Kantenfärbung mit höchstens $k + 1$ Farben, und eine solche Färbung kann in polynomieller Zeit gefunden werden.*

Beweis: Der Beweis erfolgt mittels Induktion über $|E(G)|$. Hat G gar keine Kanten, so ist die Aussage trivial. Andernfalls sei $e = \{x, y_0\}$ eine Kante; nach Induktionsvoraussetzung gibt es eine Kantenfärbung f von $G - e$ mit $k + 1$ Farben.

Für jeden Knoten v wählen wir eine Farbe $n(v) \in \{1, \dots, k+1\} \setminus \{f(w) : w \in \delta_{G-e}(v)\}$, d. h. eine Farbe, mit der keine mit v inzidente Kante gefärbt ist.

Beginnend mit y_0 , konstruieren wir nun eine maximale Folge y_0, y_1, \dots, y_t paarweise verschiedener Nachbarn von x , so dass $n(y_{i-1}) = f(\{x, y_i\})$ für $i = 1, \dots, t$.

Hat keine mit x inzidente Kante die Farbe $n(y_t)$, so bilden wir aus f eine Kantenfärbung f' von G , indem wir $f'(\{x, y_{i-1}\}) := f(\{x, y_i\})$ ($i = 1, \dots, t$) und $f'(\{x, y_t\}) := n(y_t)$ setzen. Andernfalls gibt es eine mit x inzidente Kante mit der Farbe $n(y_t)$; mit der Maximalität von t folgt $f(\{x, y_s\}) = n(y_t)$ für ein $s \in \{1, \dots, t-1\}$.

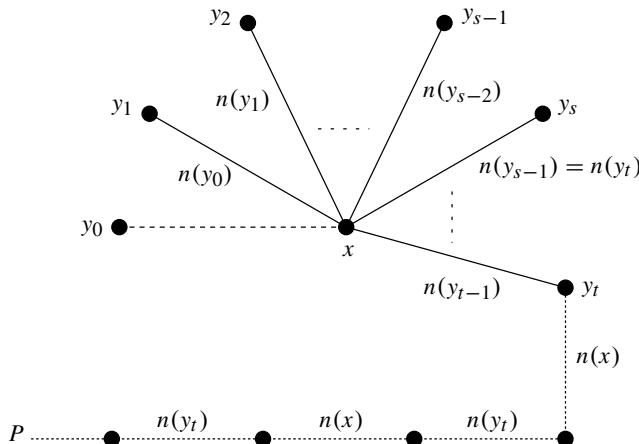


Abbildung 16.2.

Betrachte den maximalen Weg P , ausgehend von y_t , im Graphen $(V(G), \{e' \in E(G - e) : f(e') \in \{n(x), n(y_t)\}\})$ (dieser Graph hat maximalen Grad 2; siehe Abb. 16.2). Wir unterscheiden drei Fälle, in denen wir jeweils eine Kantenfärbung f' von G konstruieren.

Endet P in x , so ist $\{y_s, x\}$ die letzte Kante von P . Wir bilden f' aus f , indem wir die Farben $n(x)$ und $n(y_t)$ auf dem Weg P vertauschen und $f'(\{x, y_{i-1}\}) := f(\{x, y_i\})$ ($i = 1, \dots, s$) setzen.

Endet P in y_{s-1} , so hat die letzte Kante von P die Farbe $n(x)$, da keine mit y_{s-1} inzidente Kante die Farbe $n(y_t) = f(\{x, y_s\}) = n(y_{s-1})$ hat. Wir bilden f' aus f , indem wir die Farben $n(x)$ und $n(y_t)$ auf dem Weg P vertauschen und $f'(\{x, y_{i-1}\}) := f(\{x, y_i\})$ ($i = 1, \dots, s-1$) und $f'(\{x, y_{s-1}\}) := n(x)$ setzen.

Endet P weder in x noch in y_{s-1} , so können wir f' aus f dadurch bilden, dass wir die Farben $n(x)$ und $n(y_t)$ auf dem Weg P vertauschen und $f'(\{x, y_{i-1}\}) := f(\{x, y_i\})$ ($i = 1, \dots, t$) und $f'(\{x, y_t\}) := n(x)$ setzen. \square

Aus dem Satz von Vizing folgt die Existenz eines absoluten Approximationsalgorithmus für das KANTENFÄRBUNGS-PROBLEM in einfachen Graphen. Erlauben

wir parallele Kanten, so gilt die Aussage des Satzes von Vizing nicht mehr: Ersetzen wir jede Kante des Dreiecks K_3 durch r parallele Kanten, so erhalten wir einen $2r$ -regulären Graphen mit kantenchromatischer Zahl gleich $3r$.

Wir wenden uns nun dem KNOTENFÄRBUNGS-PROBLEM zu. Der maximale Grad eines Graphen ist hier eine obere Schranke für die chromatische Zahl:

Satz 16.18. *Sei G ein ungerichteter Graph mit maximalem Grad k . Dann besitzt G eine Knotenfärbung mit höchstens $k + 1$ Farben, und eine solche Färbung kann in linearer Zeit gefunden werden.*

Beweis: Der folgende GREEDY-FÄRBUNGSALGORITHMUS findet offensichtlich einen solche Knotenfärbung. \square

GREEDY-FÄRBUNGSALGORITHMUS

Input: Ein ungerichteter Graph G .

Output: Eine Knotenfärbung von G .

-
- ① Sei $V(G) = \{v_1, \dots, v_n\}$.
 - ② **For** $i := 1$ **to** n **do:**
Setze $f(v_i) := \min\{k \in \mathbb{N} : k \neq f(v_j) \text{ für alle } j < i \text{ mit } v_j \in \Gamma(v_i)\}$.
-

Für vollständige Graphen und für ungerade Kreise benötigt man offensichtlich $k+1$ Farben, wobei k der maximale Grad ist. Für alle anderen zusammenhängenden Graphen genügen k Farben, wie Brooks [1941] gezeigt hat. Der maximale Grad ist jedoch keine untere Schranke für die chromatische Zahl: Jeder Stern $K_{1,n}$ ($n \in \mathbb{N}$) hat die chromatische Zahl 2. Somit führen diese Resultate nicht zu einem Approximationsalgorithmus. In der Tat ist kein Algorithmus für das KNOTENFÄRBUNGS-PROBLEM mit einer vernünftigen Approximationsgüte für allgemeine Graphen bekannt; siehe Khanna, Linial und Safra [2000]. Zuckerman [2007] hat gezeigt: Gilt $P \neq NP$, so berechnet kein polynomieller Algorithmus die chromatische Zahl eines Graphen mit n Knoten bis auf den Faktor $n^{1-\epsilon}$ für ein beliebiges festes $\epsilon > 0$.

Held, Cook und Sewell [2012] haben gezeigt, wie sich gute untere Schranken für die chromatische Zahl berechnen lassen. Eine einfache untere Schranke stellt die maximale Größe einer Clique dar. Besitzt ein Graph G eine Clique der Größe k , so ist die chromatische Zahl von G offensichtlich mindestens gleich k . Wie das Beispiel eines Pentagons (Kreis der Länge 5) zeigt, kann die chromatische Zahl durchaus größer als die maximale Cliquengröße sein. In der Tat gibt es Graphen mit beliebig großer chromatischer Zahl, die kein Dreieck K_3 enthalten. Dies legt die folgende Definition nahe, die von Berge [1961,1962] stammt:

Definition 16.19. *Ein Graph G heißt perfekt, falls $\chi(H) = \omega(H)$ für jeden induzierten Teilgraphen H von G , wobei $\chi(H)$ die chromatische Zahl von H und $\omega(H)$ die maximale Kardinalität einer Clique in H ist.*

Es folgt sofort, dass das Entscheidungsproblem, ob ein gegebener perfekter Graph die chromatische Zahl k hat, eine gute Charakterisierung hat (in $NP \cap coNP$ ist). Einige Beispiele perfekter Graphen werden in Aufgabe 16 angegeben. Ein polynomieller Algorithmus zur Erkennung perfekter Graphen ist von Chudnovsky et al. [2005] gefunden worden.

Berge [1961] hat die Vermutung aufgestellt, dass ein Graph genau dann perfekt ist, wenn er weder einen ungeraden Kreis der Länge mindestens 5 noch den Komplementgraphen eines solchen Kreises als induzierten Teilgraphen enthält. Dieser sogenannte Strong-Perfect-Graph-Satz ist von Chudnovsky et al. [2006] bewiesen worden. Schon viel früher hatte Lovász [1972] die schwächere Aussage, dass ein Graph genau dann perfekt ist, wenn der Komplementgraph perfekt ist, bewiesen. Dieses Resultat ist unter dem Namen Weak-Perfect-Graph-Satz bekannt; um es zu beweisen, benötigen wir das folgende Lemma:

Lemma 16.20. *Sei G ein perfekter Graph und $x \in V(G)$. Dann ist der Graph $G' := (V(G) \dot{\cup} \{y\}, E(G) \dot{\cup} \{\{y, v\} : v \in \{x\} \cup \Gamma(x)\})$, der aus G durch Hinzufügen eines neuen mit x und allen Nachbarn von x mittels neuer Kanten verbundenen Knotens y hervorgeht, perfekt.*

Beweis: Der Beweis erfolgt mittels Induktion über $|V(G)|$. Der Fall $|V(G)| = 1$ ist trivial, da K_2 perfekt ist. Sei nun G ein perfekter Graph mit mindestens zwei Knoten. Sei $x \in V(G)$ und es gehe der Graph G' aus G durch Hinzufügen eines neuen mit x und allen Nachbarn von x mittels neuer Kanten verbundenen Knotens y hervor. Es genügt zu zeigen, dass $\omega(G') = \chi(G')$, da dies für echte Teilgraphen H von G' nach der Induktionsvoraussetzung gilt: Entweder ist H ein Teilgraph von G und somit perfekt, oder H geht aus einem echten Teilgraphen von G mittels Hinzufügen eines neuen Knotens y , wie oben beschrieben, hervor.

Da es ein Leichtes ist, G' mit $\chi(G)+1$ Farben zu färben, können wir annehmen, dass $\omega(G') = \omega(G)$. Dann ist x in keiner kardinalitätsmaximalen Clique von G . Sei f eine Knotenfärbung von G mit $\chi(G)$ Farben und setze $X := \{v \in V(G) : f(v) = f(x)\}$. Es folgt $\omega(G - X) = \chi(G - X) = \chi(G) - 1 = \omega(G) - 1$, somit gilt $\omega(G - (X \setminus \{x\})) = \omega(G) - 1$ (da x in keiner kardinalitätsmaximalen Clique von G ist). Da $(X \setminus \{x\}) \cup \{y\} = V(G') \setminus V(G - (X \setminus \{x\}))$ eine stabile Menge ist, folgt

$$\chi(G') = \chi(G - (X \setminus \{x\})) + 1 = \omega(G - (X \setminus \{x\})) + 1 = \omega(G) = \omega(G').$$

□

Satz 16.21. (Lovász [1972], Fulkerson [1972], Chvátal [1975]) *Gegeben sei ein einfacher Graph G . Dann sind die folgenden drei Aussagen äquivalent:*

- (a) *Es ist G ein perfekter Graph.*
- (b) *Der Komplementgraph von G ist perfekt.*
- (c) *Das Stabile-Mengen-Polytop, d. h. die konvexe Hülle der Inzidenzvektoren der stabilen Mengen von G , wird gegeben durch:*

$$\left\{ x \in \mathbb{R}_+^{V(G)} : \sum_{v \in S} x_v \leq 1 \text{ für alle Cliquen } S \text{ in } G \right\}. \quad (16.3)$$

Beweis: Wir zeigen, dass (a) \Rightarrow (c) \Rightarrow (b). Dies genügt: Die Anwendung von (a) \Rightarrow (b) auf den Komplementgraphen von G ergibt (b) \Rightarrow (a).

(a) \Rightarrow (c): Offensichtlich ist das Stabile-Mengen-Polytop in (16.3) enthalten. Zum Beweis der umgekehrten Inklusion, sei x ein rationaler Vektor in dem Polytop (16.3); wir können $x_v = \frac{p_v}{q}$ schreiben, wobei $q \in \mathbb{N}$ und $p_v \in \mathbb{Z}_+$ für $v \in V(G)$. Nun ersetzen wir jeden Knoten v durch eine Clique der Größe p_v ; d. h. wir betrachten den Graphen G' gegeben durch:

$$\begin{aligned} V(G') &:= \{(v, i) : v \in V(G), 1 \leq i \leq p_v\}, \\ E(G') &:= \{\{(v, i), (v, j)\} : v \in V(G), 1 \leq i < j \leq p_v\} \cup \\ &\quad \{\{(v, i), (w, j)\} : \{v, w\} \in E(G), 1 \leq i \leq p_v, 1 \leq j \leq p_w\}. \end{aligned}$$

Nach Lemma 16.20 ist G' perfekt. Für eine beliebige Clique X' in G' setzen wir $X := \{v \in V(G) : (v, i) \in X' \text{ für ein } i\}$, d. h. X ist die Projektion von X' auf G (und ist wieder eine Clique); es folgt

$$|X'| \leq \sum_{v \in X} p_v = q \sum_{v \in X} x_v \leq q.$$

Also ist $\omega(G') \leq q$. Da G' perfekt ist, hat G' somit eine Kantenfärbung f mit höchstens q Farben. Für $v \in V(G)$ und $i = 1, \dots, q$ setzen wir $a_{i,v} := 1$, falls $f((v, j)) = i$ für ein j , und $a_{i,v} := 0$ sonst. Dann ist $\sum_{i=1}^q a_{i,v} = p_v$ für alle $v \in V(G)$, und somit ist

$$x = \left(\frac{p_v}{q} \right)_{v \in V(G)} = \frac{1}{q} \sum_{i=1}^q a_i$$

eine Konvexitätskombination von Inzidenzvektoren stabiler Mengen, wobei $a_i = (a_{i,v})_{v \in V(G)}$.

(c) \Rightarrow (b): Wir zeigen mittels Induktion über $|V(G)|$, dass der Komplementgraph von G perfekt ist, falls (16.3) ganzzahlig ist. Da Graphen mit weniger als drei Knoten perfekt sind, sei G ein Graph mit $|V(G)| \geq 3$, und sei (16.3) ganzzahlig.

Wir müssen zeigen, dass die Knotenmenge eines jeden induzierten Teilgraphen H von G in $\alpha(H)$ Cliquen partitioniert werden kann, wobei $\alpha(H)$ die maximale Kardinalität einer stabilen Menge in H ist. Für echte Teilgraphen H folgt dies mit der Induktionsvoraussetzung, da (nach Satz 5.14) jede Seitenfläche des ganzzahligen Polytops (16.3) ganzzahlig ist, insbesondere die durch die stützenden Hyperebenen $x_v = 0$ ($v \in V(G) \setminus V(H)$) definierte Seitenfläche.

Somit bleibt zu zeigen, dass $V(G)$ in $\alpha(G)$ Cliquen partitioniert werden kann. Die Gleichung $\mathbf{1}x = \alpha(G)$ definiert eine stützende Hyperebene von (16.3), also ist

$$\left\{ x \in \mathbb{R}_+^{V(G)} : \sum_{v \in S} x_v \leq 1 \text{ für alle Cliques } S \text{ in } G, \sum_{v \in V(G)} x_v = \alpha(G) \right\} \quad (16.4)$$

eine Seitenfläche von (16.3). Diese Seitenfläche ist in einigen Facetten enthalten, die aber nicht alle von der Form $\{x \in (16.3) : x_v = 0\}$ für ein v sind (denn sonst wäre der Ursprung in deren Durchschnitt). Somit gibt es eine Clique S in G mit $\sum_{v \in S} x_v = 1$ für alle x in (16.4). Diese Clique S hat dann nichtleeren Durchschnitt mit jeder kardinalitätsmaximalen stabilen Menge von G . Mit der Induktionsvoraussetzung folgt nun, dass die Knotenmenge von $G - S$ in $\alpha(G - S) = \alpha(G) - 1$ Cliques partitioniert werden kann. Mittels Hinzufügen von S ist der Beweis abgeschlossen. \square

Dieser Beweis stammt von Lovász [1979b]. Das (16.3) definierende lineare Ungleichungssystem ist in der Tat TDI für perfekte Graphen (Aufgabe 17). Mit etwas mehr Aufwand kann man beweisen, dass die drei folgenden Probleme für perfekte Graphen in streng polynomieller Zeit gelöst werden können: das KNOTENFÄRBUNGS-PROBLEM, das MAXIMUM-WEIGHT-STABLE-SET-PROBLEM und das MAXIMUM-WEIGHT-CLIQUE-PROBLEM. Obwohl diese Probleme für allgemeine Graphen NP-schwer sind (Satz 15.23, Korollar 15.24, Satz 16.14(b)), gibt es eine Zahl (die von Lovász [1979a] eingeführte sogenannte Theta-Funktion des Komplementgraphen), die immer zwischen der maximalen Cliquengröße und der chromatischen Zahl liegt und für allgemeine Graphen mit der ELLIPSOIDMETHODE in polynomieller Zeit berechnet werden kann. Die Details sind etwas kompliziert; siehe Grötschel, Lovász und Schrijver [1988].

Eines der bekanntesten Probleme der Graphentheorie ist seit langem das Vierfarbenproblem: Entspricht es der Wahrheit, dass man jede planare Landkarte mit genau vier Farben färben kann, so dass keine zwei Länder mit gemeinsamer Grenze gleichfarbt sind? Nehmen wir die Länder als Gebiete und betrachten dann das planare Dual, so erhalten wir die äquivalente Formulierung: Entspricht es der Wahrheit, dass jeder planare Graph eine Knotenfärbung mit höchstens vier Farben hat? Appel und Haken [1977] und Appel, Haken und Koch [1977] haben diese Aussage positiv beantworten können; mit anderen Worten, für jeden planaren Graphen ist die chromatische Zahl höchstens gleich 4. Von Robertson et al. [1997] stammt ein einfacherer Beweis dieses Vierfarbensatzes, der jedoch auch auf einer Fallprüfung mittels Computer basiert. Hier beweisen wir ein schwächeres Resultat, den sogenannten Fünf-Farben-Satz:

Satz 16.22. (Heawood [1890]) *Jeder planare Graph besitzt eine Knotenfärbung mit höchstens fünf Farben, und eine solche Färbung kann in polynomieller Zeit gefunden werden.*

Beweis: Der Beweis erfolgt mittels Induktion über $|V(G)|$. Wir können annehmen, dass G einfach ist, und betrachten eine beliebige feste planare Einbettung $\Phi = (\psi, (J_e)_{e \in E(G)})$ von G . Nach Korollar 2.33 hat G einen Knoten v mit Grad kleiner oder gleich 5. Mit der Induktionsvoraussetzung besitzt $G - v$ eine Knotenfärbung f mit höchstens 5 Farben. Wir können annehmen, dass der Grad von v

gleich 5 ist und dass alle Nachbarn von v verschieden gefärbt sind, sonst könnten wir die Färbung leicht auf G erweitern.

Seien w_1, w_2, w_3, w_4, w_5 die Nachbarn von v , geordnet in der zyklischen Reihenfolge der in v beginnenden polygonalen Streckenzüge $J_{\{v, w_i\}}$.

Zunächst behaupten wir, dass es keine knotendisjunkten Wege P von w_1 nach w_3 und Q von w_2 nach w_4 in $G - v$ gibt. Um dies zu beweisen, sei P ein w_1 - w_3 -Weg in $G - v$ und C der aus P und den Kanten $\{v, w_1\}, \{v, w_3\}$ bestehende Kreis in G . Nach Satz 2.30 spaltet sich die Punktmenge $\mathbb{R}^2 \setminus \bigcup_{e \in E(C)} J_e$ in zwei zusammenhängende Gebiete auf, und v liegt auf deren gemeinsamem Rand. Somit liegen w_2 und w_4 in verschiedenen Gebieten dieser Punktmenge und folglich besitzt jeder w_2 - w_4 -Weg in $G - v$ einen Knoten von C .

Sei nun X die w_1 enthaltende Zusammenhangskomponente des Graphen $G[\{x \in V(G) \setminus \{v\} : f(x) \in \{f(w_1), f(w_3)\}\}]$. Liegt w_3 nicht in X , so können wir die beiden Farben $f(w_1)$ und $f(w_3)$ in X vertauschen und die resultierende Färbung auf G erweitern, indem wir v mit der ursprünglichen Farbe von w_1 färben. Somit können wir annehmen, dass es einen w_1 - w_3 -Weg P gibt, dessen Knoten nur mit den Farben $f(w_1)$ und $f(w_3)$ gefärbt sind.

Gehen wir analog mit w_2 und w_4 vor, so folgt: Gibt es keinen w_2 - w_4 -Weg Q , dessen Knoten nur mit den Farben $f(w_2)$ und $f(w_4)$ gefärbt sind, so sind wir fertig. Gilt das Gegenteil, so bedeutet dies, dass es knotendisjunkte Wege P von w_1 nach w_3 und Q von w_2 nach w_4 in $G - v$ gibt, im Widerspruch zu der oben bewiesenen Behauptung. \square

Hiermit ist dies ein zweites NP -schweres Problem, für welches es einen absoluten Approximationsalgorithmus gibt. In der Tat folgt aus dem Vierfarbensatz, dass die chromatische Zahl eines nicht-bipartiten planaren Graphen nur 3 oder 4 sein kann. Mit dem polynomiellen Algorithmus von Robertson et al. [1997], der jeden gegebenen planaren Graphen mit vier Farben färbt, erhält man einen absoluten Approximationsalgorithmus, welcher höchstens eine Farbe mehr als nötig gebraucht.

Fürer und Raghavachari [1994] haben ein drittes natürliches Problem entdeckt, welches bis auf einen additiven Fehler von 1 approximiert werden kann: In einem gegebenen ungerichteten Graphen sucht man einen aufspannenden Baum, dessen maximaler Grad minimal für alle aufspannenden Bäume ist (dieses Problem verallgemeinert das HAMILTONSCHER-WEG-PROBLEM und ist somit NP -schwer). Deren Algorithmus lässt sich auch auf einen dem STEINERBAUM-PROBLEM entsprechenden allgemeinen Fall erweitern: Für eine gegebene Menge $T \subseteq V(G)$ finde man einen Baum S in G mit $V(T) \subseteq V(S)$, der den maximalen Grad von S für alle solche Bäume minimiert. Singh und Lau [2015] haben eine Erweiterung auf den Fall aufspannender Bäume mit minimalem Gewicht und beschränktem Grad gefunden.

Andererseits besagt der folgende Satz, dass viele Probleme keinen absoluten Approximationsalgorithmus haben, sofern $P \neq NP$ ist:

Proposition 16.23. *Seien \mathcal{I} eine (unendliche) Familie von Mengensystemen, und sei \mathcal{P} das folgende Optimierungsproblem: Für ein gegebenes Mengensystem*

$(E, \mathcal{F}) \in \mathcal{I}$ und eine Funktion $c : E \rightarrow \mathbb{Z}$ finde man eine Menge $F \in \mathcal{F}$, so dass $c(F)$ minimal ist (oder entscheide, dass es keine solche Menge F gibt).

Dann hat \mathcal{P} einen absoluten Approximationsalgorithmus genau dann, wenn \mathcal{P} in polynomieller Zeit gelöst werden kann.

Beweis: Angenommen, es gibt einen polynomiellen Algorithmus A und eine ganze Zahl k , so dass

$$|A((E, \mathcal{F}, c)) - \text{OPT}((E, \mathcal{F}, c))| \leq k$$

für alle Instanzen (E, \mathcal{F}, c) von \mathcal{P} . Wir zeigen nun, wie man \mathcal{P} in polynomieller Zeit exakt löst.

Aus einer gegebenen Instanz (E, \mathcal{F}, c) von \mathcal{P} bilden wir eine neue Instanz (E, \mathcal{F}, c') , wobei $c'(e) := (k+1)c(e)$ für alle $e \in E$. Offensichtlich verändern sich die optimalen Lösungen nicht. Wenden wir nun aber A auf die neue Instanz an, so folgt

$$|A((E, \mathcal{F}, c')) - \text{OPT}((E, \mathcal{F}, c'))| \leq k$$

und somit ist $A((E, \mathcal{F}, c')) = \text{OPT}((E, \mathcal{F}, c'))$. \square

Beispiele hierzu sind das MINIMIERUNGSPROBLEM FÜR UNABHÄNGIGKEITS-SYSTEME und das MAXIMIERUNGSPROBLEM FÜR UNABHÄNGIGKEITSSYSTEME (multipliziere c mit -1) und somit alle in der Liste von Abschnitt 13.1 aufgeführten Probleme.

16.4 Approximationsschemata

Wir erinnern an den im vorigen Abschnitt besprochenen absoluten Approximationsalgorithmus für das KANTENFÄRBUNGS-PROBLEM. Dieser impliziert auch eine relative Approximationsgüte: Da man leicht entscheiden kann, ob die kan-tenchromatische Zahl 1 oder 2 ist (Proposition 16.15), ergibt der Satz von Vizing einen $\frac{4}{3}$ -Approximationsalgorithmus. Andererseits folgt aus Satz 16.14(a), dass es für kein $k < \frac{4}{3}$ einen k -Approximationsalgorithmus gibt (sofern $P \neq NP$).

Aus der Existenz eines absoluten Approximationsalgorithmus folgt somit nicht die Existenz eines k -Approximationsalgorithmus für alle $k > 1$. Eine ähnliche Situation werden wir im Zusammenhang mit dem BIN-PACKING-PROBLEM in Kapitel 18 antreffen. Die obige Feststellung legt die folgende Definition nahe:

Definition 16.24. Sei \mathcal{P} ein Optimierungsproblem mit nichtnegativen Gewichten. Ein **asymptotischer k -Approximationsalgorithmus** für \mathcal{P} ist ein polynomieller Algorithmus A für \mathcal{P} , für welchen es eine Konstante c gibt, so dass

$$\frac{1}{k} \text{OPT}(I) - c \leq A(I) \leq k \text{OPT}(I) + c$$

für alle Instanzen I von \mathcal{P} . Wir sagen auch: A hat die **asymptotische Approximationsgüte k** .

Die (**asymptotische**) **Approximationszahl** eines Optimierungsproblems \mathcal{P} mit nichtnegativen Gewichten sei das Infimum aller Zahlen k , für die es einen (asymptotischen) k -Approximationsalgorithmus für \mathcal{P} gibt; gibt es gar keinen (asymptotischen) Approximationsalgorithmus, so wird sie auf ∞ gesetzt.

Z. B. hat das oben erwähnte KANTENFÄRBUNGS-PROBLEM die Approximationszahl $\frac{4}{3}$ (sofern $P \neq NP$), aber die asymptotische Approximationszahl 1 (nicht nur in einfachen Graphen; siehe Sanders und Steurer [2008]). Optimierungsprobleme mit (asymptotischer) Approximationszahl 1 sind besonders interessant. Für diese führen wir den folgenden Begriff ein:

Definition 16.25. Sei \mathcal{P} ein Optimierungsproblem mit nichtnegativen Gewichten. Ein **Approximationsschema** für \mathcal{P} ist ein Algorithmus A , der als Input eine Instanz I von \mathcal{P} und ein $\epsilon > 0$ annimmt, so dass für jedes feste $\epsilon > 0$ gilt: A ist ein $(1 + \epsilon)$ -Approximationsalgorithmus für \mathcal{P} .

Ein **asymptotisches Approximationsschema** für \mathcal{P} ist ein Algorithmenpaar (A, A') mit den folgenden Eigenschaften: A' ist ein polynomieller Algorithmus, der als Input eine Zahl $\epsilon > 0$ annimmt und eine Zahl c_ϵ berechnet, während A eine Instanz I von \mathcal{P} und ein $\epsilon > 0$ als Input annimmt und als Output eine zulässige Lösung für I liefert, die die folgenden Ungleichungen erfüllt:

$$\frac{1}{1 + \epsilon} \text{OPT}(I) - c_\epsilon \leq A(I, \epsilon) \leq (1 + \epsilon) \text{OPT}(I) + c_\epsilon.$$

Für jedes feste ϵ ist die Laufzeit von A durch ein Polynom in $\text{size}(I)$ beschränkt.

Ein (asymptotisches) Approximationsschema heißt ein **voll-polynomielles (asymptotisches) Approximationsschema**, falls sowohl die Laufzeit als auch die größte in irgendeiner Berechnung auftretende Zahl durch ein Polynom in $\text{size}(I) + \text{size}(\epsilon) + \frac{1}{\epsilon}$ beschränkt ist und c_ϵ polynomiell in $\frac{1}{\epsilon}$ ist.

In einigen Lehrbüchern findet man die Abkürzungen PTAS für (polynomielles) Approximationsschema und FPAS oder FPTAS für voll-polynomielles Approximationsschema.

Abgesehen von absoluten Approximationsalgorithmen, kann man voll-polynomielle Approximationsschemata als das beste zu erreichende Ergebnis für ein NP -schweres Optimierungsproblem betrachten, wenigstens dann, wenn die Kosten einer zulässigen Lösung gleich einer nichtnegativen ganzen Zahl sind (wovon wir in vielen Fällen o. B. d. A. ausgehen können):

Proposition 16.26. Sei $\mathcal{P} = (X, (S_x)_{x \in X}, c, \text{goal})$ ein NP -Optimierungsproblem, wobei die Werte von c nichtnegative ganze Zahlen sind. Sei A ein Algorithmus, der für eine Instanz I von \mathcal{P} und eine Zahl $\epsilon > 0$ eine zulässige Lösung von I mit

$$\frac{1}{1 + \epsilon} \text{OPT}(I) \leq A(I, \epsilon) \leq (1 + \epsilon) \text{OPT}(I)$$

berechnet und dessen Laufzeit durch ein Polynom in $\text{size}(I) + \text{size}(\epsilon)$ beschränkt ist. Dann kann \mathcal{P} in polynomieller Zeit exakt gelöst werden.

Beweis: Zunächst wenden wir A auf $(I, 1)$ für eine gegebene Instanz I an. Setze $\epsilon := \frac{1}{1+2A(I,1)}$ und beachte, dass $\epsilon \text{OPT}(I) < 1$. Nun wenden wir A auf (I, ϵ) an. Da $\text{size}(\epsilon)$ durch ein Polynom in $\text{size}(I)$ beschränkt ist, bildet dieses Verfahren einen polynomiellen Algorithmus. Ist \mathcal{P} ein Minimierungsproblem, so haben wir

$$A(I, \epsilon) \leq (1 + \epsilon) \text{OPT}(I) < \text{OPT}(I) + 1,$$

und da c ganzzahlig ist, folgt hieraus die Optimalität. Ist \mathcal{P} andererseits ein Maximierungsproblem, so folgt analog

$$A(I, \epsilon) \geq \frac{1}{1 + \epsilon} \text{OPT}(I) > (1 - \epsilon) \text{OPT}(I) > \text{OPT}(I) - 1.$$

□

Leider gibt es nur für sehr wenige Probleme ein voll-polynomielles Approximationsschema (siehe Satz 17.11). Auch weisen wir darauf hin, dass die Existenz eines voll-polynomiellen Approximationsschemas keineswegs die Existenz eines absoluten Approximationsalgorithmus bedeutet; ein Gegenbeispiel hierzu ist das sogenannte KNAPSACK-PROBLEM.

In den Kapiteln 17 und 18 werden wir zwei Probleme, nämlich KNAPSACK bzw. BIN-PACKING, besprechen, für die es ein voll-polynomielles Approximationsschema bzw. ein voll-polynomielles asymptotisches Approximationsschema gibt. Für viele Probleme stimmen diese beiden Typen von Approximationsschemata überein:

Satz 16.27. (Papadimitriou und Yannakakis [1993]) *Sei \mathcal{P} ein Optimierungsproblem mit nichtnegativen Gewichten. Angenommen, für jede Konstante k gibt es einen polynomiellen Algorithmus, welcher entscheidet, ob der Optimalwert einer gegebenen Instanz höchstens gleich k ist und falls ja, eine optimale Lösung findet.*

Dann hat \mathcal{P} ein Approximationsschema genau dann, wenn \mathcal{P} ein asymptotisches Approximationsschema hat.

Beweis: Die Notwendigkeit ist trivial. Somit nehmen wir nun an, dass \mathcal{P} ein asymptotisches Approximationsschema (A, A') hat, und beschreiben ein Approximationsschema für \mathcal{P} .

Gegeben sei ein festes $\epsilon > 0$; wir können annehmen, dass $\epsilon < 1$. Setze $\epsilon' := \frac{\epsilon - \epsilon^2}{2 + \epsilon + \epsilon^2}$; es folgt sofort, dass $\epsilon' < \frac{\epsilon}{2}$ ist. Zunächst wenden wir A' auf den Input ϵ' an und bekommen eine Konstante $c_{\epsilon'}$.

Für eine gegebene Instanz I prüfen wir nun, ob $\text{OPT}(I)$ höchstens gleich $\frac{2c_{\epsilon'}}{\epsilon}$ ist. Dies ist eine Konstante für jedes feste ϵ , also können wir diesen Prüfschritt in polynomieller Zeit durchführen und somit eine optimale Lösung finden, falls $\text{OPT}(I) \leq \frac{2c_{\epsilon'}}{\epsilon}$.

Andernfalls wenden wir A auf I und ϵ' an und bekommen eine Lösung mit Wert V , der die folgenden Ungleichungen erfüllt:

$$\frac{1}{1 + \epsilon'} \text{OPT}(I) - c_{\epsilon'} \leq V \leq (1 + \epsilon') \text{OPT}(I) + c_{\epsilon'}.$$

Wir behaupten, diese Lösung genügt: Wir haben $c_{\epsilon'} < \frac{\epsilon}{2} \text{OPT}(I)$, woraus

$$V \leq (1 + \epsilon') \text{OPT}(I) + c_{\epsilon'} < \left(1 + \frac{\epsilon}{2}\right) \text{OPT}(I) + \frac{\epsilon}{2} \text{OPT}(I) = (1 + \epsilon) \text{OPT}(I)$$

folgt, und ferner

$$\begin{aligned} V &\geq \frac{1}{(1 + \epsilon')} \text{OPT}(I) - \frac{\epsilon}{2} \text{OPT}(I) \\ &= \frac{2 + \epsilon + \epsilon^2}{2 + 2\epsilon} \text{OPT}(I) - \frac{\epsilon}{2} \text{OPT}(I) \\ &= \left(\frac{1}{1 + \epsilon} + \frac{\epsilon}{2}\right) \text{OPT}(I) - \frac{\epsilon}{2} \text{OPT}(I) \\ &= \frac{1}{1 + \epsilon} \text{OPT}(I). \end{aligned}$$

□

Wir sehen, dass die Definition eines asymptotischen Approximationsschemas nur für Probleme (wie Bin-Packing oder Färbungsprobleme) sinnvoll ist, deren Beschränkung auf einen konstanten Optimalwert immer noch schwer ist. Für viele Probleme kann diese eingeschränkte Version mittels einer geeigneten vollständigen Enumeration in polynomieller Zeit gelöst werden.

16.5 Maximum-Satisfiability

Das SATISFIABILITY-Problem war unser erstes *NP*-vollständiges Problem. In diesem Abschnitt werden wir das entsprechende Optimierungsproblem analysieren:

MAX-SAT

Instanz: Eine Menge X von Variablen, eine Familie \mathcal{Z} von Klauseln über X und eine Gewichtsfunktion $c : \mathcal{Z} \rightarrow \mathbb{R}_+$.

Aufgabe: Bestimme eine Wahrheitsbelegung T von X , so dass das Gesamtgewicht aller von T erfüllten Klauseln in \mathcal{Z} maximal ist.

Wie wir sehen werden, ist die Approximation von MAX-SAT ein schönes Beispiel (und eines der ersten) für die algorithmische Anwendung der sogenannten probabilistischen Methode.

Zunächst betrachten wir den folgenden trivialen randomisierten Algorithmus: Setze jede Variable unabhängig auf *true* mit der Wahrscheinlichkeit $\frac{1}{2}$. Offensichtlich erfüllt dieser Algorithmus jede Klausel Z mit Wahrscheinlichkeit $1 - 2^{-|Z|}$.

Zufallsvariablen, die *true* mit Wahrscheinlichkeit $\frac{1}{2}$ und sonst *false* sind, werden wir mit r bezeichnen, und es sei $R = (r, r, \dots, r)$ die über alle Wahrheitsbelegungen uniform verteilte Zufallsvariable (die verschiedenen Kopien von r sind alle unabhängig von einander). Sei ferner $c(T)$ das Gesamtgewicht aller von der Wahrheitsbelegung T erfüllten Klauseln. Dann wird der Erwartungswert des Gesamtgewichts der von R erfüllten Klauseln gegeben durch

$$\begin{aligned}
 \text{Exp}(c(R)) &= \sum_{Z \in \mathcal{Z}} c(Z) \text{Prob}(R \text{ erfüllt } Z) \\
 &= \sum_{Z \in \mathcal{Z}} c(Z) (1 - 2^{-|Z|}) \\
 &\geq (1 - 2^{-p}) \sum_{Z \in \mathcal{Z}} c(Z),
 \end{aligned} \tag{16.5}$$

wobei $p := \min_{Z \in \mathcal{Z}} |Z|$; Exp bzw. Prob bezeichnet den Erwartungswert bzw. die Wahrscheinlichkeit.

Da das Optimum nicht größer als $\sum_{Z \in \mathcal{Z}} c(Z)$ werden kann, erwarten wir, dass R eine um höchstens den Faktor $\frac{1}{1-2^{-p}}$ vom Optimum abweichende Lösung liefert. Was wir aber eigentlich haben möchten, ist ein deterministischer Approximationsalgorithmus. In der Tat können wir unseren (trivialen) randomisierten Algorithmus unter Beibehaltung der Approximationsgüte in einen deterministischen Algorithmus umwandeln. Dieser Schritt ist auch bekannt als Derandomisierung.

Wir werden nun die Wahrheitsbelegung Schritt für Schritt festlegen. Wir nehmen an, es sei $X = \{x_1, \dots, x_n\}$ und wir haben bereits eine Wahrheitsbelegung T für x_1, \dots, x_k ($0 \leq k < n$) festgelegt. Setzen wir nun x_{k+1}, \dots, x_n zufällig und unabhängig auf *true* mit der Wahrscheinlichkeit $\frac{1}{2}$, so erfüllen wir Klauseln, dessen Gesamtgewicht den Erwartungswert $e_0 = \text{Exp}(c(T(x_1), \dots, T(x_k), r, \dots, r))$ hat. Setzen wir x_{k+1} auf *true* bzw. *false* und x_{k+2}, \dots, x_n zufällig, so hat das Gesamtgewicht der erfüllten Klauseln den Erwartungswert etwa e_1 bzw. e_2 . Die Werte e_1 und e_2 können als bedingte Erwartungswerte betrachtet werden. Es ist trivialerweise $e_0 = \frac{e_1 + e_2}{2}$, also ist entweder e_1 oder e_2 größer als e_0 , oder beide sind gleich e_0 . Wir setzen x_{k+1} auf *true*, falls $e_1 \geq e_2$, und sonst auf *false*. Dieses Verfahren wird gelegentlich die Methode der bedingten Wahrscheinlichkeiten genannt.

JOHNSONS ALGORITHMUS

Input: Eine Menge $X = \{x_1, \dots, x_n\}$ von Variablen, eine Familie \mathcal{Z} von Klauseln über X und eine Gewichtsfunktion $c : \mathcal{Z} \rightarrow \mathbb{R}_+$.

Output: Eine Wahrheitsbelegung $T : X \rightarrow \{\text{true}, \text{false}\}$.

① **For** $k := 1$ **to** n **do:**

If $\text{Exp}(c(T(x_1), \dots, T(x_{k-1}), \text{true}, r, \dots, r))$
 $\geq \text{Exp}(c(T(x_1), \dots, T(x_{k-1}), \text{false}, r, \dots, r))$
then setze $T(x_k) := \text{true}$
else setze $T(x_k) := \text{false}$.

Die Erwartungswerte können leicht mittels (16.5) berechnet werden.

Satz 16.28. (Johnson [1974]) JOHNSONS ALGORITHMUS ist ein $\frac{1}{1-2^{-p}}$ -Approximationsalgorithmus für MAX-SAT, wobei p die minimale Kardinalität einer Klausel ist.

Beweis: Wir definieren für $k = 0, \dots, n$ den bedingten Erwartungswert

$$s_k := \text{Exp}(c(T(x_1), \dots, T(x_k), r, \dots, r)).$$

Beachte, dass $s_n = c(T)$ das Gesamtgewicht der durch unseren Algorithmus erfüllten Klauseln ist und dass mit (16.5) folgende Ungleichung gilt: $s_0 = \text{Exp}(c(R)) \geq (1 - 2^{-p}) \sum_{Z \in \mathcal{Z}} c(Z)$.

Ferner gilt $s_i \geq s_{i-1}$ nach Wahl von $T(x_i)$ in ① (für $i = 1, \dots, n$). Somit haben wir $s_n \geq s_0 \geq (1 - 2^{-p}) \sum_{Z \in \mathcal{Z}} c(Z)$. Der Optimalwert ist aber höchstens gleich $\sum_{Z \in \mathcal{Z}} c(Z)$, womit der Beweis abgeschlossen ist. \square

Da $p \geq 1$ ist, haben wir einen 2-Approximationsalgorithmus. Das ist aber nicht besonders interessant, da es einen viel einfacheren 2-Approximationsalgorithmus gibt: Setze entweder alle Variablen auf *true* oder alle auf *false* und nimm die bessere der beiden Wahlen. Chen, Friesen und Zheng [1999] haben jedoch gezeigt, dass JOHNSONS ALGORITHMUS tatsächlich ein $\frac{3}{2}$ -Approximationsalgorithmus ist.

Gibt es keine eelementigen Klauseln ($p \geq 2$), so ist JOHNSONS ALGORITHMUS nach Satz 16.28 ein $\frac{4}{3}$ -Approximationsalgorithmus und für $p \geq 3$ ein $\frac{8}{7}$ -Approximationsalgorithmus.

Yannakakis [1994] hat einen $\frac{4}{3}$ -Approximationsalgorithmus für den allgemeinen Fall mittels Netzwerkfluss-Methoden konstruiert. Hier werden wir einen einfacheren $\frac{4}{3}$ -Approximationsalgorithmus von Goemans und Williamson [1994] beschreiben.

Wir können MAX-SAT ohne weiteres als ganzzahliges LP formulieren: Für die Variablen $X = \{x_1, \dots, x_n\}$, Klauseln $\mathcal{Z} = \{Z_1, \dots, Z_m\}$ und Gewichte c_1, \dots, c_m haben wir

$$\begin{aligned} \max \quad & \sum_{j=1}^m c_j z_j \\ \text{bzgl.} \quad & z_j \leq \sum_{i: x_i \in Z_j} y_i + \sum_{i: \bar{x}_i \in Z_j} (1 - y_i) \quad (j = 1, \dots, m) \\ & y_i, z_j \in \{0, 1\} \quad (i = 1, \dots, n, j = 1, \dots, m). \end{aligned}$$

Hier bedeutet $y_i = 1$, dass die Variable x_i auf *true* gesetzt ist, und $z_j = 1$, dass die Klausel Z_j erfüllt ist. Nun betrachten wir die folgende LP-Relaxierung:

$$\begin{aligned} \max \quad & \sum_{j=1}^m c_j z_j \\ \text{bzgl.} \quad & z_j \leq \sum_{i: x_i \in Z_j} y_i + \sum_{i: \bar{x}_i \in Z_j} (1 - y_i) \quad (j = 1, \dots, m) \\ & y_i \leq 1 \quad (i = 1, \dots, n) \\ & y_i \geq 0 \quad (i = 1, \dots, n) \\ & z_j \leq 1 \quad (j = 1, \dots, m) \\ & z_j \geq 0 \quad (j = 1, \dots, m). \end{aligned} \tag{16.6}$$

Sei (y^*, z^*) eine optimale Lösung des LP (16.6). Nun setzen wir jede Variable x_i unabhängig und mit Wahrscheinlichkeit y_i^* auf *true*. Dieser Schritt ist als zufälliges Runden bekannt und wurde von Raghavan und Thompson [1987] eingeführt. Obiges Verfahren bildet einen weiteren randomisierten Algorithmus für

MAX-SAT, der wie oben derandomisiert werden kann. Sei r_p die Zufallsvariable, die mit Wahrscheinlichkeit p true ist und sonst false.

GOEMANS-WILLIAMSON-ALGORITHMUS

Input: Eine Menge $X = \{x_1, \dots, x_n\}$ von Variablen, eine Familie \mathcal{Z} von Klauseln über X und eine Gewichtsfunktion $c : \mathcal{Z} \rightarrow \mathbb{R}_+$.
Output: Eine Wahrheitsbelegung $T : X \rightarrow \{\text{true}, \text{false}\}$.

-
- ① Löse das LP (16.6); sei (y^*, z^*) eine optimale Lösung.
 - ② **For** $k := 1$ **to** n **do**
 - If** $\text{Exp}(c(T(x_1), \dots, T(x_{k-1}), \text{true}, r_{y_{k+1}^*}, \dots, r_{y_n^*}) \geq \text{Exp}(c(T(x_1), \dots, T(x_{k-1}), \text{false}, r_{y_{k+1}^*}, \dots, r_{y_n^*})$
 - then** setze $T(x_k) := \text{true}$
 - else** setze $T(x_k) := \text{false}$.
-

Satz 16.29. (Goemans und Williamson [1994]) Der GOEMANS-WILLIAMSON-ALGORITHMUS ist ein $\frac{1}{1 - (1 - \frac{1}{q})^q}$ -Approximationsalgorithmus, wobei q die maximale Kardinalität einer Klausel ist.

Beweis: Für $k = 0, \dots, n$ setzen wir

$$s_k := \text{Exp}(c(T(x_1), \dots, T(x_k), r_{y_{k+1}^*}, \dots, r_{y_n^*})).$$

Wiederum haben wir $s_i \geq s_{i-1}$ für $i = 1, \dots, n$, und $s_n = c(T)$ ist das Gesamtgewicht der von unserem Algorithmus erfüllten Klauseln. Somit müssen wir nur noch $s_0 = \text{Exp}(c(R_{y^*}))$ schätzen, wobei $R_{y^*} = (r_{y_1^*}, \dots, r_{y_n^*})$.

Für $j = 1, \dots, m$ ist die Wahrscheinlichkeit, dass die Klausel Z_j von R_{y^*} erfüllt wird, gleich

$$1 - \left(\prod_{i: x_i \in Z_j} (1 - y_i^*) \right) \cdot \left(\prod_{i: \bar{x}_i \in Z_j} y_i^* \right).$$

Da das geometrische Mittel immer kleiner oder gleich dem arithmetischen Mittel ist, beträgt diese Wahrscheinlichkeit mindestens

$$\begin{aligned} & 1 - \left(\frac{1}{|Z_j|} \left(\sum_{i: x_i \in Z_j} (1 - y_i^*) + \sum_{i: \bar{x}_i \in Z_j} y_i^* \right) \right)^{|Z_j|} \\ &= 1 - \left(1 - \frac{1}{|Z_j|} \left(\sum_{i: x_i \in Z_j} y_i^* + \sum_{i: \bar{x}_i \in Z_j} (1 - y_i^*) \right) \right)^{|Z_j|} \\ &\geq 1 - \left(1 - \frac{z_j^*}{|Z_j|} \right)^{|Z_j|} \\ &\geq \left(1 - \left(1 - \frac{1}{|Z_j|} \right)^{|Z_j|} \right) z_j^*. \end{aligned}$$

Zum Beweis der letzten Ungleichung: Beachte, dass für jedes $0 \leq a \leq 1$ und jedes $k \in \mathbb{N}$

$$1 - \left(1 - \frac{a}{k}\right)^k \geq a \left(1 - \left(1 - \frac{1}{k}\right)^k\right)$$

gilt: Beide Seiten der Ungleichung sind für $a \in \{0, 1\}$ gleich, und die linke Seite ist (als Funktion von a) konkav, während die rechte Seite linear ist.

Somit haben wir

$$\begin{aligned} s_0 = \text{Exp}(c(R_{y^*})) &= \sum_{j=1}^m c_j \text{Prob}(R_{y^*} \text{ erfüllt } Z_j) \\ &\geq \sum_{j=1}^m c_j \left(1 - \left(1 - \frac{1}{|Z_j|}\right)^{|Z_j|}\right) z_j^* \\ &\geq \left(1 - \left(1 - \frac{1}{q}\right)^q\right) \sum_{j=1}^m c_j z_j^* \end{aligned}$$

(beachte, dass die Folge $\left(\left(1 - \frac{1}{k}\right)^k\right)_{k \in \mathbb{N}}$ monoton steigend ist und gegen $\frac{1}{e}$ konvergiert). Da das Optimum kleiner oder gleich dem optimalen Zielfunktionswert $\sum_{j=1}^m z_j^* c_j$ der LP-Relaxierung ist, ist der Beweis abgeschlossen. \square

Da $\left(1 - \frac{1}{q}\right)^q < \frac{1}{e}$, haben wir einen $\frac{e}{e-1}$ -Approximationsalgorithmus ($\frac{e}{e-1}$ ist annähernd gleich 1,582).

Wir haben nun zwei ähnliche Algorithmen mit verschiedenen Eigenschaften: der erste funktioniert besser bei langen Klauseln und der zweite bei kurzen. Somit liegt es nahe, beide Algorithmen zu kombinieren:

Satz 16.30. (Goemans und Williamson [1994]) *Das folgende Verfahren ist ein $\frac{4}{3}$ -Approximationsalgorithmus für MAX-SAT: Wende sowohl JOHNSONS ALGORITHMUS als auch den GOEMANS-WILLIAMSON-ALGORITHMUS an und wähle die bessere der beiden Lösungen.*

Beweis: Wir bedienen uns der in den obigen Beweisen eingeführten Notation. Der Algorithmus liefert als Output eine Wahrheitsbelegung, welche eine gewisse Menge von Klauseln mit mindestens dem folgenden Gesamtgewicht erfüllt:

$$\begin{aligned} &\max\{\text{Exp}(c(R)), \text{Exp}(c(R_{y^*}))\} \\ &\geq \frac{1}{2} (\text{Exp}(c(R)) + \text{Exp}(c(R_{y^*}))) \\ &\geq \frac{1}{2} \sum_{j=1}^m \left(\left(1 - 2^{-|Z_j|}\right) c_j + \left(1 - \left(1 - \frac{1}{|Z_j|}\right)^{|Z_j|}\right) z_j^* c_j \right) \\ &\geq \frac{1}{2} \sum_{j=1}^m \left(2 - 2^{-|Z_j|} - \left(1 - \frac{1}{|Z_j|}\right)^{|Z_j|} \right) z_j^* c_j \\ &\geq \frac{3}{4} \sum_{j=1}^m z_j^* c_j. \end{aligned}$$

Beachte zur letzten Ungleichung, dass $2 - 2^{-k} - \left(1 - \frac{1}{k}\right)^k \geq \frac{3}{2}$ für alle $k \in \mathbb{N}$: Für $k \in \{1, 2\}$ haben wir Gleichheit; für $k \geq 3$ haben wir $2 - 2^{-k} - \left(1 - \frac{1}{k}\right)^k \geq 2 - \frac{1}{8} - \frac{1}{e} > \frac{3}{2}$. Da das Optimum mindestens gleich $\sum_{j=1}^m z_j^* c_j$ ist, ist der Satz bewiesen. \square

Etwas bessere Approximationsalgorithmen für MAX-SAT (die semidefinite Optimierung benutzen) sind entwickelt worden; siehe Goemans und Williamson [1995], Mahajan und Ramesh [1999], Feige und Goemans [1995], und Asano [2006]. Der beste momentan bekannte Algorithmus erreicht eine Approximationsgüte von 1,256 (Avidor, Berkovitch und Zwick [2006]).

Bellare und Sudan [1994] haben in der Tat beweisen können, dass das Problem der Approximation von MAX-SAT bis auf den Faktor $\frac{74}{73}$ NP-schwer ist. Sogar für MAX-3SAT (d. h. MAX-SAT beschränkt auf Instanzen mit Klauseln, die genau drei Literale enthalten) gibt es kein Approximationsschema (sofern $P \neq NP$), wie wir im nächsten Abschnitt zeigen werden.

16.6 Der PCP-Satz

Viele Nicht-Approximierbarkeits-Resultate basieren auf einem tiefgehenden Satz, der eine neue Charakterisierung der Klasse NP beschreibt. Wir erinnern daran, dass ein Entscheidungsproblem genau dann in NP ist, wenn es einen polynomiellen Zertifikat-Prüfalgorithmus gibt. Hier betrachten wir randomisierte Zertifikat-Prüfalgorithmen, die die Instanz vollständig lesen, das zu prüfende Zertifikat aber nur teilweise. Sie akzeptieren immer Ja-Instanzen mit korrekten Zertifikaten, und manchmal auch Nein-Instanzen.

Welche Bits eines Zertifikats gelesen werden, wird vorweg zufällig bestimmt; genauer: Diese Entscheidung hängt von der Instanz x ab und von $O(\log(\text{size}(x)))$ Zufallsbits.

Diesen Begriff definieren wir nun formal. Für einen String s und $t \in \mathbb{N}^k$ sei s_t der String der Länge k , dessen i -te Komponente gleich der t_i -ten Komponente von s ($i = 1, \dots, k$) ist.

Definition 16.31. Ein Entscheidungsproblem (X, Y) ist in der Klasse $PCP(\log n, 1)$, falls es Folgendes gibt: ein Polynom p , eine Konstante $k \in \mathbb{N}$, eine in polynomieller Zeit berechenbare Funktion

$$f : \left\{ (x, r) : x \in X, r \in \{0, 1\}^{\lfloor \log(p(\text{size}(x))) \rfloor} \right\} \rightarrow \mathbb{N}^k$$

mit $f(x, r) \in \{1, \dots, \lfloor p(\text{size}(x)) \rfloor\}^k$ für alle x und r und ein Entscheidungsproblem (X', Y') in P mit

$$X' := \{(x, \pi, \gamma) : x \in X, \pi \in \{1, \dots, \lfloor p(\text{size}(x)) \rfloor\}^k, \gamma \in \{0, 1\}^k\},$$

so dass für jede Instanz $x \in X$ Folgendes gilt: Ist $x \in Y$, so gibt es ein $c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor}$ mit $\text{Prob}((x, f(x, r), c_{f(x, r)}) \in Y') = 1$. Ist $x \notin Y$, so ist $\text{Prob}((x, f(x, r), c_{f(x, r)}) \in Y') < \frac{1}{2}$ für alle $c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor}$.

Hier gelten die Wahrscheinlichkeiten für eine Gleichverteilung der Zufallsstrings $r \in \{0, 1\}^{\lfloor \log(p(\text{size}(x))) \rfloor}$.

Die Buchstaben „PCP“ sind die Abkürzung für „probabilistically checkable proof“, auf Deutsch „probabilistisch prüfbarer Beweis“. Die Parameter $\log n$ und 1 deuten an, dass für eine Instanz der Größe n , $O(\log n)$ Zufallsbits gebraucht werden und $O(1)$ Bits des Zertifikats gelesen werden.

Für jede Ja-Instanz gibt es ein Zertifikat, welches immer akzeptiert wird; für Nein-Instanzen hingegen wird kein String mit Wahrscheinlichkeit größer oder gleich $\frac{1}{2}$ als Zertifikat akzeptiert. Beachte, dass diese Fehlerwahrscheinlichkeit von $\frac{1}{2}$ äquivalent durch irgendeine Zahl zwischen 0 und 1 ersetzt werden kann (Aufgabe 20).

Proposition 16.32. $PCP(\log n, 1) \subseteq NP$.

Beweis: Sei $(X, Y) \in PCP(\log n, 1)$ und seien $p, k, f, (X', Y')$ wie in Definition 16.31 angegeben. Setze $X'' := \{(x, c) : x \in X, c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor}\}$ und

$$Y'' := \{(x, c) \in X'' : \text{Prob}((x, f(x, r), c_{f(x, r)}) \in Y') = 1\}.$$

Um zu zeigen, dass $(X, Y) \in NP$, genügt es zu zeigen, dass $(X'', Y'') \in P$. Da es aber nur $2^{\lfloor \log(p(\text{size}(x))) \rfloor}$, d. h. höchstens $p(\text{size}(x))$ viele Strings $r \in \{0, 1\}^{\lfloor \log(p(\text{size}(x))) \rfloor}$ gibt, können wir sie alle prüfen. Für jeden berechnen wir $f(x, r)$ und prüfen, ob $(x, f(x, r), c_{f(x, r)}) \in Y'$ (wir benutzen die Tatsache, dass $(X', Y') \in P$). Die Gesamlaufzeit ist polynomiell in $\text{size}(x)$. \square

Das überraschende Ergebnis ist nun, dass diese randomisierten Zertifikat-Prüfalgorithmen, die nur eine konstante Anzahl von Bits in einem Zertifikat lesen, genau so mächtig sind, wie die normalen (deterministischen) Zertifikat-Prüfalgorithmen, die ja über die volle Information verfügen. Dies ist der Inhalt des sogenannten *PCP*-Satzes:

Satz 16.33. (Arora et al. [1998])

$$NP = PCP(\log n, 1).$$

Der Beweis der Inklusion $NP \subseteq PCP(\log n, 1)$ ist sehr schwer und würde den Rahmen dieses Buches sprengen. Er beruht auf früheren (schwächeren) Resultaten von Feige et al. [1996] und von Arora und Safra [1998]. Siehe auch Arora [1994], Hougardy, Prömel und Steger [1994] oder Ausiello et al. [1999] für einen vollständigen Beweis des *PCP*-Satzes (Satz 16.33). Stärkere Resultate wurden später von Bellare, Goldreich und Sudan [1998] und von Håstad [2001] bewiesen. Z. B. kann die Zahl k in Definition 16.31 gleich 9 genommen werden. Ein neuer Beweis des *PCP*-Satzes wurde von Dinur [2007] entwickelt.

Wir werden jedoch einige der Konsequenzen des *PCP*-Satzes für die Nicht-Approximierbarkeit kombinatorischer Optimierungsprobleme besprechen. Zunächst betrachten wir das **MAXIMUM-CLIQUE-PROBLEM** und das **MAXIMUM-STABLE-SET-PROBLEM**: Für einen gegebenen ungerichteten Graphen G bestimme man eine Clique bzw. eine stabile Menge maximaler Kardinalität in G .

Wir erinnern an Proposition 2.2 (und Korollar 15.24): Die drei Probleme der Bestimmung einer kardinalitätsmaximalen Clique, einer kardinalitätsmaximalen stabilen Menge, bzw. einer kardinalitätsminimalen Knotenüberdeckung sind äquivalent. Die Existenz des 2-Approximationsalgorithmus für das **MINIMUM-VERTEX-COVER-PROBLEM** (siehe Abschnitt 16.1) impliziert jedoch nicht die Existenz eines Approximationsalgorithmus für das **MAXIMUM-STABLE-SET-PROBLEM** oder das **MAXIMUM-CLIQUE-PROBLEM**.

Es kann nämlich passieren, dass der Algorithmus eine Knotenüberdeckung C der Größe $n - 2$ liefert, während das Optimum gleich $\frac{n}{2} - 1$ ist, wobei $n = |V(G)|$. Das Komplement $V(G) \setminus C$ ist dann eine stabile Menge der Kardinalität 2, aber die maximale Kardinalität einer stabilen Menge ist $\frac{n}{2} + 1$. Dieses Beispiel zeigt, dass bei der Übertragung eines Algorithmus auf ein anderes Problem mittels einer polynomiellen Transformation die Approximationsgüte im Allgemeinen nicht erhalten bleibt. Im nächsten Abschnitt werden wir einen eingeschränkten Transformationstyp betrachten. Hier leiten wir noch das folgende Nicht-Approximierbarkeits-Resultat für das **MAXIMUM-CLIQUE-PROBLEM** aus dem *PCP*-Satz ab:

Satz 16.34. (Arora und Safra [1998]) *Gilt $P \neq NP$, so gibt es keinen 2-Approximationsalgorithmus für das MAXIMUM-CLIQUE-PROBLEM.*

Beweis: Sei $\mathcal{P} = (X, Y)$ ein *NP*-vollständiges Problem. Nach dem *PCP*-Satz (Satz 16.33) ist \mathcal{P} in $PCP(\log n, 1)$, also seien p, k, f und $\mathcal{P}' := (X', Y')$ wie in Definition 16.31 angegeben.

Nun konstruieren wir einen Graphen G_x für ein gegebenes $x \in X$ wie folgt. Sei

$$V(G_x) := \left\{ (r, a) : r \in \{0, 1\}^{\lfloor \log(p(\text{size}(x))) \rfloor}, a \in \{0, 1\}^k, (x, f(x, r), a) \in Y' \right\}$$

(entsprechend allen „akzeptierenden Läufen“ des randomisierten Zertifikat-Prüfalgorithmus). Zwei Knoten (r, a) und (r', a') werden durch eine Kante verbunden, falls jedes Mal, wenn die i -te Komponente von $f(x, r)$ gleich der j -ten Komponente von $f(x, r')$ ist, auch $a_i = a'_j$ gilt (beachte, dass daraus $r \neq r'$ folgt). Da \mathcal{P}' in P ist und es nur eine polynomielle Anzahl von Zufallsstrings gibt, kann G_x in polynomieller Zeit berechnet werden (und hat polynomielle Größe).

Ist $x \in Y$, so gibt es definitionsgemäß ein Zertifikat $c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor}$, so dass $(x, f(x, r), c_{f(x, r)}) \in Y'$ für alle $r \in \{0, 1\}^{\lfloor \log(p(\text{size}(x))) \rfloor}$. Damit gibt es eine Clique der Größe $2^{\lfloor \log(p(\text{size}(x))) \rfloor}$ in G_x .

Ist andererseits $x \notin Y$, so gibt es keine Clique der Größe $\frac{1}{2}2^{\lfloor \log(p(\text{size}(x))) \rfloor}$ in G_x : Angenommen, es seien $(r^{(1)}, a^{(1)}), \dots, (r^{(t)}, a^{(t)})$ die Knoten einer Clique. Dann sind die $r^{(1)}, \dots, r^{(t)}$ paarweise verschieden. Wir setzen $c_i := a_k^{(j)}$, falls

die k -te Komponente von $f(x, r^{(j)})$ gleich i ist, und alle anderen Komponenten von c (falls vorhanden) beliebig. Auf diese Weise erhalten wir ein Zertifikat c mit $(x, f(x, r^{(i)}), c_{f(x, r^{(i)})}) \in Y'$ für alle $i = 1, \dots, t$. Ist $x \notin Y$, so gilt $t < \frac{1}{2}2^{\lfloor \log(p(\text{size}(x))) \rfloor}$.

Somit kann jeder 2-Approximationsalgorithmus für das MAXIMUM-CLIQUE-PROBLEM entscheiden, ob $x \in Y$, d.h. er kann \mathcal{P} lösen. Es ist \mathcal{P} aber NP -vollständig, also ist dies nur möglich, falls $P = NP$. \square

Die Reduktion im obigen Beweis stammt von Feige et al. [1996]. Da die Fehlerwahrscheinlichkeit $\frac{1}{2}$ in Definition 16.31 durch jede Zahl zwischen 0 und 1 ersetzt werden kann (Aufgabe 20), folgt, dass es für $\rho \geq 1$ keinen ρ -Approximationsalgorithmus für das MAXIMUM-CLIQUE-PROBLEM gibt, sofern $P \neq NP$.

Mit einem größeren Aufwand hat Zuckerman [2007] gezeigt, dass es, sofern $P \neq NP$, keinen polynomiellen Algorithmus gibt, der die maximale Größe einer Clique in irgendeinem Graphen mit n Knoten bis auf den Faktor $n^{1-\epsilon}$ für irgendein festes $\epsilon > 0$ berechnet. Der beste bekannte Algorithmus findet in diesem Fall stets eine Clique der Größe $\frac{k \log^3 n}{n(\log \log n)^2}$ (Feige [2004]). Natürlich gilt all dies auch für das MAXIMUM-STABLE-SET-PROBLEM (betrachte den Komplementgraphen des gegebenen Graphen).

Nun wenden wir uns der folgenden eingeschränkten Version von MAX-SAT zu:

MAX-3SAT

Instanz: Eine Menge X von Variablen und eine Familie \mathcal{Z} von Klauseln über X , jede mit genau drei Literalen.

Aufgabe: Bestimme eine Wahrheitsbelegung T von X , so dass die Anzahl der von T erfüllten Klauseln in \mathcal{Z} maximal ist.

In Abschnitt 16.5 hatten wir einen einfachen $\frac{8}{7}$ -Approximationsalgorithmus für MAX-3SAT, sogar für die gewichtete Variante (Satz 16.28). Håstad [2001] hat bewiesen, dass dies das bestmögliche Ergebnis ist: Für $\rho < \frac{8}{7}$ gibt es keinen ρ -Approximationsalgorithmus für MAX-3SAT, sofern $P \neq NP$. Hier beweisen wir das folgende schwächere Resultat:

Satz 16.35. (Arora et al. [1998]) *Gilt $P \neq NP$, so gibt es kein Approximationsschema für MAX-3SAT.*

Beweis: Sei $\mathcal{P} = (X, Y)$ ein NP -vollständiges Problem. Nach dem PCP-Satz (Satz 16.33) ist \mathcal{P} in $PCP(\log n, 1)$, also seien p, k, f und $\mathcal{P}' := (X', Y')$ wie in Definition 16.31 angegeben.

Nun konstruieren wir eine 3SAT-Instanz J_x für ein gegebenes $x \in X$ wie folgt. Für jeden Zufallsstring $r \in \{0, 1\}^{\lfloor \log(p(\text{size}(x))) \rfloor}$ definieren wir eine Familie \mathcal{Z}_r von 3SAT-Klauseln (die Vereinigung all dieser Familien wird J_x bilden). Zunächst konstruieren wir eine Familie \mathcal{Z}'_r von Klauseln mit beliebig vielen Literalen und wenden dann Proposition 15.21 an.

Sei also $r \in \{0, 1\}^{\lfloor \log(p(\text{size}(x))) \rfloor}$ und $f(x, r) = (t_1, \dots, t_k)$. Sei $\{a^{(1)}, \dots, a^{(s_r)}\}$ die Menge der Strings $a \in \{0, 1\}^k$ mit $(x, f(x, r), a) \in Y'$.

Ist $s_r = 0$, so setzen wir einfach $\mathcal{Z}' := \{\{y\}, \{\bar{y}\}\}$, wobei y eine Variable ist, die sonst nirgends vorkommt.

Andernfalls, sei $c \in \{0, 1\}^{\lfloor p(\text{size}(x)) \rfloor}$. Es ist $(x, f(x, r), c_{f(x, r)}) \in Y'$ genau dann, wenn

$$\bigvee_{j=1}^{s_r} \left(\bigwedge_{i=1}^k (c_{t_i} = a_i^{(j)}) \right).$$

Dies ist äquivalent mit

$$\bigwedge_{(i_1, \dots, i_{s_r}) \in \{1, \dots, k\}^{s_r}} \left(\bigvee_{j=1}^{s_r} (c_{t_{i_j}} = a_{i_j}^{(j)}) \right).$$

Diese Konjunktion von Klauseln kann in polynomieller Zeit gebildet werden, da $\mathcal{P}' \in P$ und k eine Konstante ist. Führen wir boolesche Variablen $\pi_1, \dots, \pi_{\lfloor p(\text{size}(x)) \rfloor}$ entsprechend den Bits $c_1, \dots, c_{\lfloor p(\text{size}(x)) \rfloor}$ ein, so erhalten wir eine Familie \mathcal{Z}'_r von k^{s_r} Klauseln (jede mit s_r Literalen) mit der Eigenschaft: \mathcal{Z}'_r wird genau dann erfüllt, wenn $(x, f(x, r), c_{f(x, r)}) \in Y'$.

Nach Proposition 15.21 können wir jedes \mathcal{Z}'_r äquivalent als Konjunktion von 3SAT-Klauseln schreiben, wobei die Anzahl der Klauseln um höchstens den Faktor $\max\{s_r - 2, 4\}$ wächst. Es bezeichne \mathcal{Z}_r diese Familie von Klauseln. Da $s_r \leq 2^k$, besteht jedes \mathcal{Z}_r aus höchstens $l := k^{2^k} \max\{2^k - 2, 4\}$ 3SAT-Klauseln.

Unsere 3SAT-Instanz J_x ist die Vereinigung dieser Familien \mathcal{Z}_r für alle r . Beachte, dass J_x in polynomieller Zeit berechnet werden kann.

Ist nun x eine Ja-Instanz, so gibt es ein Zertifikat c wie in Definition 16.31. Dieses c definiert sofort eine J_x erfüllende Wahrheitsbelegung.

Ist andererseits x eine Nein-Instanz, so ist nur die Hälfte der Formeln \mathcal{Z}_r gleichzeitig erfüllbar. Somit lässt jede Wahrheitsbelegung in diesem Fall mindestens den Bruchteil $\frac{1}{2^l}$ der Klauseln unerfüllt.

Damit erfüllt jeder k -Approximationssalgorithmus für MAX-3SAT mit $k < \frac{2l}{2l-1}$ mehr als den Bruchteil $\frac{2l-1}{2l} = 1 - \frac{1}{2^l}$ der Klauseln irgendeiner erfüllbaren Instanz. Somit kann ein solcher Algorithmus entscheiden, ob $x \in Y$. Da \mathcal{P} aber NP-vollständig ist, gibt es keinen solchen Algorithmus, sofern $P \neq NP$. \square

16.7 L-Reduktionen

Unser Ziel ist es, zu zeigen, dass auch andere Probleme außer MAX-3SAT kein Approximationsschema haben, sofern $P \neq NP$. Wie bei den NP-Vollständigkeitsbeweisen (Abschnitt 15.5) ist es auch hier nicht notwendig, einen direkten Beweis für jedes Problem unter Verwendung der Definition von $PCP(\log n, 1)$ zu führen. Stattdessen benutzen wir einen besonderen Reduktionstyp, bei dem Approximierbarkeit erhalten bleibt (bei allgemeinen polynomiellen Transformationen ist dies nicht der Fall):

Definition 16.36. Seien $\mathcal{P} = (X, (S_x)_{x \in X}, c, \text{goal})$ und $\mathcal{P}' = (X', (S'_x)_{x \in X'}, c', \text{goal}')$ zwei Optimierungsprobleme mit nichtnegativen Gewichten. Eine **L-Reduktion** von \mathcal{P} auf \mathcal{P}' besteht aus zwei Funktionen f und g , beide berechenbar in polynomieller Zeit, und zwei Konstanten $\alpha, \beta > 0$, so dass für jede Instanz x von \mathcal{P} Folgendes gilt:

- (a) Es ist $f(x)$ eine Instanz von \mathcal{P}' mit $\text{OPT}(f(x)) \leq \alpha \text{OPT}(x)$;
- (b) Für jede zulässige Lösung y' von $f(x)$ ist $g(x, y')$ eine zulässige Lösung von x , so dass $|c(x, g(x, y')) - \text{OPT}(x)| \leq \beta |c'(f(x), y') - \text{OPT}(f(x))|$.

Wir sagen, dass \mathcal{P} auf \mathcal{P}' **L-reduzierbar** ist, falls es eine L-Reduktion von \mathcal{P} auf \mathcal{P}' gibt.

Der Buchstabe „L“ bedeutet „linear“. L-Reduktionen wurden von Papadimitriou und Yannakakis [1991] eingeführt. Aus der Definition folgt sofort, dass L-Reduktionen zusammengesetzt werden können:

Proposition 16.37. Seien $\mathcal{P}, \mathcal{P}'$ und \mathcal{P}'' Optimierungsprobleme mit nichtnegativen Gewichten. Ist (f, g, α, β) eine L-Reduktion von \mathcal{P} auf \mathcal{P}' und $(f', g', \alpha', \beta')$ eine L-Reduktion von \mathcal{P}' auf \mathcal{P}'' , so ist die Komposition $(f'', g'', \alpha\alpha', \beta\beta')$ dieser L-Reduktionen die L-Reduktion von \mathcal{P} auf \mathcal{P}'' mit $f''(x) = f'(f(x))$ und $g''(x, y'') = g(x, g'(x', y''))$. \square

Die entscheidende Eigenschaft von L-Reduktionen ist, dass sie Approximierbarkeit erhalten:

Satz 16.38. (Papadimitriou und Yannakakis [1991]) Seien \mathcal{P} und \mathcal{P}' zwei Optimierungsprobleme mit nichtnegativen Gewichten. Sei ferner (f, g, α, β) eine L-Reduktion von \mathcal{P} auf \mathcal{P}' . Gibt es ein Approximationsschema für \mathcal{P}' , so auch für \mathcal{P} .

Beweis: Gegeben sei eine Instanz x von \mathcal{P} und eine Zahl $0 < \epsilon < 1$. Wir wenden nun das Approximationsschema für \mathcal{P}' auf $f(x)$ und $\epsilon' := \frac{\epsilon}{2\alpha\beta}$ an. Dabei erhalten wir eine zulässige Lösung y' von $f(x)$ und als Output schließlich eine zulässige Lösung $y := g(x, y')$ von x . Da

$$\begin{aligned} |c(x, y) - \text{OPT}(x)| &\leq \beta |c'(f(x), y') - \text{OPT}(f(x))| \\ &\leq \beta \max \left\{ (1 + \epsilon') \text{OPT}(f(x)) - \text{OPT}(f(x)), \right. \\ &\quad \left. \text{OPT}(f(x)) - \frac{1}{1 + \epsilon'} \text{OPT}(f(x)) \right\} \\ &\leq \beta \epsilon' \text{OPT}(f(x)) \\ &\leq \alpha \beta \epsilon' \text{OPT}(x) \\ &= \frac{\epsilon}{2} \text{OPT}(x), \end{aligned}$$

bekommen wir

$$c(x, y) \leq \text{OPT}(x) + |c(x, y) - \text{OPT}(x)| \leq \left(1 + \frac{\epsilon}{2}\right) \text{OPT}(x)$$

und

$$c(x, y) \geq \text{OPT}(x) - |\text{OPT}(x) - c(x, y)| \geq \left(1 - \frac{\epsilon}{2}\right) \text{OPT}(x) > \frac{1}{1+\epsilon} \text{OPT}(x).$$

Demnach bildet dies ein Approximationsschema für \mathcal{P} . \square

Dieser Satz und Satz 16.35 legen die folgende Definition nahe:

Definition 16.39. Ein Optimierungsproblem \mathcal{P} mit nichtnegativen Gewichten heißt **MAXSNP-schwer**, falls MAX-3SAT auf \mathcal{P} L-reduzierbar ist.

Der Name *MAXSNP* bezieht sich auf eine von Papadimitriou und Yannakakis [1991] definierte Klasse von Optimierungsproblemen. Wir benötigen diese Klasse hier nicht, weswegen wir ihre (nicht triviale) Definition weglassen.

Korollar 16.40. Gilt $P \neq NP$, so hat kein MAXSNP-schweres Problem ein Approximationsschema.

Beweis: Der Beweis folgt unmittelbar aus den Sätzen 16.35 und 16.38. \square

Wir werden für einige Probleme mittels L-Reduktion zeigen, dass sie MAXSNP-schwer sind. Wir beginnen mit einer eingeschränkten Variante von MAX-3SAT:

3-OCCURRENCE-MAX-SAT-PROBLEM

Instanz: Eine Menge X von Variablen und eine Familie \mathcal{Z} von Klauseln über X , jede mit höchstens drei Literalen, so dass jede Variable in höchstens drei Klauseln vorkommt.

Aufgabe: Bestimme eine Wahrheitsbelegung T von X , so dass die Anzahl der von T erfüllten Klauseln in \mathcal{Z} maximal ist.

Dass 3-OCCURRENCE MAX-SAT *NP*-schwer ist, lässt sich mittels einer einfachen Transformation von 3SAT (oder MAX-3SAT) beweisen, siehe Aufgabe 11, Kapitel 15. Da diese Transformation keine L-Reduktion ist, folgt daraus nicht, dass 3-OCCURRENCE MAX-SAT MAXSNP-schwer ist. Dafür benötigen wir eine kompliziertere Konstruktion mittels sogenannter Expandergraphen:

Definition 16.41. Sei G ein ungerichteter Graph und $\gamma > 0$ eine Konstante. Es ist G ein γ -Expander, falls $|\Gamma(A)| \geq \gamma |A|$ für jedes $A \subseteq V(G)$ mit $|A| \leq \frac{|V(G)|}{2}$.

Es ist z. B. ein vollständiger Graph ein 1-Expander. Uns interessieren jedoch Expander mit einer geringen Anzahl von Kanten. Wir zitieren den folgenden Satz, führen hier aber nicht seinen recht komplizierten Beweis:

Satz 16.42. (Ajtai [1994]) Es gibt eine positive Konstante γ , so dass man für jede gegebene gerade ganze Zahl $n \geq 4$ einen 3-regulären γ -Expander mit n Knoten in $O(n^3 \log^2 n)$ -Zeit konstruieren kann.

Das folgende Korollar wurde von Papadimitriou [1994] erwähnt (und benutzt), und ein korrekter Beweis wurde von Fernández-Baca und Lagergren [1998] angegeben:

Korollar 16.43. *Für jede gegebene ganze Zahl $n \geq 3$ kann ein Digraph G mit $O(n)$ Knoten und eine Menge $S \subseteq V(G)$ der Kardinalität n mit den folgenden Eigenschaften in $O(n^3 \log^3 n)$ -Zeit konstruiert werden:*

$$|\delta^-(v)| + |\delta^+(v)| \leq 3 \text{ für jedes } v \in V(G);$$

$$|\delta^-(v)| + |\delta^+(v)| = 2 \text{ für jedes } v \in S; \text{ und}$$

$$|\delta^+(A)| \geq \min\{|S \cap A|, |S \setminus A|\} \text{ für jedes } A \subseteq V(G).$$

Beweis: Sei $\gamma > 0$ die Konstante aus Satz 16.42 und $k := \left\lceil \frac{1}{\gamma} \right\rceil$. Zunächst konstruieren wir einen 3-regulären γ -Expander H mit n oder $n+1$ Knoten mittels Satz 16.42.

Wir ersetzen jede Kante $\{v, w\}$ durch k parallele Kanten (v, w) und k parallele Kanten (w, v) . Es sei H' der resultierende Digraph. Beachte, dass für jedes $A \subseteq V(H')$ mit $|A| \leq \frac{|V(H')|}{2}$ Folgendes gilt:

$$|\delta_{H'}^+(A)| = k|\delta_H(A)| \geq k|\Gamma_H(A)| \geq k\gamma|A| \geq |A|.$$

Analog gilt für jedes $A \subseteq V(H')$ mit $|A| > \frac{|V(H')|}{2}$:

$$\begin{aligned} |\delta_{H'}^+(A)| &= k|\delta_H(V(H') \setminus A)| \geq k|\Gamma_H(V(H') \setminus A)| \\ &\geq k\gamma|V(H') \setminus A| \geq |V(H') \setminus A|. \end{aligned}$$

Somit haben wir in beiden Fällen $|\delta_{H'}^+(A)| \geq \min\{|A|, |V(H') \setminus A|\}$.

Nun spalten wir jeden Knoten $v \in V(H')$ auf in $6k+1$ Knoten $x_{v,i}$ ($i = 0, \dots, 6k$), so dass jeder Knoten außer $x_{v,0}$ den Grad 1 hat. Für jeden Knoten $x_{v,i}$ fügen wir nun die Knoten $w_{v,i,j}$ und $y_{v,i,j}$ ($j = 0, \dots, 6k$) hinzu, wobei diese durch Kanten zu einem Weg der Länge $12k+2$ und in der Reihenfolge $w_{v,i,0}, w_{v,i,1}, \dots, w_{v,i,6k}, x_{v,i}, y_{v,i,0}, \dots, y_{v,i,6k}$ verbunden sind. Schließlich fügen wir noch die Kanten $(y_{v,i,j}, w_{v,j,i})$ für alle $v \in V(H')$, alle $i \in \{0, \dots, 6k\}$ und alle $j \in \{0, \dots, 6k\} \setminus \{i\}$ hinzu.

Insgesamt haben wir somit eine Knotenmenge Z_v der Kardinalität $(6k+1)(12k+3)$ für jedes $v \in V(H')$. Der endgültig resultierende Graph G hat $|V(H')|(6k+1)(12k+3) = O(n)$ Knoten, jeder mit Grad 2 oder 3. Gemäß der Konstruktion besitzt $G[Z_v]$ für jedes Paar disjunkter Teilmengen X_1, X_2 von $\{x_{v,i} : i = 0, \dots, 6k\}$ genau $\min\{|X_1|, |X_2|\}$ paarweise knotendisjunkte Wege von X_1 nach X_2 . Wir wählen eine n -elementige Teilmenge S von $\{x_{v,0} : v \in V(H')\}$; beachte, dass es für jeden dieser Knoten eine dort beginnende und eine dort endende Kante gibt.

Es bleibt zu zeigen, dass $|\delta^+(A)| \geq \min\{|S \cap A|, |S \setminus A|\}$ für jedes $A \subseteq V(G)$. Der Beweis folgt mittels Induktion über $|\{v \in V(H') : \emptyset \neq A \cap Z_v \neq Z_v\}|$. Ist diese Zahl gleich Null, d. h. $A = \bigcup_{v \in B} Z_v$ für ein $B \subseteq V(H')$, so folgt

$$|\delta_G^+(A)| = |\delta_{H'}^+(B)| \geq \min\{|B|, |V(H') \setminus B|\} \geq \min\{|S \cap A|, |S \setminus A|\}.$$

Anderenfalls sei $v \in V(H')$ mit $\emptyset \neq A \cap Z_v \neq Z_v$. Sei $P := \{x_{v,i} : i = 0, \dots, 6k\} \cap A$ und $Q := \{x_{v,i} : i = 0, \dots, 6k\} \setminus A$. Ist $|P| \leq 3k$, so folgt mit der Eigenschaft von $G[Z_v]$, dass

$$\begin{aligned} |E_G^+(Z_v \cap A, Z_v \setminus A)| &\geq |P| = |P \setminus S| + |P \cap S| \\ &\geq |E_G^+(A \setminus Z_v, A \cap Z_v)| + |P \cap S|. \end{aligned}$$

Nach der Induktionsvoraussetzung, angewendet auf $A \setminus Z_v$, folgt sodann

$$\begin{aligned} |\delta_G^+(A)| &\geq |\delta_G^+(A \setminus Z_v)| - |E_G^+(A \setminus Z_v, A \cap Z_v)| + |E_G^+(Z_v \cap A, Z_v \setminus A)| \\ &\geq |\delta_G^+(A \setminus Z_v)| + |P \cap S| \\ &\geq \min\{|S \cap (A \setminus Z_v)|, |S \setminus (A \setminus Z_v)|\} + |P \cap S| \\ &\geq \min\{|S \cap A|, |S \setminus A|\}. \end{aligned}$$

Analog haben wir: Ist $|P| \geq 3k + 1$, so ist $|Q| \leq 3k$ und mit der Eigenschaft von $G[Z_v]$ folgt

$$\begin{aligned} |E_G^+(Z_v \cap A, Z_v \setminus A)| &\geq |Q| = |Q \setminus S| + |Q \cap S| \\ &\geq |E_G^+(Z_v \setminus A, V(G) \setminus (A \cup Z_v))| + |Q \cap S|. \end{aligned}$$

Nach der Induktionsvoraussetzung, angewendet auf $A \cup Z_v$, folgt sodann

$$\begin{aligned} |\delta_G^+(A)| &\geq |\delta_G^+(A \cup Z_v)| \\ &\quad - |E_G^+(Z_v \setminus A, V(G) \setminus (A \cup Z_v))| + |E_G^+(Z_v \cap A, Z_v \setminus A)| \\ &\geq |\delta_G^+(A \cup Z_v)| + |Q \cap S| \\ &\geq \min\{|S \cap (A \cup Z_v)|, |S \setminus (A \cup Z_v)|\} + |Q \cap S| \\ &\geq \min\{|S \cap A|, |S \setminus A|\}. \end{aligned}$$

□

Wir können nun den folgenden Satz beweisen:

Satz 16.44. (Papadimitriou und Yannakakis [1991], Papadimitriou [1994], Fernández-Baca und Lagergren [1998]) *Das 3-OCCURRENCE-MAX-SAT-PROBLEM ist MAXSNP-schwer.*

Beweis: Der Beweis erfolgt mittels einer L-Reduktion (f, g, α, β) von MAX-3SAT. Zur Definition von f sei (X, \mathcal{Z}) eine Instanz von MAX-3SAT. Für jede Variable $x \in X$, die in mehr als drei, etwa in k Klauseln, vorkommt, ändern wir die Instanz folgendermaßen. In jeder dieser k Klauseln ersetzen wir x durch eine neue noch nicht vorhandene Variable und bekommen somit die neuen Variablen x_1, \dots, x_k . Ferner fügen wir zusätzliche Nebenbedingungen (und auch weitere Variablen) hinzu, deren Zweck es ist, sicherzustellen, dass es vorteilhaft ist, den Variablen x_1, \dots, x_k sämtlich denselben Wahrheitswert zuzuteilen.

Wir konstruieren G und S wie in Korollar 16.43 und nennen die Knoten so um, dass $S = \{1, \dots, k\}$. Für jeden Knoten $v \in V(G) \setminus S$ führen wir eine neue Variable x_v ein und für jede Kante $(v, w) \in E(G)$ eine Klausel $\{\bar{x}_v, x_w\}$. Insgesamt ist die Anzahl der hinzugefügten neuen Klauseln höchstens gleich

$$\frac{3}{2}(k+1) \left(6 \left\lceil \frac{1}{\gamma} \right\rceil + 1 \right) \left(12 \left\lceil \frac{1}{\gamma} \right\rceil + 3 \right) \leq 315 \left\lceil \frac{1}{\gamma} \right\rceil^2 k,$$

wobei γ wiederum die Konstante aus Satz 16.42 ist.

Führen wir die obige Substitution für jede Variable durch, so erhalten wir eine Instanz $(X', \mathcal{Z}') = f(X, \mathcal{Z})$ des 3-OCCURRENCE MAX-SAT-PROBLEMS mit

$$|\mathcal{Z}'| \leq |\mathcal{Z}| + 315 \left\lceil \frac{1}{\gamma} \right\rceil^2 3|\mathcal{Z}| \leq 946 \left\lceil \frac{1}{\gamma} \right\rceil^2 |\mathcal{Z}|.$$

Daraus folgt

$$\text{OPT}(X', \mathcal{Z}') \leq |\mathcal{Z}'| \leq 946 \left\lceil \frac{1}{\gamma} \right\rceil^2 |\mathcal{Z}| \leq 1892 \left\lceil \frac{1}{\gamma} \right\rceil^2 \text{OPT}(X, \mathcal{Z}),$$

da mindestens die Hälfte der Klauseln einer MAX-SAT-Instanz erfüllt werden können (indem wir entweder alle Variablen auf *true*, oder alle auf *false* setzen).

Somit können wir $\alpha := 1892 \left\lceil \frac{1}{\gamma} \right\rceil^2$ nehmen.

Zur Beschreibung von g , sei T' eine Wahrheitsbelegung von X' . Zunächst konstruieren wir eine Wahrheitsbelegung T'' von X' , die mindestens so viele Klauseln von \mathcal{Z}' erfüllt wie T' und die alle neuen Klauseln erfüllt (die den Kanten der obigen Graphen G entsprechen). Dazu gehen wir folgendermaßen vor. Für jede Variable x , die mehr als dreimal in (X, \mathcal{Z}) vorkommt, sei G der oben konstruierte Graph und $A := \{v \in V(G) : T'(x_v) = \text{true}\}$. Ist $|S \cap A| \geq |S \setminus A|$, so setzen wir $T''(x_v) := \text{true}$ für alle $v \in V(G)$, sonst setzen wir $T''(x_v) := \text{false}$ für alle $v \in V(G)$. Offensichtlich werden alle neuen Klauseln (die den Kanten entsprechen) erfüllt.

Es gibt höchstens $\min\{|S \cap A|, |S \setminus A|\}$ alte Klauseln, die von T' , aber nicht von T'' , erfüllt werden. Andererseits erfüllt T' keine der Klauseln $\{\bar{x}_v, x_w\}$ für $(v, w) \in \delta_G^+(A)$. Mittels der Eigenschaften von G ist die Anzahl dieser Klauseln mindestens gleich $\min\{|S \cap A|, |S \setminus A|\}$.

Nun liefert T'' eine Wahrheitsbelegung $T = g(X, \mathcal{Z}, T')$ von X auf nahe liegende Weise: Setze $T(x) := T''(x) = T'(x)$ für $x \in X \cap X'$, und setze $T(x) := T''(x_i)$, falls x_i eine x ersetzzende Variable in der Konstruktion von (X, \mathcal{Z}) nach (X', \mathcal{Z}') ist.

Die Wahrheitsbelegung T verletzt so viele Klauseln wie T'' . Bezeichnet $c(X, \mathcal{Z}, T)$ die Anzahl der von T erfüllten Klauseln der Instanz (X, \mathcal{Z}) , so folgt also

$$|\mathcal{Z}| - c(X, \mathcal{Z}, T) = |\mathcal{Z}'| - c(X', \mathcal{Z}', T'') \leq |\mathcal{Z}'| - c(X', \mathcal{Z}', T'). \quad (16.7)$$

Andererseits führt jede Wahrheitsbelegung T von X zu einer Wahrheitsbelegung T' von X' , die dieselbe Anzahl von Klauseln verletzt (indem wir für jede Variable x und den entsprechenden Graphen G in obiger Konstruktion die Variablen x_v ($v \in V(G)$) gleichmäßig auf $T(x)$ setzen). Somit gilt

$$|\mathcal{Z}| - \text{OPT}(X, \mathcal{Z}) \geq |\mathcal{Z}'| - \text{OPT}(X', \mathcal{Z}'). \quad (16.8)$$

Kombinieren wir (16.7) und (16.8), so erhalten wir

$$\begin{aligned} |\text{OPT}(X, \mathcal{Z}) - c(X, \mathcal{Z}, T)| &= (|\mathcal{Z}| - c(X, \mathcal{Z}, T)) - (|\mathcal{Z}| - \text{OPT}(X, \mathcal{Z})) \\ &\leq \text{OPT}(X', \mathcal{Z}') - c(X', \mathcal{Z}', T') \\ &= |\text{OPT}(X', \mathcal{Z}') - c(X', \mathcal{Z}', T')|, \end{aligned}$$

wobei $T = g(X, \mathcal{Z}, T')$. Also ist $(f, g, \alpha, 1)$ tatsächlich eine L-Reduktion. \square

Obiges Resultat bildet den Ausgangspunkt einiger Beweise dafür, dass gewisse Probleme MAXSNP-schwer sind, z. B.

Korollar 16.45. (Papadimitriou und Yannakakis [1991]) *Das MAXIMUM-STABLE-SET-PROBLEM, beschränkt auf Graphen mit maximalem Grad 4, ist MAXSNP-schwer.*

Beweis: Die Konstruktion im Beweis von Satz 15.23 definiert eine L-Reduktion von dem 3-OCCURRENCE MAX-SAT-PROBLEM auf das MAXIMUM-STABLE-SET-PROBLEM beschränkt auf Graphen mit maximalem Grad 4: Für jede Instanz (X, \mathcal{Z}) wird ein Graph G konstruiert, so dass man von jeder Wahrheitsbelegung, die k Klauseln erfüllt, leicht eine stabile Menge der Kardinalität k erhält, und umgekehrt. \square

In der Tat ist das MAXIMUM-STABLE-SET-PROBLEM sogar dann MAXSNP-schwer, wenn es auf 3-reguläre Graphen beschränkt wird (Berman und Fujito [1999]). Andererseits ist ein einfacher Greedy-Algorithmus, der schrittweise einen Knoten v mit minimalem Grad wählt und ihn und all seine Nachbarn entfernt, ein $\frac{(k+2)}{3}$ -Approximationsalgorithmus für das MAXIMUM-STABLE-SET-PROBLEM in Graphen mit maximalem Grad k (Halldórsson und Radhakrishnan [1997]). Für $k = 4$ ergibt dies die Approximationsgüte 2, bedeutend besser als die Approximationsgüte 8, die wir durch den folgenden Beweis erreichen (unter Verwendung des 2-Approximationsalgorithmus für das MINIMUM-VERTEX-COVER-PROBLEM).

Satz 16.46. (Papadimitriou und Yannakakis [1991]) *Das auf Graphen mit maximalem Grad 4 beschränkte MINIMUM-VERTEX-COVER-PROBLEM ist MAXSNP-schwer.*

Beweis: Betrachte die triviale Transformation vom MAXIMUM-STABLE-SET-PROBLEM (Proposition 2.2) mit $f(G) := G$ und $g(G, X) := V(G) \setminus X$ für alle Graphen G und alle $X \subseteq V(G)$. Obwohl dies im Allgemeinen keine L-Reduktion ist, ist es eine L-Reduktion, wenn wir uns auf Graphen mit maximalem Grad 4 beschränken, wie wir jetzt zeigen werden.

Hat G maximalen Grad 4, so gibt es eine stabile Menge mit Kardinalität mindestens gleich $\frac{|V(G)|}{5}$. Bezeichnen wir mit $\alpha(G)$ die maximale Kardinalität einer stabilen Menge und mit $\tau(G)$ die minimale Kardinalität einer Knotenüberdeckung, so folgt

$$\alpha(G) \geq \frac{1}{4}(|V(G)| - \alpha(G)) = \frac{1}{4}\tau(G)$$

und $\alpha(G) - |X| = |V(G) \setminus X| - \tau(G)$ für jede stabile Menge $X \subseteq V(G)$. Somit ist $(f, g, 4, 1)$ eine L-Reduktion. \square

Siehe Clementi und Trevisan [1999] und Chlebík und Chlebíková [2006] für stärkere Aussagen. Insbesondere gibt es kein Approximationsschema für das MINIMUM-VERTEX-COVER-PROBLEM (sofern $P \neq NP$). In späteren Kapiteln werden wir beweisen, dass einige weitere Probleme MAXSNP-schwer sind; siehe auch Aufgabe 23.

Aufgaben

1. Man formuliere einen 2-Approximationsalgorithmus für das folgende Problem. In einem gegebenen Digraphen mit Kantengewichten bestimme man einen azyklischen Teilgraphen maximalen Gewichtes.
Bemerkung: Ein k -Approximationsalgorithmus mit $k < 2$ ist für dieses Problem nicht bekannt.
2. Das k -CENTER-PROBLEM wird wie folgt definiert: Für einen gegebenen ungerichteten Graphen G mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ und einer Zahl $k \in \mathbb{N}$ mit $k \leq |V(G)|$ bestimme man eine Menge $X \subseteq V(G)$ der Kardinalität k , so dass

$$\max_{v \in V(G)} \min_{x \in X} \text{dist}(v, x)$$

minimal ist. Wie üblich, wird der optimale Wert mit $\text{OPT}(G, c, k)$ bezeichnet.

- (a) Sei S eine inklusionsmaximale stabile Menge in $(V(G), \{\{v, w\} : \text{dist}(v, w) \leq 2R\})$. Man zeige, dass $\text{OPT}(G, c, |S| - 1) > R$.
- (b) Man verwende (a), um einen 2-Approximationsalgorithmus für das k -CENTER-PROBLEM zu beschreiben.
(Hochbaum und Shmoys [1985])
- * (c) Man zeige, dass es für kein $r < 2$ einen r -Approximationsalgorithmus für das k -CENTER-PROBLEM gibt.
Hinweis: Man verweise Aufgabe 14, Kapitel 15.
(Hsu und Nemhauser [1979])
- 3. Kann man eine kardinalitätsminimale Knotenüberdeckung (oder eine kardinalitätsmaximale stabile Menge) in einem bipartiten Graphen in polynomieller Zeit finden?
- 4. Man zeige, dass die Approximationsgüte in Satz 16.5 die bestmögliche ist.

5. Man zeige, dass das folgende Verfahren ein 2-Approximationsalgorithmus für das MINIMUM-VERTEX-COVER-PROBLEM ist: Man berechne einen DFS-Baum und liefere als Output alle Knoten des DFS-Baums mit nicht verschwindendem Ausgangsgrad.
(Bar-Yehuda [unveröffentlicht])
6. Man zeige, dass die LP-Relaxierung $\min\{cx : M^\top x \geq \mathbb{1}, x \geq 0\}$ des MINIMUM-WEIGHT-VERTEX-COVER-PROBLEMS, wobei M die Inzidenzmatrix eines ungerichteten Graphen und $c \in \mathbb{R}_+^{V(G)}$ ist, immer eine halbganzzahlige optimale Lösung hat (d. h. eine optimale Lösung, deren Komponenten nur 0, $\frac{1}{2}$, oder 1 sind). Man leite einen weiteren 2-Approximationsalgorithmus aus dieser Tatsache ab. Benutzen Sie die Aussage außerdem, um in Satz 15.45 die Schranke auf $|V(G')| \leq 2l$ zu verbessern.
- * 7. Man betrachte das MINIMUM-WEIGHT-FEEDBACK-VERTEX-SET-PROBLEM: Man bestimme für einen ungerichteten Graphen G mit Gewichten $c : V(G) \rightarrow \mathbb{R}_+$ eine Knotenmenge $X \subseteq V(G)$ minimalen Gewichtes, so dass $G - X$ ein Wald ist. Man betrachte den folgenden rekursiven Algorithmus A :
Ist $E(G) = \emptyset$, so sei der Output $A(G, c) := \emptyset$. Ist $|\delta_G(x)| \leq 1$ für ein $x \in V(G)$, so sei der Output $A(G, c) := A(G - x, c)$. Ist $c(x) = 0$ für ein $x \in V(G)$, so sei der Output $A(G, c) := \{x\} \cup A(G - x, c)$. Andernfalls setze man

$$\epsilon := \min_{x \in V(G)} \frac{c(v)}{|\delta(v)|}$$

und $c'(v) := c(v) - \epsilon |\delta(v)|$ ($v \in V(G)$). Sei $X := A(G, c')$. Für jedes $x \in X$ setze man $X := X \setminus \{x\}$, falls $G - (X \setminus \{x\})$ ein Wald ist, und der Output sei $A(G, c) := x$.

Man beweise, dass dies ein 2-Approximationsalgorithmus für das MINIMUM-WEIGHT-FEEDBACK-VERTEX-SET-PROBLEM ist.

(Becker und Geiger [1996])

8. Man zeige, dass das MAXIMUM-CUT-PROBLEM sogar für einfache Graphen NP-schwer ist.
9. Man beweise, dass der am Anfang von Abschnitt 16.2 beschriebene einfache Greedy-Algorithmus für MAX-CUT ein 2-Approximationsalgorithmus ist.
10. Man betrachte den folgenden lokalen Suchalgorithmus für das MAXIMUM-CUT-PROBLEM. Man beginne mit einer beliebigen nichtleeren, echten Teilmenge S von $V(G)$. Nun prüfe man iterativ, ob irgendein Knoten zu S hinzugefügt oder aus S entfernt werden kann, so dass $|\delta(S)|$ wächst. Ist eine solche Verbesserung nicht möglich, so terminiere man.
- (a) Man beweise, dass das obige Verfahren ein 2-Approximationsalgorithmus ist. (Beachte Aufgabe 14, Kapitel 2.)
 - (b) Kann der Algorithmus auf das MAXIMUM-WEIGHT-CUT-PROBLEM mit nichtnegativen Kantengewichten erweitert werden?
 - (c) Findet der Algorithmus immer eine optimale Lösung für planare bzw. für bipartite Graphen? Für beide Klassen gibt es einen polynomiellen Algorithmus (Aufgabe 7, Kapitel 12, und Proposition 2.27).

11. Das GERICHTETE MAXIMUM-WEIGHT-CUT-PROBLEM lautet: Für einen gegebenen Digraphen G mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ bestimme man eine Menge $X \subseteq V(G)$ mit $\sum_{e \in \delta^+(X)} c(e)$ maximal. Man zeige, dass es einen 4-Approximationsalgorithmus für dieses Problem gibt.

Hinweis: Man verwende Aufgabe 10.

Bemerkung: Es gibt einen 1,165-, aber keinen 1,09-Approximationsalgorithmus, sofern $P \neq NP$ (Feige und Goemans [1995], Håstad [2001]).

12. Man gebe für das ungewichtete Pentagon C_5 eine Lösung von (16.2) an. Welche Lösungen kann der GOEMANS-WILLIAMSON-MAX-CUT ALGORITHMUS für diese Instanz ausgeben?

13. Man zeige, dass $(\frac{1}{\pi} \arccos(y_i^\top y_j))_{1 \leq i, j \leq n}$ eine Konvexitätskombination von Schnitt-Semimetriken δ^R , $R \subseteq \{1, \dots, n\}$ ist, wobei $\delta_{i,j}^R = 1$, falls $|R \cap \{i, j\}| = 1$, und $\delta_{i,j}^R = 0$ sonst.

Hinweis: Man schreibe

$$(\mu(H(y_i) \triangle H(y_j)))_{1 \leq i, j \leq n} = \sum_{R \subseteq \{1, \dots, n\}} \mu\left(\bigcap_{i \in R} H(y_i) \setminus \bigcup_{i \notin R} H(y_i)\right) \delta^R.$$

Bemerkung: Siehe Deza und Laurent [1997] für viele weitere Informationen.

14. Man zeige, dass es für jedes $n \in \mathbb{N}$ einen bipartiten Graphen mit $2n$ Knoten gibt, für den der GREEDY-FÄRBUNGSALGORITHMUS n Farben benötigt. Somit kann dieser Algorithmus beliebig schlechte Ergebnisse liefern. Man zeige jedoch, dass es immer eine Reihenfolge der Knoten gibt, für die der Algorithmus eine optimale Färbung findet.

15. Man zeige, dass man jeden mit drei Farben färbbaren Graphen G mit höchstens $2\sqrt{2n}$ Farben in polynomieller Zeit färben kann, wobei $n := |V(G)|$.

Hinweis: So lange es einen Knoten v mit Grad mindestens gleich $\sqrt{2n}$ gibt, färbe man $\Gamma(v)$ optimal mit höchstens zwei Farben (die man nicht nochmals verwendet) und entferne diesen Knoten. Abschließend wende man den GREEDY-FÄRBUNGSALGORITHMUS an.

(Wigderson [1983])

16. Man zeige, dass die folgenden Graphen perfekt sind:

- (a) bipartite Graphen;
- (b) Intervallgraphen: $(\{v_1, \dots, v_n\}, \{\{v_i, v_j\} : i \neq j, [a_i, b_i] \cap [a_j, b_j] \neq \emptyset\})$, wobei $[a_1, b_1], \dots, [a_n, b_n]$ abgeschlossene Intervalle sind;
- (c) chordale Graphen (siehe Aufgabe 38, Kapitel 8).

- * 17. Sei G ein ungerichteter Graph. Man beweise, dass die folgenden sechs Aussagen äquivalent sind:

- (a) Es ist G perfekt.
- (b) Für jede Gewichtsfunktion $c : V(G) \rightarrow \mathbb{Z}_+$ ist das maximale Gewicht einer Clique in G gleich der minimalen Anzahl stabiler Mengen, so dass jeder Knoten v von G in genau $c(v)$ von ihnen enthalten ist.

- (c) Für jede Gewichtsfunktion $c : V(G) \rightarrow \mathbb{Z}_+$ ist das maximale Gewicht einer stabilen Menge in G gleich der minimalen Anzahl von Cliquen, so dass jeder Knoten v von G in genau $c(v)$ von ihnen enthalten ist.

(d) Das (16.3) definierende lineare Ungleichungssystem ist TDI.

- (e) Das Cliques-Polytop von G , d. h. die konvexe Hülle der Inzidenzvektoren aller Cliques in G , wird gegeben durch

$$\left\{ x \in \mathbb{R}_+^{V(G)} : \sum_{v \in S} x_v \leq 1 \text{ für alle stabilen Mengen } S \text{ in } G \right\}. \quad (16.9)$$

- (f) Das (16.9) definierende lineare Ungleichungssystem ist TDI.

Bemerkung: Das Polytop (16.9) heißt der Antiblocker des Polytops (16.3).

18. Eine Instanz von MAX-SAT heißt k -erfüllbar, falls jede Menge von k Klauseln gleichzeitig erfüllt werden kann. Sei r_k das Infimum desjenigen Bruchteils der Klauseln, den man in einer k -erfüllbaren Instanz erfüllen kann.

(a) Man beweise, dass $r_1 = \frac{1}{2}$.

(b) Man beweise, dass $r_2 = \frac{\sqrt{5}-1}{2}$.

(Hinweis: Manche der Variablen kommen in ein-elementigen Klauseln vor (o. B. d. A. können wir annehmen, dass alle ein-elementigen Klauseln positiv sind); man setze diese mit Wahrscheinlichkeit a (für ein a mit $\frac{1}{2} < a < 1$) auf *true* und die anderen Variablen mit Wahrscheinlichkeit $\frac{1}{2}$ auf *true*. Man wende die Derandomisierungsmethode an und wähle ein geeignetes a .)

(c) Man beweise, dass $r_3 \geq \frac{2}{3}$.

(Lieberherr und Specker [1981])

19. Erdős [1967] hat Folgendes gezeigt: Für jede Konstante $k \in \mathbb{N}$ ist der (asymptotisch) beste Bruchteil der Kanten, der mit Sicherheit in einem Schnitt maximalen Gewichtes liegt, gleich $\frac{1}{2}$, selbst wenn wir uns auf Graphen ohne ungerade Kreise der Länge höchstens gleich k beschränken; (siehe Aufgabe 10(a).)

(a) Was kann man über $k = \infty$ sagen?

(b) Man zeige, wie das MAXIMUM-CUT-PROBLEM auf MAX-SAT reduziert werden kann.

Hinweis: Man benutze eine Variable pro Knoten und zwei Klauseln $\{x, y\}, \{\bar{x}, \bar{y}\}$ pro Kante $\{x, y\}$.

(c) Man verwende (b) und den obigen Satz von Erdős, um zu beweisen, dass $r_k \leq \frac{3}{4}$ für alle k . (Es wurde r_k in Aufgabe 18 definiert.)

Bemerkung: Trevisan [2004] hat bewiesen, dass $\lim_{k \rightarrow \infty} r_k = \frac{3}{4}$.

20. Man beweise, dass die Fehlerwahrscheinlichkeit $\frac{1}{2}$ in Definition 16.31 durch jede beliebige Zahl zwischen 0 und 1 äquivalent ersetzt werden kann. Man leite hieraus (und aus dem Beweis von Satz 16.34) ab, dass es für kein $\rho \geq 1$ einen ρ -Approximationsalgorithmus für das MAXIMUM-CLIQUE-PROBLEM gibt (sofern $P \neq NP$).

21. Man beweise, dass das MAXIMUM-CLIQUE-PROBLEM auf das SET-PACKING-PROBLEM L-reduzierbar ist: Für ein gegebenes Mengensystem (U, \mathcal{S}) finde

- man eine Untermenge $\mathcal{R} \subseteq \mathcal{S}$ maximaler Kardinalität, deren Elemente paarweise disjunkt sind.
22. Man beweise, dass es für das MINIMUM-VERTEX-COVER-PROBLEM keinen absoluten Approximationsalgorithmus gibt (sofern $P \neq NP$).
 23. Man beweise: MAX-2SAT ist MAXSNP-schwer.
Hinweis: Man verwende Korollar 16.45.
 (Papadimitriou und Yannakakis [1991])

Literatur

Allgemeine Literatur:

- Asano, T., Iwama, K., Takada, H., und Yamashita, Y. [2000]: Designing high-quality approximation algorithms for combinatorial optimization problems. IEICE Transactions on Communications/Electronics/Information and Systems E83-D (2000), 462–478
- Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., und Protasi, M. [1999]: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer, Berlin 1999
- Garey, M.R., und Johnson, D.S. [1979]: Computers and Intractability; A Guide to the Theory of NP -Completeness. Freeman, San Francisco 1979, Kapitel 4
- Hochbaum, D.S. [1996]: Approximation Algorithms for NP -Hard Problems. PWS, Boston, 1996
- Horowitz, E., und Sahni, S. [1978]: Fundamentals of Computer Algorithms. Computer Science Press, Potomac 1978, Kapitel 12
- Shmoys, D.B. [1995]: Computing near-optimal solutions to combinatorial optimization problems. In: Combinatorial Optimization; DIMACS Series in Discrete Mathematics and Theoretical Computer Science 20 (W.J. Cook, L. Lovász, P. Seymour, Hrsg.), AMS, Providence 1995
- Papadimitriou, C.H. [1994]: Computational Complexity. Addison-Wesley, Reading 1994, Kapitel 13
- Vazirani, V.V. [2001]: Approximation Algorithms. Springer, Berlin, 2001
- Williamson, D.P., und Shmoys, D.B. [2011]: The Design of Approximation Algorithms. Cambridge University Press, Cambridge 2011

Zitierte Literatur:

- Ajtai, M. [1994]: Recursive construction for 3-regular expanders. Combinatorica 14 (1994), 379–416
- Alizadeh, F. [1995]: Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM Journal on Optimization 5 (1995), 13–51
- Appel, K., und Haken, W. [1977]: Every planar map is four colorable; Part I; Discharging. Illinois Journal of Mathematics 21 (1977), 429–490
- Appel, K., Haken, W., und Koch, J. [1977]: Every planar map is four colorable; Part II; Reducibility. Illinois Journal of Mathematics 21 (1977), 491–567
- Arora, S. [1994]: Probabilistic checking of proofs and the hardness of approximation problems. Ph.D. thesis, U.C. Berkeley, 1994
- Arora, S., und Kale, S. [2016]: A combinatorial, primal-dual approach to semidefinite programs. Journal of the ACM 63 (2016), Article 12

- Arora, S., Lund, C., Motwani, R., Sudan, M., und Szegedy, M. [1998]: Proof verification and hardness of approximation problems. *Journal of the ACM* 45 (1998), 501–555
- Arora, S., und Safra, S. [1998]: Probabilistic checking of proofs. *Journal of the ACM* 45 (1998), 70–122
- Asano, T. [2006]: An improved analysis of Goemans und Williamson's LP-relaxation for MAX SAT. *Theoretical Computer Science* 354 (2006), 339–353
- Avidor, A., Berkovitch, I., und Zwick, U. [2006]: Improved approximation algorithms for MAX NAE-SAT and MAX SAT. *Approximation and Online Algorithms – Proceedings of the 3rd WAOA workshop (2005); LNCS 3879* (Erlebach, T., Persiano, G., Hrsg.), Springer, Berlin 2006, pp. 27–40
- Bar-Yehuda, R., und Even, S. [1981]: A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms* 2 (1981), 198–203
- Becker, A., und Geiger, D. [1996]: Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence Journal* 83 (1996), 1–22
- Bellare, M., und Sudan, M. [1994]: Improved non-approximability results. *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing* (1994), 184–193
- Bellare, M., Goldreich, O., und Sudan, M. [1998]: Free bits, PCPs and nonapproximability – towards tight results. *SIAM Journal on Computing* 27 (1998), 804–915
- Berge, C. [1961]: Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind. *Wissenschaftliche Zeitschrift, Martin Luther Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche Reihe* (1961), 114–115
- Berge, C. [1962]: Sur une conjecture relative au problème des codes optimaux. *Communication, 13ème assemblée générale de l'URSI*, Tokyo 1962
- Berman, P., und Fujito, T. [1999]: On approximation properties of the independent set problem for low degree graphs. *Theory of Computing Systems* 32 (1999), 115–132
- Brooks, R.L. [1941]: On colouring the nodes of a network. *Proceedings of the Cambridge Philosophical Society* 37 (1941), 194–197
- Chen, J., Friesen, D.K., und Zheng, H. [1999]: Tight bound on Johnson's algorithm for maximum satisfiability. *Journal of Computer and System Sciences* 58 (1999), 622–640
- Chlebík, M. und Chlebíková, J. [2006]: Complexity of approximating bounded variants of optimization problems. *Theoretical Computer Science* 354 (2006), 320–338
- Chudnovsky, M., Cornuéjols, G., Liu, X., Seymour, P., und Vušković, K. [2005]: Recognizing Berge graphs. *Combinatorica* 25 (2005), 143–186
- Chudnovsky, M., Robertson, N., Seymour, P., und Thomas, R. [2006]: The strong perfect graph theorem. *Annals of Mathematics* 164 (2006), 51–229
- Chvátal, V. [1975]: On certain polytopes associated with graphs. *Journal of Combinatorial Theory B* 18 (1975), 138–154
- Chvátal, V. [1979]: A greedy heuristic for the set cover problem. *Mathematics of Operations Research* 4 (1979), 233–235
- Clementi, A.E.F., und Trevisan, L. [1999]: Improved non-approximability results for minimum vertex cover with density constraints. *Theoretical Computer Science* 225 (1999), 113–128
- Deza, M.M., und Laurent, M. [1997]: *Geometry of Cuts and Metrics*. Springer, Berlin 1997
- Dinur, I. [2007]: The PCP theorem by gap amplification. *Journal of the ACM* 54 (2007), Article 12
- Dinur, I., und Safra, S. [2002]: On the hardness of approximating minimum vertex cover. *Annals of Mathematics* 162 (2005), 439–485

- Erdős, P. [1967]: On bipartite subgraphs of graphs. *Mat. Lapok.* 18 (1967), 283–288
- Feige, U. [1998]: A threshold of $\ln n$ for approximating set cover. *Journal of the ACM* 45 (1998), 634–652
- Feige, U. [2004]: Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics* 18 (2004), 219–225
- Feige, U., und Goemans, M.X. [1995]: Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. *Proceedings of the 3rd Israel Symposium on Theory of Computing and Systems* (1995), 182–189
- Feige, U., Goldwasser, S., Lovász, L., Safra, S., und Szegedy, M. [1996]: Interactive proofs and the hardness of approximating cliques. *Journal of the ACM* 43 (1996), 268–292
- Fernández-Baca, D., und Lagergren, J. [1998]: On the approximability of the Steiner tree problem in phylogeny. *Discrete Applied Mathematics* 88 (1998), 129–145
- Fulkerson, D.R. [1972]: Anti-blocking polyhedra. *Journal of Combinatorial Theory B* 12 (1972), 50–71
- Fürer, M., und Raghavachari, B. [1994]: Approximating the minimum-degree Steiner tree to within one of optimal. *Journal of Algorithms* 17 (1994), 409–423
- Garey, M.R., Johnson, D.S., und Stockmeyer, L. [1976]: Some simplified *NP*-complete graph problems. *Theoretical Computer Science* 1 (1976), 237–267
- Goemans, M.X., und Williamson, D.P. [1994]: New $3/4$ -approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics* 7 (1994), 656–666
- Goemans, M.X., und Williamson, D.P. [1995]: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM* 42 (1995), 1115–1145
- Grötschel, M., Lovász, L., und Schrijver, A. [1988]: *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin 1988
- Halldórsson, M.M., und Radhakrishnan, J. [1997]: Greed is good: approximating independent sets in sparse and bounded degree graphs. *Algorithmica* 18 (1997), 145–163
- Håstad, J. [2001]: Some optimal inapproximability results. *Journal of the ACM* 48 (2001), 798–859
- Heawood, P.J. [1890]: Map colour theorem. *Quarterly Journal of Pure Mathematics* 24 (1890), 332–338
- Held, S., Cook, W., und Sewell, E.C. [2012]: Maximum-weight stable sets and safe lower bounds for graph coloring. *Mathematical Programming Computation* 4 (2012), 363–381
- Hochbaum, D.S. [1982]: Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing* 11 (1982), 555–556
- Hochbaum, D.S., und Shmoys, D.B. [1985]: A best possible heuristic for the k -center problem. *Mathematics of Operations Research* 10 (1985), 180–184
- Holyer, I. [1981]: The *NP*-completeness of edge-coloring. *SIAM Journal on Computing* 10 (1981), 718–720
- Hougardy, S., Prömel, H.J., und Steger, A. [1994]: Probabilistically checkable proofs and their consequences for approximation algorithms. *Discrete Mathematics* 136 (1994), 175–223
- Hsu, W.L., und Nemhauser, G.L. [1979]: Easy and hard bottleneck location problems. *Discrete Applied Mathematics* 1 (1979), 209–216
- Johnson, D.S. [1974]: Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences* 9 (1974), 256–278
- Khanna, S., Linial, N., und Safra, S. [2000]: On the hardness of approximating the chromatic number. *Combinatorica* 20 (2000), 393–415

- Khot, S., und Regev, O. [2008]: Vertex cover might be hard to approximate to within $2 - \epsilon$. *Journal of Computer and System Sciences* 74 (2008), 335–349
- Knuth, D.E. [1969]: *The Art of Computer Programming*; Vol. 2. Seminumerical Algorithms. Addison-Wesley, Reading 1969 (3. Aufl., 1997)
- König, D. [1916]: Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Mathematische Annalen* 77 (1916), 453–465
- Lieberherr, K., und Specker, E. [1981]: Complexity of partial satisfaction. *Journal of the ACM* 28 (1981), 411–421
- Lovász, L. [1972]: Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics* 2 (1972), 253–267
- Lovász, L. [1975]: On the ratio of optimal integral and fractional covers. *Discrete Mathematics* 13 (1975), 383–390
- Lovász, L. [1979a]: On the Shannon capacity of a graph. *IEEE Transactions on Information Theory* 25 (1979), 1–7
- Lovász, L. [1979b]: Graph theory and integer programming. In: *Discrete Optimization I; Annals of Discrete Mathematics* 4 (P.L. Hammer, E.L. Johnson, B.H. Korte, Hrsg.), North-Holland, Amsterdam 1979, pp. 141–158
- Lovász, L. [2003]: Semidefinite programs and combinatorial optimization. In: *Recent Advances in Algorithms and Combinatorics* (B.A. Reed, C. Linhares Sales, Hrsg.), Springer, New York (2003), pp. 137–194
- Mahajan, S., und Ramesh, H. [1999]: Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal on Computing* 28 (1999), 1641–1663
- Papadimitriou, C.H., und Steiglitz, K. [1982]: *Combinatorial Optimization; Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs 1982, pp. 406–408
- Papadimitriou, C.H., und Yannakakis, M. [1991]: Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences* 43 (1991), 425–440
- Papadimitriou, C.H., und Yannakakis, M. [1993]: The traveling salesman problem with distances one and two. *Mathematics of Operations Research* 18 (1993), 1–12
- Raghavan, P., und Thompson, C.D. [1987]: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica* 7 (1987), 365–374
- Raz, R., und Safra, S. [1997]: A sub constant error probability low degree test, and a sub constant error probability PCP characterization of NP. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (1997), 475–484
- Robertson, N., Sanders, D.P., Seymour, P., und Thomas, R. [1997]: The four colour theorem. *Journal of Combinatorial Theory B* 70 (1997), 2–44
- Sanders, P., und Steurer, D. [2008]: An asymptotic approximation scheme for multigraph edge coloring. *ACM Transactions on Algorithms* 4 (2008), Article 21
- Singh, M., und Lau, L.C. [2015]: Approximating minimum bounded degree spanning trees to within one of optimal. *Journal of the ACM* 62 (2015), Article 1
- Slavík, P. [1997]: A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms* 25 (1997), 237–254
- Stockmeyer, L.J. [1973]: Planar 3-colorability is polynomial complete. *ACM SIGACT News* 5 (1973), 19–25
- Trevisan, L. [2004]: On local versus global satisfiability. *SIAM Journal on Discrete Mathematics* 17 (2004), 541–547
- Vizing, V.G. [1964]: On an estimate of the chromatic class of a p-graph. *Diskret. Analiz* 3 (1964), 23–30 [auf Russisch]
- Wigderson, A. [1983]: Improving the performance guarantee for approximate graph coloring. *Journal of the ACM* 30 (1983), 729–735

- Yannakakis, M. [1994]: On the approximation of maximum satisfiability. *Journal of Algorithms* 17 (1994), 475–502
- Zuckerman, D. [2007]: Linear degree extractors and the inapproximability of Max Clique and Chromatic Number. *Theory of Computing* 3 (2007), 103–128



17 Das Knapsack-Problem

Das MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM und das GEWICHTETE MATROID-INTERSEKTIONS-PROBLEM, die beide in vorausgegangenen Kapiteln besprochen worden sind, gehören zu den „schwersten“ Problemen, für die ein polynomieller Algorithmus bekannt ist. In diesem Kapitel werden wir uns mit dem folgenden Problem befassen, welches sich in einem gewissen Sinne als das „leichteste“ *NP*-schwere Problem herausstellen wird:

KNAPSACK-PROBLEM

Instanz: Nichtnegative ganze Zahlen $n, c_1, \dots, c_n, w_1, \dots, w_n$ und W .

Aufgabe: Bestimme eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} w_j \leq W$, die $\sum_{j \in S} c_j$ maximiert.

Dieses Problem liegt immer dann vor, wenn man eine optimale Teilmenge mit beschränktem Gesamtgewicht aus einer Menge gewichteter und Gewinn bringender Elemente wählen möchte.

Zunächst werden wir in Abschnitt 17.1 die sogenannte gebrochene Version betrachten, die in linearer Zeit lösbar ist. Das ganzzahlige Knapsack-Problem ist *NP*-schwer, wie wir in Abschnitt 17.2 zeigen werden, aber es kann mit einem pseudopolynomiellen Algorithmus optimal gelöst werden. Dieser kann mit einem Rundungsverfahren so kombiniert werden, dass wir ein voll-polynomielles Approximationsschema erhalten; dies zeigen wir in Abschnitt 17.3. In Abschnitt 17.4 werden wir eine multidimensionale Verallgemeinerung besprechen. In Abschnitt 17.5 betrachten wir schließlich einen Algorithmus, der in der Praxis effizient ist, und erklären, warum das so ist.

17.1 Das gebrochene Knapsack-Problem und das gewichtete Median-Problem

Wir betrachten das folgende Problem:

GEBROCHENES KNAPSACK-PROBLEM

Instanz: Nichtnegative ganze Zahlen $n, c_1, \dots, c_n, w_1, \dots, w_n$ und W .

Aufgabe: Bestimme Zahlen $x_1, \dots, x_n \in [0, 1]$ mit $\sum_{j=1}^n x_j w_j \leq W$, die $\sum_{j=1}^n x_j c_j$ maximieren.

Das folgende Resultat legt einen einfachen Lösungsalgorithmus nahe, der eine geeignete Sortierung der Elemente gebraucht:

Proposition 17.1. (Dantzig [1957]) *Seien $c_1, \dots, c_n, w_1, \dots, w_n$ und W nicht-negative ganze Zahlen mit $\sum_{i=1}^n w_i > W$, $\{1 \leq i \leq n : w_i = 0\} = \{1, \dots, h\}$ und*

$$\frac{c_{h+1}}{w_{h+1}} \geq \frac{c_{h+2}}{w_{h+2}} \geq \dots \geq \frac{c_n}{w_n},$$

und setze

$$k := \min \left\{ j \in \{1, \dots, n\} : \sum_{i=1}^j w_i > W \right\}.$$

Dann ist der folgende Vektor x eine optimale Lösung der gegebenen Instanz des GEBROCHENEN KNAPSACK-PROBLEMS:

$$\begin{aligned} x_j &:= 1 && \text{für } j = 1, \dots, k-1, \\ x_k &:= \frac{W - \sum_{j=1}^{k-1} w_j}{w_k}, \\ x_j &:= 0 && \text{für } j = k+1, \dots, n. \end{aligned}$$

□

Die Sortierung der Elemente benötigt $O(n \log n)$ -Zeit (Satz 1.5), und die Berechnung von k erfolgt in $O(n)$ -Zeit mittels einfachen linearen Scannings. Obwohl dieser Algorithmus recht schnell ist, geht es noch besser. Beachte, dass das Problem auf die Suche eines gewichteten Medians zurückgeführt werden kann:

Definition 17.2. *Sei $n \in \mathbb{N}$, $z_1, \dots, z_n \in \mathbb{R}$, $w_1, \dots, w_n \in \mathbb{R}_+$ und $W \in \mathbb{R}$ mit $0 < W \leq \sum_{i=1}^n w_i$. Der $(w_1, \dots, w_n; W)$ -gewichtete Median bezüglich (z_1, \dots, z_n) ist die eindeutig definierte Zahl z^* mit*

$$\sum_{i:z_i < z^*} w_i < W \leq \sum_{i:z_i \leq z^*} w_i.$$

Somit müssen wir das folgende Problem lösen:

GEWICHTETES MEDIAN-PROBLEM

Instanz: Eine natürliche Zahl n , Zahlen $z_1, \dots, z_n \in \mathbb{R}$, $w_1, \dots, w_n \in \mathbb{R}_+$ und eine Zahl W mit $0 < W \leq \sum_{i=1}^n w_i$.

Aufgabe: Bestimme den $(w_1, \dots, w_n; W)$ -gewichteten Median bezüglich (z_1, \dots, z_n) .

Ein wichtiger Spezialfall ist folgender:

AUSWAHLPROBLEM

Instanz: Eine natürliche Zahl n , Zahlen $z_1, \dots, z_n \in \mathbb{R}$ und eine ganze Zahl $k \in \{1, \dots, n\}$.

Aufgabe: Bestimme die k -kleinste Zahl unter den Zahlen z_1, \dots, z_n , genauer: einen Index $i \in \{1, \dots, n\}$ mit $|\{j : z_j < z_i\}| < k \leq |\{j : z_j \leq z_i\}|$.

Der gewichtete Median kann in $O(n)$ -Zeit bestimmt werden: Der folgende Algorithmus ist eine Version des Algorithmus von Blum et al. [1973] für den gewichteten Fall; siehe auch Vygen [1997].

GEWICHTETER MEDIAN-ALGORITHMUS

Input: Eine natürliche Zahl n , Zahlen $z_1, \dots, z_n \in \mathbb{R}$, $w_1, \dots, w_n \in \mathbb{R}_+$ und eine Zahl W mit $0 < W \leq \sum_{i=1}^n w_i$.

Output: Der $(w_1, \dots, w_n; W)$ -gewichtete Median bezüglich (z_1, \dots, z_n) .

- ① Partitioniere die Liste z_1, \dots, z_n in 5-elementige Blöcke (der letzte Block kann weniger Elemente enthalten). Bestimme (rekursiv) den (ungewichteten) Median eines jeden Blocks. Sei M die Liste dieser $\lceil \frac{n}{5} \rceil$ Mediane.
- ② Bestimme (rekursiv) den ungewichteten Median z_m von M .
- ③ Vergleiche jedes Element der Liste z_1, \dots, z_n mit z_m . O. B. d. A. können wir annehmen, dass $z_i < z_m$ für $i = 1, \dots, k$, $z_i = z_m$ für $i = k+1, \dots, l$ und $z_i > z_m$ für $i = l+1, \dots, n$.
- ④ **If** $\sum_{i=1}^k w_i < W \leq \sum_{i=1}^l w_i$ **then stop** ($z^* := z_m$).
If $\sum_{i=1}^l w_i < W$ **then** bestimme den $(w_{l+1}, \dots, w_n; W - \sum_{i=1}^l w_i)$ -gewichteten Median rekursiv bezüglich (z_{l+1}, \dots, z_n) . **Stop**.
If $\sum_{i=1}^k w_i \geq W$ **then** bestimme den $(w_1, \dots, w_k; W)$ -gewichteten Median rekursiv bezüglich (z_1, \dots, z_k) . **Stop**.

Satz 17.3. Der GEWICHTETE MEDIAN-ALGORITHMUS arbeitet korrekt und benötigt nur $O(n)$ -Zeit.

Beweis: Die Korrektheit lässt sich leicht prüfen. Sei nun $f(n)$ die Worst-Case-Laufzeit für n Elemente. Dann gilt

$$f(n) = O(n) + f\left(\lceil \frac{n}{5} \rceil\right) + O(n) + f\left(\frac{1}{2}\lceil \frac{n}{5} \rceil 5 + \frac{1}{2}\lceil \frac{n}{5} \rceil 2\right),$$

da der rekursive Aufruf in ④ mindestens drei Elemente aus mindestens der Hälfte der 5-elementigen Blöcke auslässt. Obige Rekursionsformel ergibt $f(n) = O(n)$:

Da $\lceil \frac{n}{5} \rceil \leq \frac{9}{41}n$ für alle $n \geq 37$, erhalten wir $f(n) \leq cn + f\left(\frac{9}{41}n\right) + f\left(\frac{7}{2}\frac{9}{41}n\right)$ für ein geeignetes c und $n \geq 37$. Damit lässt sich nun $f(n) \leq (82c + f(36))n$ leicht mittels Induktion beweisen. Somit ist die Laufzeit in der Tat linear. \square

Daraus folgen sofort die beiden Korollare:

Korollar 17.4. (Blum et al. [1973]) *Das AUSWAHLPROBLEM kann in $O(n)$ -Zeit gelöst werden.*

Beweis: Setze $w_i := 1$ für $i = 1, \dots, n$ und $W := k$ und wende Satz 17.3 an. \square

Korollar 17.5. *Das GEBROCHENE KNAPSACK-PROBLEM kann in linearer Zeit gelöst werden.*

Beweis: Das GEBROCHENE KNAPSACK-PROBLEM wird auf das GEWICHTETE MEDIAN-PROBLEM dadurch zurückgeführt, dass wir $z_i := -\frac{c_i}{w_i}$ ($i = 1, \dots, n$) setzen. \square

17.2 Ein pseudopolynomieller Algorithmus

Wir wenden uns nun dem (ganzzahligen) KNAPSACK-PROBLEM zu. Die im vorigen Abschnitt eingeführten Methoden erweisen sich auch hier als nützlich:

Proposition 17.6. *Seien $c_1, \dots, c_n, w_1, \dots, w_n$ und W nichtnegative ganze Zahlen mit $w_j \leq W$ für $j = 1, \dots, n$, $\sum_{i=1}^n w_i > W$, und sei*

$$\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \dots \geq \frac{c_n}{w_n}.$$

Sei ferner

$$k := \min \left\{ j \in \{1, \dots, n\} : \sum_{i=1}^j w_i > W \right\}.$$

Dann ergibt die Wahl der besseren der beiden zulässigen Lösungen $\{1, \dots, k-1\}$ und $\{k\}$ einen 2-Approximationsalgorithmus für das KNAPSACK-PROBLEM mit $O(n)$ -Laufzeit.

Beweis: Für jede gegebene Instanz des KNAPSACK-PROBLEMS sind die Elemente $i \in \{1, \dots, n\}$ mit $w_i > W$ nutzlos und können vorab entfernt werden. Gilt nun $\sum_{i=1}^n w_i \leq W$, so ist $\{1, \dots, n\}$ eine optimale Lösung. Andernfalls berechnen wir die Zahl k in $O(n)$ -Zeit ohne zu sortieren: Dies ist nichts anderes als ein GEWICHTETES MEDIAN-PROBLEM wie oben (Satz 17.3).

Nach Proposition 17.1 ist $\sum_{i=1}^k c_i$ eine obere Schranke für den Optimalwert des GEBROCHENEN KNAPSACK-PROBLEMS und somit auch für den Optimalwert des ganzzahligen KNAPSACK-PROBLEMS. Damit erreicht die bessere der beiden zulässigen Lösungen $\{1, \dots, k-1\}$ und $\{k\}$ mindestens die Hälfte des Optimalwertes. \square

Wir sind jedoch mehr an einer exakten Lösung des KNAPSACK-PROBLEMS interessiert, haben andererseits aber folgendes Resultat:

Satz 17.7. (Karp [1972]) *Das KNAPSACK-PROBLEM ist NP-schwer.*

Beweis: Wir werden beweisen, dass das folgende verwandte Entscheidungsproblem *NP*-vollständig ist: Gegeben seien nichtnegative ganze Zahlen $n, c_1, \dots, c_n, w_1, \dots, w_n, W$ und K . Gibt es eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} w_j \leq W$ und $\sum_{j \in S} c_j \geq K$?

Offensichtlich ist dieses Entscheidungsproblem in *NP*. Um zu zeigen, dass es *NP*-vollständig ist, zeigen wir, dass sich SUBSET-SUM darauf transformieren lässt (siehe Korollar 15.27). Für eine gegebene Instanz c_1, \dots, c_n, K von SUBSET-SUM definieren wir $w_j := c_j$ ($j = 1, \dots, n$) und $W := K$. Offensichtlich liefert dies eine äquivalente Instanz des obigen Entscheidungsproblems. \square

Da wir nicht bewiesen haben, dass das KNAPSACK-PROBLEM stark *NP*-schwer ist, besteht die Hoffnung, dass es einen pseudopolynomiellen Algorithmus gibt. Tatsächlich kann der im Beweis von Satz 15.39 gegebene Algorithmus leicht verallgemeinert werden, indem wir Kantengewichte einführen und ein Kürzeste-Wege-Problem lösen. Dies ergibt einen Algorithmus mit $O(nW)$ Laufzeit (Aufgabe 7).

Mit einem ähnlichen Trick können wir auch einen Algorithmus mit $O(nC)$ -Laufzeit erhalten, wobei $C := \sum_{j=1}^n c_j$. Wir werden diesen Algorithmus direkt beschreiben, ohne den Umweg über die Angabe eines Graphen und die Betrachtung von kürzesten Wegen. Da der Algorithmus auf einfachen Rekursionsformeln basiert, spricht man hier auch von dynamischer Optimierung. Der Algorithmus stammt im Grunde von Bellman [1956,1957] und Dantzig [1957].

EXAKTER KNAPSACK-ALGORITHMUS

Input: Nichtnegative ganze Zahlen $n, c_1, \dots, c_n, w_1, \dots, w_n$ und W .

Output: Eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} w_j \leq W$, die $\sum_{j \in S} c_j$ maximiert.

- ① Sei C eine beliebige obere Schranke für den Optimalwert,
z. B. $C := \sum_{j=1}^n c_j$.
 - ② Setze $x(0, 0) := 0$ und $x(0, k) := \infty$ für $k = 1, \dots, C$.
 - ③ **For** $j := 1$ **to** n **do:**
 - For** $k := 0$ **to** C **do:**
 - Setze $s(j, k) := 0$ und $x(j, k) := x(j - 1, k)$.
 - For** $k := c_j$ **to** C **do:**
 - If** $x(j - 1, k - c_j) + w_j \leq \min\{W, x(j, k)\}$ **then:**
 - Setze $x(j, k) := x(j - 1, k - c_j) + w_j$ und $s(j, k) := 1$.
- ④ Sei $k = \max\{i \in \{0, \dots, C\} : x(n, i) < \infty\}$. Setze $S := \emptyset$.
 - For** $j := n$ **down to** 1 **do:**
 - If** $s(j, k) = 1$ **then** setze $S := S \cup \{j\}$ und $k := k - c_j$.

Satz 17.8. *Der EXAKTE KNAPSACK-ALGORITHMUS bestimmt eine optimale Lösung in $O(nC)$ -Zeit.*

Beweis: Die Laufzeit ist klar. Die Variable $x(j, k)$ bezeichnet das minimale Gesamtgewicht einer Teilmenge $S \subseteq \{1, \dots, j\}$ mit $\sum_{i \in S} w_i \leq W$ und $\sum_{i \in S} c_i = k$. Der Algorithmus berechnet diese Werte korrekt mittels der folgenden Rekursionsformeln für $j = 1, \dots, n$ und $k = 0, \dots, C$:

$$x(j, k) = \begin{cases} x(j-1, k-c_j) + w_j & \text{für } c_j \leq k \text{ und} \\ & x(j-1, k-c_j) + w_j \leq \min\{W, x(j-1, k)\}, \\ x(j-1, k) & \text{sonst.} \end{cases}$$

Die Variablen $s(j, k)$ zeigen, welche dieser beiden Fälle zutrifft. Somit enumeriert der Algorithmus alle Teilmengen $S \subseteq \{1, \dots, n\}$, außer den unzulässigen und denen, die von anderen dominiert werden: Es wird S von S' dominiert, wenn $\sum_{j \in S} c_j = \sum_{j \in S'} c_j$ und $\sum_{j \in S} w_j \geq \sum_{j \in S'} w_j$. In ④ wird die beste zulässige Teilmenge gewählt. \square

Natürlich ist es wünschenswert, eine bessere obere Schranke C als $\sum_{i=1}^n c_i$ zur Verfügung zu haben. Z. B. könnte man den 2-Approximationsalgorithmus von Proposition 17.6 anwenden; verdoppelt man den Wert der gelieferten Lösung, so erhält man eine obere Schranke für den Optimalwert. Wir werden diese Idee später noch benutzen.

Die $O(nC)$ -Schranke ist nicht polynomiell in der Größe des Inputs, weil die Inputgröße nur durch $O(n \log C + n \log W)$ beschränkt werden kann (wir können hier annehmen, dass $w_j \leq W$ für alle j). Wir haben jedoch einen pseudopolynomiellen Algorithmus, der sich als recht effizient erweist, falls die vorkommenden Zahlen nicht zu groß sind. Sind sowohl die Gewichte w_1, \dots, w_n als auch die Gewinne c_1, \dots, c_n klein, so ist der $O(nc_{\max} w_{\max})$ -Algorithmus von Pisinger [1999] momentan der schnellste ($c_{\max} := \max\{c_1, \dots, c_n\}$, $w_{\max} := \max\{w_1, \dots, w_n\}$).

17.3 Ein voll-polynomielles Approximationsschema

In diesem Abschnitt untersuchen wir Approximationsalgorithmen für das KNAPSACK-PROBLEM. Nach Proposition 16.23 hat das KNAPSACK-PROBLEM keinen absoluten Approximationsalgorithmus, sofern $P \neq NP$.

Wir werden jedoch beweisen, dass es für das KNAPSACK-PROBLEM ein voll-polynomielles Approximationsschema gibt. Der erste solche Algorithmus stammt von Ibarra und Kim [1975] und verbesserte den Ansatz von Sahni [1975].

Da die Laufzeit des EXAKTEN KNAPSACK-ALGORITHMUS von C abhängt, liegt es nahe, alle Zahlen c_1, \dots, c_n zu halbieren und abzurunden. Dadurch wird die Laufzeit verringert, es können jetzt aber ungenaue Lösungen auftreten. Allgemeiner können wir die Laufzeit um den Faktor t verringern, indem wir

$$\bar{c}_j := \left\lfloor \frac{c_j}{t} \right\rfloor \quad (j = 1, \dots, n)$$

setzen. Genauigkeit gegen Laufzeitverringerung auszutauschen ist ein typisches Vorgehen bei Approximationsschemata. Ist $S \subseteq \{1, \dots, n\}$, so sei $c(S) := \sum_{i \in S} c_i$.

KNAPSACK-APPROXIMATIONSSCHEMA

Input: Nichtnegative ganze Zahlen $n, c_1, \dots, c_n, w_1, \dots, w_n$ und W . Eine Zahl $\epsilon > 0$.

Output: Eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} w_j \leq W$, so dass $c(S) \geq \frac{1}{1+\epsilon}c(S')$ für alle $S' \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S'} w_j \leq W$.

-
- ① Wende den 2-Approximationsalgorithmus von Proposition 17.6 an.
Sei S_1 die gelieferte Lösung. **If** $c(S_1) = 0$ **then** setze $S := S_1$ und **stop**.
 - ② Setze $t := \max \left\{ 1, \frac{\epsilon c(S_1)}{n} \right\}$.
Setze $\bar{c}_j := \lfloor \frac{c_j}{t} \rfloor$ für $j = 1, \dots, n$.
 - ③ Wende den EXAKTEN KNAPSACK-ALGORITHMUS auf die Instanz $(n, \bar{c}_1, \dots, \bar{c}_n, w_1, \dots, w_n, W)$ an; setze $C := \frac{2c(S_1)}{t}$. Sei S_2 die gelieferte Lösung.
 - ④ **If** $c(S_1) > c(S_2)$ **then** setze $S := S_1$, **else** setze $S := S_2$.
-

Satz 17.9. (Ibarra und Kim [1975], Gens und Levner [1979]) Das KNAPSACK-APPROXIMATIONSSCHEMA ist ein voll-polynomielles Approximationsschema für das KNAPSACK-PROBLEM mit Laufzeit $O\left(n^2 \cdot \frac{1}{\epsilon}\right)$.

Beweis: Terminiert der Algorithmus in ①, so ist S_1 optimal nach Proposition 17.6. Also nehmen wir an, dass $c(S_1) > 0$. Sei S^* eine optimale Lösung der ursprünglichen Instanz. Da nach Proposition 17.6 $2c(S_1) \geq c(S^*)$ gilt, ist C in ③ eine korrekte obere Schranke für den Wert der optimalen Lösung der gerundeten Instanz. Nach Satz 17.8 ist S_2 dann eine optimale Lösung der gerundeten Instanz. Somit haben wir:

$$\begin{aligned} \sum_{j \in S_2} c_j &\geq \sum_{j \in S_2} t\bar{c}_j = t \sum_{j \in S_2} \bar{c}_j \geq t \sum_{j \in S^*} \bar{c}_j = \sum_{j \in S^*} t\bar{c}_j > \sum_{j \in S^*} (c_j - t) \\ &\geq c(S^*) - nt. \end{aligned}$$

Ist $t = 1$, so ist S_2 optimal nach Satz 17.8. Andernfalls folgt aus der obigen Ungleichung, dass $c(S_2) \geq c(S^*) - \epsilon c(S_1)$, und daraus folgern wir, dass

$$(1 + \epsilon)c(S) \geq c(S_2) + \epsilon c(S_1) \geq c(S^*).$$

Demnach haben wir einen $(1 + \epsilon)$ -Approximationsalgorithmus für ein beliebiges festes $\epsilon > 0$. Nach Satz 17.8 wird die Laufzeit von ③ durch

$$O(nC) = O\left(\frac{nc(S_1)}{t}\right) = O\left(n^2 \cdot \frac{1}{\epsilon}\right)$$

beschränkt. Die weiteren Schritte können leicht in $O(n)$ -Zeit bewerkstelligt werden. \square

Lawler [1979] hat ein ähnliches voll-polynomielles Approximationsschema mit Laufzeit $O\left(n \log\left(\frac{1}{\epsilon}\right) + \frac{1}{\epsilon^4}\right)$ gefunden. Dieses wurde von Kellerer und Pferschy [2004] und Chan [2018] verbessert.

Leider gibt es nur wenige Probleme mit einem voll-polynomiellem Approximationsschema. Um dies zu präzisieren, betrachten wir das MAXIMIERUNGSPROBLEM FÜR UNABHÄNGIGKEITSSYSTEME.

Bei der Bildung des EXAKTEN KNAPSACK-ALGORITHMUS und des KNAPSACK-APPROXIMATIONSSCHEMAS haben wir eine bestimmte Dominanzrelation benutzt. Diese werden wir nun verallgemeinern.

Definition 17.10. Gegeben sei ein Unabhängigkeitssystem (E, \mathcal{F}) , eine Kostenfunktion $c : E \rightarrow \mathbb{Z}_+$, Teilmengen $S_1, S_2 \subseteq E$ und $\epsilon > 0$. Man sagt:

S_1 ϵ -dominiert S_2 , falls

$$\frac{1}{1+\epsilon} c(S_1) \leq c(S_2) \leq (1+\epsilon) c(S_1)$$

und es eine Basis B_1 mit $S_1 \subseteq B_1$ gibt, so dass für jede Basis B_2 mit $S_2 \subseteq B_2$ folgende Ungleichung gilt:

$$(1+\epsilon) c(B_1) \geq c(B_2).$$

ϵ -DOMINANZ-PROBLEM

Instanz: Ein Unabhängigkeitssystem (E, \mathcal{F}) , eine Kostenfunktion $c : E \rightarrow \mathbb{Z}_+$, eine Zahl $\epsilon > 0$ und zwei Teilmengen $S_1, S_2 \subseteq E$.

Frage: Stimmt die Aussage: S_1 ϵ -dominiert S_2 ?

Natürlich wird das Unabhängigkeitssystem durch ein Orakel gegeben, z.B. ein Unabhängigkeits-Orakel. Der EXAKTE KNAPSACK-ALGORITHMUS hat oft von 0-Dominanz Gebrauch gemacht. Es stellt sich heraus, dass die Existenz eines effizienten Algorithmus für das ϵ -DOMINANZ-PROBLEM unerlässlich ist für die Existenz eines voll-polynomielles Approximationsschemas.

Satz 17.11. (Korte und Schrader [1981]) Sei \mathcal{I} eine Familie von Unabhängigkeitssystemen. Sei \mathcal{I}' die Familie der Instanzen (E, \mathcal{F}, c) des MAXIMIERUNGSPROBLEMS FÜR UNABHÄNGIGKEITSSYSTEME mit $(E, \mathcal{F}) \in \mathcal{I}$ und $c : E \rightarrow \mathbb{Z}_+$, und \mathcal{I}'' die Familie der Instanzen $(E, \mathcal{F}, c, \epsilon, S_1, S_2)$ des ϵ -DOMINANZ-PROBLEMS mit $(E, \mathcal{F}) \in \mathcal{I}$.

Dann folgt: Es gibt genau dann ein voll-polynomielles Approximationsschema für das MAXIMIERUNGSPROBLEM FÜR UNABHÄNGIGKEITSSYSTEME beschränkt auf \mathcal{I}' , wenn es einen Algorithmus für das ϵ -DOMINANZ-PROBLEM beschränkt auf \mathcal{I}'' gibt, dessen Laufzeit durch ein Polynom in der Länge des Inputs und $\frac{1}{\epsilon}$ beschränkt ist.

Dass die Bedingung hinreichend ist, wird durch Verallgemeinerung des KNAPSACK-APPROXIMATIONSSCHEMAS bewiesen (Aufgabe 14). Der Beweis der Notwendigkeit ist recht kompliziert und wird hier nicht geführt. Die Schlussfolgerung

ist: Falls es überhaupt ein voll-polynomielles Approximationsschema gibt, so wird es eine Modifizierung des KNAPSACK-APPROXIMATIONSSCHEMAS tun. Für ein ähnliches Resultat siehe auch Woeginger [2000].

Um zu beweisen, dass es für ein bestimmtes Optimierungsproblem kein voll-polynomielles Approximationsschema gibt, ist der folgende Satz meist von größerem Nutzen:

Satz 17.12. (Garey und Johnson [1978]) *Ein stark NP-schweres Optimierungsproblem mit ganzzahliger Zielfunktion, für welches*

$$\text{OPT}(I) \leq p(\text{size}(I), \text{largest}(I))$$

für ein Polynom p und alle Instanzen I gilt, besitzt genau dann ein voll-polynomielles Approximationsschema, wenn $P = NP$.

Beweis: Angenommen, das Optimierungsproblem besitzt ein voll-polynomielles Approximationsschema. Dann wenden wir dieses mit

$$\epsilon = \frac{1}{p(\text{size}(I), \text{largest}(I)) + 1}$$

an und erhalten einen exakten pseudopolynomiellen Algorithmus. Nach Proposition 15.41 ist dies aber unmöglich, sofern $P \neq NP$. \square

Im folgenden Abschnitt betrachten wir ein Problem, für welches es einen pseudopolynomiellen Algorithmus aber kein voll-polynomielles Approximationsschema gibt.

17.4 Das multidimensionale Knapsack-Problem

In diesem gesamten Abschnitt betrachten wir $m \in \mathbb{N}$ als fest gegeben. Wir betrachten das folgende Problem:

m -DIMENSIONALES KNAPSACK-PROBLEM

Instanz: Eine Zahl $n \in \mathbb{N}$ und nichtnegative ganze Zahlen c_i , w_{ij} und W_j für $i = 1, \dots, n$ und $j = 1, \dots, m$.

Aufgabe: Bestimme eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit der Eigenschaft: $\sum_{i \in S} w_{ij} \leq W_j$ für alle $j = 1, \dots, m$ und $\sum_{i \in S} c_i$ ist maximal.

Sogar für diese Verallgemeinerung des KNAPSACK-PROBLEMS (das dem Fall $m = 1$ entspricht), gibt es einen pseudopolynomiellen Algorithmus, siehe Aufgabe 15. Diese Tatsache führt jedoch nicht zu einem voll-polynomiellen Approximationsschema:

Proposition 17.13. (Korte und Schrader [1981]) *Gilt $P \neq NP$, so gibt es für kein $m \geq 2$ ein voll-polynomielles Approximationsschema für das m -DIMENSIONALE KNAPSACK-PROBLEM.*

Beweis: Es genügt, den Fall $m = 2$ zu betrachten. Gegeben sei eine Instanz (P, Q, R, T) von 3DM, und es sei $T = \{t_1, \dots, t_n\} \subseteq P \times Q \times R$. Setze $k := |P| = |Q| = |R|$ und o. B. d. A. $P \cup Q \cup R = \{1, \dots, 3k\}$. Wir benutzen nun eine ähnliche Konstruktion wie im Beweis von Korollar 15.27. Für $i = 1, \dots, n$ setzen wir $t_i = (p, q, r) \in P \times Q \times R$; ferner setzen wir $c_i = 1$, $w_{i1} = (n+1)^p + (n+1)^q + (n+1)^r$ und $w_{i2} = (n+1)^{3k+1} - w_{i1}$. Schließlich setzen wir noch $W_1 = \sum_{p=1}^{3k} (n+1)^p$ und $W_2 = k(n+1)^{3k+1} - W_1$. Damit haben wir eine Instanz des 2-DIMENSIONALEN KNAPSACK-PROBLEMS definiert.

Gibt es ein 3-dimensionales Matching $\{t_i : i \in S\}$ (wobei $S \subseteq \{1, \dots, n\}$), so gilt $|S| = k$ und $\sum_{i \in S} w_{i1} = W_1$ und somit $\sum_{i \in S} w_{i2} = W_2$. Andererseits gilt: Jede zulässige Lösung $S \subseteq \{1, \dots, n\}$ mit $|S| = k$ für diese Instanz des 2-DIMENSIONALEN KNAPSACK-PROBLEMS erfüllt $\sum_{i \in S} w_{i1} \leq W_1$ und

$$\sum_{i \in S} w_{i1} = k(n+1)^{3k+1} - \sum_{i \in S} w_{i2} \geq k(n+1)^{3k+1} - W_2 = W_1.$$

Somit gibt es genau dann eine zulässige Lösung S mit $|S| = k$, wenn die 3DM-Instanz (U, V, W, T) eine Ja-Instanz ist.

Daraus schließen wir: Der Optimalwert dieser Instanz des 2-DIMENSIONALEN KNAPSACK-PROBLEMS ist k wenn I eine Ja-Instanz ist, und höchstens $k - 1$ sonst. Gäbe es ein voll-polynomielles Approximationsschema, so könnten wir in polynomieller Zeit eine $(1 + \epsilon)$ -Approximation für $\epsilon = \frac{1}{k}$ bestimmen und somit zwischen den beiden Fällen entscheiden. Damit könnten wir 3DM in polynomieller Zeit lösen. Da aber 3DM NP-vollständig ist (Satz 15.26), würde daraus $P = NP$ folgen. \square

Wenigstens gibt es aber ein Approximationsschema:

Satz 17.14. (Frieze und Clarke [1984]) *Für jedes feste $m \in \mathbb{N}$ und $\epsilon > 0$ gibt es einen $(1 + \epsilon)$ -Approximationsalgorithmus für das m -DIMENSIONALE KNAPSACK-PROBLEM.*

Beweis: Wir setzen $k := \lceil \frac{m}{\epsilon} \rceil$; dies ist eine Konstante. Wir enumerieren alle Teilmengen von $\{1, \dots, n\}$ mit weniger als k Elementen. Für jedes $S \subseteq \{1, \dots, n\}$ mit $|S| = k$ setzen wir $S^> := \{i \in \{1, \dots, n\} \setminus S : c_i > \min\{c_j : j \in S\}\}$ und lösen das LP

$$\max \left\{ cx \quad : \quad \sum_{i=1}^n w_{ij} x_i \leq W_j \quad (j = 1, \dots, m), \quad 0 \leq x \leq 1, \right. \\ \left. x_i = 1 \quad (i \in S), \quad x_i = 0 \quad (i \in S^>) \right\}.$$

Sei x eine optimale Basislösung. Da x mindestens n Nebenbedingungen des LP mit Gleichheit erfüllt, hat es höchstens m nicht-ganzzahlige Komponenten. Sei $S' := \{i \in \{1, \dots, n\} : x_i = 1\}$.

Aus allen so gefundenen Mengen S und S' wählen wir die beste zulässige Lösung als Output. Die Laufzeit wird durch das Lösen von $O(n^k)$ LPs mit $O(n)$ Variablen und Nebenbedingungen dominiert.

Entweder hat die optimale Lösung höchstens $k - 1$ Elemente (dann finden wir sie), oder sie entspricht einer zulässigen 0-1-Lösung z eines der obigen LPs, nämlich desjenigen mit der Eigenschaft: S enthält k seiner Elemente mit dem größten Profit. Sodann bestimmen wir eine Lösung S' (durch Abrunden einer optimalen Basislösung x), die nicht sehr viel schlechter ist, da $cz \leq cx = \sum_{i \in S'} c_i + \sum_{i=1}^n c_i (x_i - \lfloor x_i \rfloor) \leq \sum_{i \in S'} c_i + m \max\{c_i : i \in \{1, \dots, n\} \setminus (S \cup S^>)\} \leq \sum_{i \in S'} c_i + m \min\{c_i : i \in S\} \leq \sum_{i \in S'} c_i + \frac{m}{k} \sum_{i \in S} c_i \leq \sum_{i \in S'} c_i (1 + \frac{m}{k}) \leq \sum_{i \in S'} c_i (1 + \epsilon)$. \square

17.5 Der Nemhauser-Ullmann-Algorithmus

In diesem Abschnitt betrachten wir einen weiteren Algorithmus für das KNAPSACK-PROBLEM, der von Nemhauser und Ullmann [1969] stammt.

Gegeben sei eine Instanz $n, c_1, \dots, c_n, w_1, \dots, w_n$ und W des KNAPSACK-PROBLEMS und eine Teilmenge $S \subseteq \{1, \dots, n\}$. Wir schreiben $w(S) := \sum_{i \in S} w_i$ und $c(S) := \sum_{i \in S} c_i$.

Definition 17.15. Gegeben sei eine Instanz $n, c_1, \dots, c_n, w_1, \dots, w_n$ und W des KNAPSACK-PROBLEMS. Eine Teilmenge $S \subseteq \{1, \dots, n\}$ heißt **Pareto-optimal**, falls es keine andere Teilmenge $S' \subseteq \{1, \dots, n\}$ mit $c(S) \leq c(S')$ und $w(S) \geq w(S')$ gibt, wobei mindestens eine dieser beiden Ungleichungen streng erfüllt ist.

In diesem Sinne wird eine Teilmenge genau dann dominiert, wenn sie nicht Pareto-optimal ist. Pareto-Optimalität ist ein wichtiger Begriff in der Optimierung mit mehreren Zielfunktionen. Der DYNAMISCHE-OPTIMIERUNGS-KNAPSACK-ALGORITHMUS enumeriert Pareto-optimale Lösungen für die aus den ersten j Elementen (für $j = 1, \dots, n$) bestehende Teilinstanz. Man kann im Wesentlichen dasselbe auf eine etwas andere Art bewerkstelligen, indem man sich die folgende Tatsache zunutze macht:

Lemma 17.16. Sei \mathcal{P}_j für $j \in \{1, \dots, n\}$ die Menge aller Pareto-optimalen Mengen der auf die Elemente $1, \dots, j$ reduzierten Instanz. Setze $\mathcal{P}_0 := \{\emptyset\}$. Dann haben wir $S \setminus \{j\} \in \mathcal{P}_{j-1}$ für jedes $j = 1, \dots, n$ und jedes $S \in \mathcal{P}_j$.

Beweis: Wird $S \setminus \{j\}$ durch irgendein $S' \subseteq \{1, \dots, j-1\}$ dominiert, so wird S durch $S' \cup (\{j\} \cap S)$ dominiert. \square

Der folgende Algorithmus berechnet die oben definierten Mengen $\mathcal{P}_0, \dots, \mathcal{P}_n$ und speichert sie als sortierte Listen.

NEMHAUSER-ULLMANN-ALGORITHMUS

Input: Nichtnegative ganze Zahlen $n, c_1, \dots, c_n, w_1, \dots, w_n$ und W .

Output: Eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} w_j \leq W$ und maximalem $\sum_{j \in S} c_j$.

① Sei $\mathcal{P}_0 := \{\emptyset\}$.

② **For** $j := 1$ **to** n **do:**

Setze $\mathcal{P}_{j-1}^+ := \{S \cup \{j\} : S \in \mathcal{P}_{j-1}\}$ und $\mathcal{P}_j := \emptyset$.

Sei S_1 das erste Element in \mathcal{P}_{j-1} und S_2 das erste in \mathcal{P}_{j-1}^+ .

While $S_1 \neq$ Ende oder $S_2 \neq$ Ende **do:**

If $(w(S_1) \leq w(S_2))$ oder $(w(S_1) = w(S_2)$ und $c(S_1) \geq c(S_2))$
(wobei $w(\text{Ende}) := \infty$)

then:

If $\mathcal{P}_j = \emptyset$ oder $c(S_1) > c(S)$ **then**

setze $S := S_1$ und füge S zu \mathcal{P}_j hinzu.

Sei S_1 das nächste Element in \mathcal{P}_{j-1} (oder $S_1 =$ Ende, falls es keines gibt).

else:

If $\mathcal{P}_j = \emptyset$ oder $c(S_2) > c(S)$ **then**

setze $S := S_2$ und füge S zu \mathcal{P}_j hinzu.

Sei S_2 das nächste Element in \mathcal{P}_{j-1}^+ (oder $S_2 =$ Ende, falls es keines gibt).

③ Gib $S \in \mathcal{P}_n$ mit $\sum_{i \in S} w_i \leq W$ und maximalem $\sum_{i \in S} c_i$ aus.

Abbildung 17.1 illustriert dies (die leeren Kreise stellen Elemente aus \mathcal{P}_{j-1} dar, die vollen Kreise Elemente aus \mathcal{P}_{j-1}^+ ; die beiden durchgekreuzten Lösungen werden dominiert und deswegen nicht zu \mathcal{P}_j hinzugefügt).

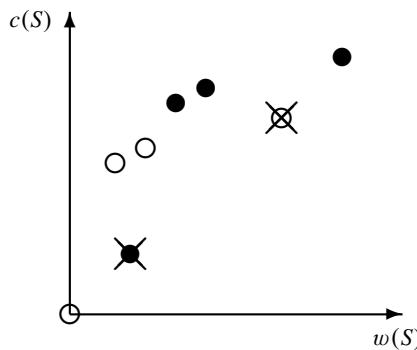


Abbildung 17.1.

Satz 17.17. *Der NEMHAUSER-ULLMANN-ALGORITHMUS arbeitet korrekt und kann mit $O(\sum_{i=0}^{n-1} |\mathcal{P}_i|)$ Laufzeit implementiert werden.*

Beweis: Die Listen werden stets so gespeichert, dass S genau dann vor S' steht, wenn $w(S) < w(S')$. Die **While**-Schleife durchläuft die Elemente von $\mathcal{P}_{j-1} \cup \mathcal{P}_{j-1}^+$ in dieser Reihenfolge und entfernt dominante Elemente; der Rest bildet dann \mathcal{P}_j . Die Korrektheit folgt nun mit Lemma 17.16.

Die Laufzeit kann erreicht werden, indem man die Elemente von \mathcal{P}_j nicht explizit speichert. Stattdessen speichert man $c(S)$, $w(S)$ und einen Bezug auf $S \setminus \{j\}$ in \mathcal{P}_{j-1} , und man merkt sich, ob j in S enthalten ist oder nicht. Daraus folgt, dass die Gesamlaufzeit zu $\sum_{i=0}^n |\mathcal{P}_i|$ proportional ist. \square

Dieser Algorithmus ist natürlich nicht polynomiell (siehe Aufgabe 16). Er funktioniert aber oft sehr gut in der Praxis. Dies kann zum Teil durch geglättete Analyse (im Englischen *smoothed analysis*) erklärt werden, indem man die Instanz zufällig stört:

Satz 17.18. (Beier, Rögl und Vöcking [2007]) *Gegeben sei ein $\epsilon > 0$. Ferner sei $n, c_1, \dots, c_n, w_1^{\min}, \dots, w_n^{\min}$ und W eine Instanz des KNAPSACK-PROBLEMS mit $w_i^{\min} \leq (1 - \epsilon)W$ für alle $i = 1, \dots, n$. Man betrachte eine zufällig gewählte Instanz, für die man unabhängig für jedes $i = 1, \dots, n$ ein $w_i \in [w_i^{\min}, w_i^{\min} + \epsilon W]$ (gleichmäßig verteilt) wählt. Dann ist die erwartete Anzahl Pareto-optimaler Lösungen höchstens $1 + \frac{n^2}{\epsilon}$.*

Beweis: Sei \mathcal{P} die Menge der Pareto-optimalen Lösungen. Dann gilt

$$\text{Exp}(|\mathcal{P}|) = 1 + \lim_{k \rightarrow \infty} \sum_{i=1}^k \text{Prob} \left(\exists S \in \mathcal{P} : (i-1)\frac{nW}{k} < w(S) \leq i\frac{nW}{k} \right), \quad (17.1)$$

da die leere Menge Pareto-optimal ist und das Gewicht 0 hat, und es mit wachsendem k extrem unwahrscheinlich wird, dass sich die Gewichte zweier Lösungen um weniger als $\frac{nW}{k}$ unterscheiden.

Für eine feste Zahl t definieren wir

$$c^*(t) := \max \{c(S^*) : S^* \subseteq \{1, \dots, n\}, w(S^*) \leq t\} \quad (17.2)$$

und

$$\Lambda(t) := \min \{w(S) - t : S \subseteq \{1, \dots, n\}, c(S) > c^*(t)\}. \quad (17.3)$$

Offensichtlich ist $\Lambda(t)$ stets positiv. Wir beweisen nun die

Behauptung: Für alle $t \geq 0$ und $\delta > 0$ gilt $\text{Prob}[\Lambda(t) \leq \delta] \leq \frac{n\delta}{\epsilon W}$.

Bevor wir die Behauptung beweisen, zeigen wir, dass der Satz leicht aus ihr abgeleitet werden kann: Für jedes i gibt es genau dann ein $S \in \mathcal{P}$ mit $(i-1)\frac{nW}{k} < w(S) \leq i\frac{nW}{k}$, wenn $\Lambda((i-1)\frac{nW}{k}) \leq \frac{nW}{k}$, und die Behauptung impliziert nun, dass dies mit einer Wahrscheinlichkeit passiert, die höchstens $\frac{n}{\epsilon W} \cdot \frac{nW}{k} = \frac{n^2}{\epsilon k}$ beträgt. Setzen wir dies in (17.1) ein, so folgt der Beweis.

Um die Behauptung zu beweisen, definieren wir für $h = 1, \dots, n$:

$$c^{*,h}(t) := \max \{c(S^*) : S^* \subseteq \{1, \dots, n\} \setminus \{h\}, w(S^*) \leq t\}$$

und

$$\Lambda^h(t) := \min \left\{ w(S) - t : h \in S \subseteq \{1, \dots, n\}, c(S) > c^{*,h}(t) \right\}. \quad (17.4)$$

Da es für alle Mengen S^* bzw. S , die (17.2) bzw. (17.3) erfüllen, einen Index $h \in S \setminus S^*$ gibt, folgt $\Lambda^h(t) = \Lambda(t)$ für wenigstens ein h .

Für feste $w_1, \dots, w_{h-1}, w_{h+1}, \dots, w_n$ folgt, dass $c^{*,h}(t)$ fest ist, und ferner, dass eine Menge S , die das Minimum in (17.4) liefert, unabhängig von der Zufallsvariable w_h ist. Somit folgt: $\Lambda^h(t) \leq \delta$ gilt nur dann, wenn $t - w(S \setminus \{h\}) < w_h \leq t + \delta - w(S \setminus \{h\})$. Die Wahrscheinlichkeit, dass dies passiert, ist höchstens $\frac{\delta}{\epsilon W}$. Die Behauptung folgt nun, indem wir über $h = 1, \dots, n$ summieren. \square

Wir zeigen nun, dass es sogar besser geht. Man kann annehmen, dass $w(\{1, \dots, n\}) > W$. Man löse zunächst das gebrochene Knapsack-Problem: Sei $x^* \in [0, 1]^n$ eine optimale gebrochene Lösung. Nach Proposition 17.1 gibt es ein Element j und eine Zahl $r = \frac{c_j}{w_j}$, so dass $x_i^* = 1$ für alle i mit $\frac{c_i}{w_i} > r$ und $x_i^* = 0$ für alle i mit $\frac{c_i}{w_i} < r$. Für jedes $\gamma \geq 0$ können wir nun das Kernproblem $P(\gamma)$ betrachten, nämlich die Instanz bestehend aus den Elementen i mit $|c_i - rw_i| \leq \gamma$ und der Kapazität $W - w(\{i : c_i > rw_i + \gamma\})$. Sei S eine optimale Lösung dieses Kernproblems und $S^\gamma := S \cup \{i : c_i > rw_i + \gamma\}$. Für genügend großes γ reicht es (Vorschlag von Balas und Zemel [1980]), das Kernproblem zu lösen:

Lemma 17.19. *Gilt $c^\top x^* - c(S^\gamma) \leq \gamma$, so ist S^γ eine optimale Lösung der ursprünglichen Instanz.*

Beweis: Sei S^* eine optimale Lösung. Angenommen, dass $c(S^*) > c(S^\gamma)$. Dann existiert ein Index $j \notin S^*$ mit $c_j > rw_j + \gamma$ oder ein Index $j \in S^*$ mit $c_j < rw_j - \gamma$. Es seien $x_i := 1$ für $i \in S^*$ und $x_i := 0$ für $i \notin S^*$. Im ersten Fall wird die Erhöhung von x_j bis auf 1 und die Verringerung irgendeines x_i bis auf mindestens x_i^* (so dass $w^\top x$ konstant bleibt) den Profit um mehr als $(rw_j + \gamma) - rw_j$ erhöhen. Im letzten Fall wird die Verringerung von x_j bis auf 0 und die Erhöhung irgendeines x_i bis auf höchstens x_i^* (so dass $w^\top x$ konstant bleibt) den Profit um mehr als $rw_j - (rw_j - \gamma)$ erhöhen. In beiden Fällen bekommen wir eine gebrochene Lösung x' mit $c^\top x^* \geq c^\top x' \geq c^\top x + \gamma = c(S^*) + \gamma > c(S^\gamma) + \gamma$. \square

Nun sortiere man die Elemente so, dass $|c_1 - rw_1| \leq \dots \leq |c_n - rw_n|$ gilt und wende den NEMHAUSER-ULLMANN-ALGORITHMUS mit dieser Sortierung der Elemente an. Nach jeder Iteration $i = 1, \dots, n$, setze man $P_i := \{j \in \{i+1, \dots, n\} : c_j > rw_j\}$, und es sei S_i eine Lösung, die $\max\{c(S) : S \subseteq \{1, \dots, i\}, w(S) \leq W - w(P_i)\}$ annimmt.

Nach 17.19 kann man aufhören, sobald $c^\top x^* - c(S_i \cup P_i) < |c_{i+1} - rw_{i+1}|$ gilt. Oft passiert das recht bald (siehe Beier und Vöcking [2006]). Somit kann das KNAPSACK-PROBLEM in der Praxis oft optimal gelöst werden, obwohl es NP-schwer ist.

Aufgaben

1. Man betrachte das Gebrochene Multi-Knapsack-Problem: Eine Instanz besteht aus zwei nichtnegativen ganzen Zahlen m und n und Zahlen w_j , c_{ij} und W_i ($1 \leq i \leq m$, $1 \leq j \leq n$). Man möchte Zahlen $x_{ij} \in [0, 1]$ finden, mit $\sum_{i=1}^m x_{ij} = 1$ für alle j und $\sum_{j=1}^n x_{ij} w_j \leq W_i$ für alle i , so dass $\sum_{i=1}^m \sum_{j=1}^n x_{ij} c_{ij}$ minimal ist.

Kann man einen polynomiellen kombinatorischen Algorithmus für dieses Problem finden, ohne LINEARE OPTIMIERUNG zu gebrauchen?

Hinweis: Reduktion auf ein MINIMUM-COST-FLOW-PROBLEM.

2. Man betrachte den folgenden Greedy-Algorithmus für das KNAPSACK-PROBLEM (ähnlich demjenigen in Proposition 17.6): Man sortiere die Indizes, so dass $\frac{c_1}{w_1} \geq \dots \geq \frac{c_n}{w_n}$, und setze $S := \emptyset$. **For** $i := 1$ **to** n **do:** **If** $\sum_{j \in S \cup \{i\}} w_j \leq W$ **then** setze $S := S \cup \{i\}$. Man zeige, dass dies für kein k einen k -Approximationsalgorithmus ergibt.
3. Man zeige, dass die Approximationszahl des in Proposition 17.6 angegebenen Algorithmus nicht besser als 2 ist.
4. Nach Proposition 17.6 und Aufgabe 3 folgt, dass die Ganzzahligkeitslücke des LP

$$\min \{c^\top x : w^\top x \leq W, 0 \leq x \leq 1\}$$

gleich 2 ist. Man zeige dagegen, dass die Ganzzahligkeitslücke der LP-Relaxierung

$$\min \{c^\top x : w^\top x \geq W, 0 \leq x \leq 1\} \quad (17.5)$$

unbeschränkt ist.

- * 5. Man kann dem obigen LP (17.5) sogenannte Knapsack-Cover-Ungleichungen hinzufügen: Jede zulässige ganzzahlige Lösung hat die Eigenschaft:
Für jedes $I \subseteq \{1, \dots, n\}$ gilt

$$\sum_{j \notin I} \min \{w_j, W - w(I)\} x_j \geq W - w(I),$$

wobei $w(I) = \sum_{i \in I} w_i$. Man zeige, dass die Ganzzahligkeitslücke des resultierenden LP gleich 2 ist.

Hinweis: Gegeben sei eine optimale Lösung x des LP und setze $I := \{i : x_i \geq \frac{1}{2}\}$. Man ordne $\{1, \dots, n\} \setminus I = \{j_1, \dots, j_k\}$ so, dass $w_{j_1} \geq \dots \geq w_{j_k}$. Sei $0 \leq \alpha < 1$ und setze $S_\alpha := \{j_h : \lceil \alpha + \sum_{l=1}^h 2x_{j_l} \rceil > \lceil \alpha + \sum_{l=1}^{h-1} 2x_{j_l} \rceil\}$ und ferner $x_i := 1$ für $i \in I \cup S_\alpha$ und $x_i := 0$ sonst. Man zeige, dass dies für jedes α eine zulässige Lösung ist, unter Benutzung der Tatsache, dass sich die Zahlen $f(\alpha) := \sum_{j \in S_\alpha} \min \{w_j, W - w(I)\}$ (für $\alpha \in [0, 1]$) um höchstens $W - w(I)$ unterscheiden und $\int_0^1 f(\alpha) d\alpha = \sum_{j \notin I} \min \{w_j, W - w(I)\} 2x_j$ gilt.

(Carr et al. [2000])

6. Man finde einen exakten $O(2^{n/2})$ -Algorithmus für das KNAPSACK-PROBLEM.

7. Man erstelle einen exakten $O(nW)$ -Algorithmus für das KNAPSACK-PROBLEM.
8. Man betrachte das folgende Problem: Für gegebene nichtnegative ganze Zahlen $n, c_1, \dots, c_n, w_1, \dots, w_n$ und W finde man eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} w_j \geq W$, die $\sum_{j \in S} c_j$ minimiert. Wie kann man dieses Problem mittels eines pseudopolynomiellen Algorithmus lösen?
- * 9. Kann man das ganzzahlige Multi-Knapsack-Problem (siehe Aufgabe 1) mit festem m in pseudopolynomieller Zeit lösen?
10. Sei $c \in \{0, \dots, k\}^m$ und $s \in [0, 1]^m$. Wie kann man in $O(mk)$ -Zeit entscheiden, ob $\max \{cx : x \in \mathbb{Z}_+^m, sx \leq 1\} \leq k$?
11. Man betrachte die zwei Lagrange-Relaxierungen in Aufgabe 21, Kapitel 5. Man zeige, dass eine der beiden in linearer Zeit gelöst werden kann, während die andere auf m Instanzen des KNAPSACK-PROBLEMS zurückgeführt werden kann.
12. Man betrachte die folgende Variante des PARTITION-Problems:
Für gegebene $a_1, \dots, a_n \in \mathbb{Z}_+$ finde man eine Menge $S \subseteq \{1, \dots, n\}$, für die $\max \left\{ \sum_{i \in S} a_i, \sum_{i \in \{1, \dots, n\} \setminus S} a_i \right\}$ minimal ist.
Man zeige, dass dieses Problem ein voll-polynomielles Approximationsschema besitzt.
13. Man gebe einen polynomiellen Algorithmus für das auf Matroide beschränkte ϵ -DOMINANZ-PROBLEM an.
- * 14. Man beweise, dass die Bedingung in Satz 17.11 hinreichend ist.
15. Man finde einen pseudopolynomiellen Algorithmus für das m -DIMENSIONALE KNAPSACK-PROBLEM für ein beliebiges festes $m \in \mathbb{N}$.
Bemerkung: Dies ist eine Verallgemeinerung von Aufgabe 7.
16. Man zeige, dass es Instanzen für das KNAPSACK-PROBLEM mit n Elementen gibt, für die alle 2^n Teilmengen Pareto-optimal sind.

Literatur

Allgemeine Literatur:

- Garey, M.R., und Johnson, D.S. [1979]: Computers and Intractability; A Guide to the Theory of NP-Completeness. Freeman, San Francisco 1979, Kapitel 4
- Kellerer, H., Pferschy, U., und Pisinger, D. [2004]: Knapsack Problems. Springer, Berlin 2004
- Martello, S., und Toth, P. [1990]: Knapsack Problems; Algorithms and Computer Implementations. Wiley, Chichester 1990
- Papadimitriou, C.H., und Steiglitz, K. [1982]: Combinatorial Optimization; Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, Abschnitte 16.2, 17.3 und 17.4

Zitierte Literatur:

- Balas, E., und Zemel, E. [1980]: An algorithm for large zero-one knapsack problems. Operations Research 28 (1980), 1130–1154

- Beier, R., Röglin, H., und Vöcking, B. [2007]: The smoothed number of Pareto optimal solutions in bicriteria integer optimization. In: Integer Programming and Combinatorial Optimization; Proceedings of the 12th International IPCO Conference; LNCS 4513 (M. Fischetti, D.P. Williamson, Hrsg.), Springer, Berlin 2007, pp. 53–67
- Beier, R., und Vöcking, B. [2006]: An experimental study of random knapsack problems. *Algorithmica* 45 (2006), 121–136
- Bellman, R. [1956]: Notes on the theory of dynamic programming IV – maximization over discrete sets. *Naval Research Logistics Quarterly* 3 (1956), 67–70
- Bellman, R. [1957]: Comment on Dantzig's paper on discrete variable extremum problems. *Operations Research* 5 (1957), 723–724
- Blum, M., Floyd, R.W., Pratt, V., Rivest, R.L., und Tarjan, R.E. [1973]: Time bounds for selection. *Journal of Computer and System Sciences* 7 (1973), 448–461
- Carr, R.D., Fleischer, L.K., Leung, V.J., und Phillips, C.A. [2000]: Strengthening integrality gaps for capacitated network design and covering problems. *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms* (2000), 106–115
- Chan, T.M. [2018]: Approximation schemes for 0-1 knapsack. *Proceedings of the 1st Symposium on Simplicity in Algorithms* (2018), Artikel 5
- Dantzig, G.B. [1957]: Discrete variable extremum problems. *Operations Research* 5 (1957), 266–277
- Frieze, A.M. und Clarke, M.R.B. [1984]: Approximation algorithms for the m -dimensional 0-1 knapsack problem: worst case and probabilistic analyses. *European Journal of Operations Research* 15 (1984), 100–109
- Garey, M.R., und Johnson, D.S. [1978]: Strong NP-completeness results: motivation, examples, and implications. *Journal of the ACM* 25 (1978), 499–508
- Gens, G.V., und Levner, E.V. [1979]: Computational complexity of approximation algorithms for combinatorial problems. In: Mathematical Foundations of Computer Science; LNCS 74 (J. Bečvar, Hrsg.), Springer, Berlin 1979, pp. 292–300
- Ibarra, O.H., und Kim, C.E. [1975]: Fast approximation algorithms for the knapsack and sum of subset problem. *Journal of the ACM* 22 (1975), 463–468
- Karp, R.M. [1972]: Reducibility among combinatorial problems. In: Complexity of Computer Computations (R.E. Miller, J.W. Thatcher, Hrsg.), Plenum Press, New York 1972, pp. 85–103
- Kellerer, H., und Pferschy, U. [2004]: Improved dynamic programming in connection with an FPTAS for the knapsack problem. *Journal on Combinatorial Optimization* 8 (2004), 5–11
- Kellerer, H., Pferschy, U., und Pisinger, D. [2004]: Knapsack Problems. Springer, Berlin 2004
- Korte, B., und Schrader, R. [1981]: On the existence of fast approximation schemes. In: Nonlinear Programming; Vol. 4 (O. Mangasarian, R.R. Meyer, S.M. Robinson, Hrsg.), Academic Press, New York 1981, pp. 415–437
- Lawler, E.L. [1979]: Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research* 4 (1979), 339–356
- Nemhauser, G.L., und Ullmann, Z. [1969]: Discrete dynamic programming and capital allocation. *Management Science* 15 (1969), 494–505
- Pisinger, D. [1999]: Linear time algorithms for knapsack problems with bounded weights. *Journal of Algorithms* 33 (1999), 1–14
- Sahni, S. [1975]: Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM* 22 (1975), 115–124
- Vygen, J. [1997]: The two-dimensional weighted median problem. *Zeitschrift für Angewandte Mathematik und Mechanik* 77 (1997), Supplement, S433–S436

Woeginger, G.J. [2000]: When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? INFORMS Journal on Computing 12 (2000), 57–74



18 Bin-Packung

Angenommen, wir haben n Objekte verschiedener fester Größen und einige Behälter von gleicher Größe. Unser Problem ist es, die Objekte den Behältern zuzuordnen, mit dem Ziel, die Anzahl der benutzten Behälter zu minimieren. Natürlich darf die Gesamtgröße der einem Behälter zugeordneten Objekte die Größe des Behälters nicht übersteigen.

O. B. d. A. können wir die Größe der (gleichgroßen) Behälter auf Eins setzen. Das Problem kann nun folgendermaßen formuliert werden:

BIN-PACKING-PROBLEM

Instanz: Eine Liste nichtnegativer Zahlen $a_1, \dots, a_n \leq 1$.

Aufgabe: Bestimme ein $k \in \mathbb{N}$ und eine Zuordnung

$$f : \{1, \dots, n\} \rightarrow \{1, \dots, k\} \text{ mit } \sum_{i:f(i)=j} a_i \leq 1 \text{ für alle } j \in \{1, \dots, k\}, \text{ die } k \text{ minimiert.}$$

Es gibt nicht sehr viele kombinatorische Optimierungsprobleme, deren praktische Relevanz derart klar auf der Hand liegt. Die einfachste Fassung des Cutting-Stock-Problems, zum Beispiel, ist ein äquivalentes Problem: Wir haben eine (als unbegrenzt zu betrachtende) Anzahl von gleichlangen Balken (o. Ä.) und n Zahlen a_1, \dots, a_n . Ziel ist es, mit möglichst wenig Verschnitt Balken der Längen a_1, \dots, a_n zu erhalten.

Obwohl eine Instanz I des Bin-Packung-Problems aus einer Zahlenliste besteht, in der Zahlen durchaus mehrfach vorkommen dürfen, bezeichnen wir mit $x \in I$ ein Element der Liste I , welches gleich x ist. Es bezeichne $|I|$ die Anzahl der Elemente in der Liste I . Wir werden auch die Abkürzung $\text{SUM}(a_1, \dots, a_n) := \sum_{i=1}^n a_i$ benutzen. Offensichtlich ist dies eine untere Schranke: Es gilt $\lceil \text{SUM}(I) \rceil \leq \text{OPT}(I)$ für jede Instanz I .

In Abschnitt 18.1 werden wir beweisen, dass das BIN-PACKING-PROBLEM stark NP -schwer ist. Ferner werden wir einige einfache Approximationsalgorithmen besprechen. Es wird sich herausstellen, dass kein Algorithmus eine bessere Approximationsgüte als $\frac{3}{2}$ haben kann, sofern $P \neq NP$. Asymptotisch kann man jedoch eine beliebig gute Approximationsgüte erreichen: In den Abschnitten 18.2 und 18.3 werden wir ein voll-polynomielles asymptotisches Approximationsschema beschreiben. Dieses basiert auf der ELLIPSOIDMETHODE und Resultaten aus Kapitel 17.

18.1 Greedy-Heuristiken

In diesem Abschnitt werden wir einige Greedy-Heuristiken für das BIN-PACKING-PROBLEM analysieren. Da Letzteres *NP*-schwer ist, besteht keine Hoffnung, einen exakten polynomiellen Algorithmus zu finden:

Satz 18.1. *Das folgende Problem ist NP-vollständig: Entscheide für eine gegebene Instanz I des BIN-PACKING-PROBLEMS, ob es eine Lösung für I mit zwei Behältern gibt.*

Beweis: Das angegebene Entscheidungsproblem ist trivialerweise in *NP*. Nun transformieren wir das nach Korollar 15.28 *NP*-vollständige Problem PARTITION in das obige Entscheidungsproblem. Sei eine Instanz c_1, \dots, c_n von PARTITION gegeben und betrachte die Instanz a_1, \dots, a_n des BIN-PACKING-PROBLEMS, wobei

$$a_i = \frac{2c_i}{\sum_{j=1}^n c_j}.$$

Offensichtlich genügen zwei Behälter genau dann, wenn es eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$ gibt. \square

Korollar 18.2. *Gilt $P \neq NP$, so gibt es keinen ρ -Approximationsalgorithmus für das BIN-PACKING-PROBLEM mit $\rho < \frac{3}{2}$.* \square

Für jedes feste k gibt es einen pseudopolynomiellen Algorithmus, der für eine gegebene Instanz I entscheidet, ob k Behälter genügen (Aufgabe 1). Im Allgemeinen ist dieses Problem aber stark *NP*-vollständig:

Satz 18.3. (Garey und Johnson [1975]) *Das folgende Problem ist stark NP-vollständig: Gegeben sei eine Instanz I des BIN-PACKING-PROBLEMS und eine Zahl B . Entscheide, ob es für I eine Lösung mit B Behältern gibt.*

Beweis: Der Beweis erfolgt mittels Transformation von 3-DIMENSIONALES MATCHING (3DM) (Satz 15.26).

Für eine gegebene Instanz U, V, W, T von 3DM konstruieren wir eine Bin-Packing-Instanz I mit $4|T|$ Objekten: Die Objektmenge sei

$$S := \bigcup_{t=(u,v,w) \in T} \{t, (u, t), (v, t), (w, t)\}.$$

Seien $U = \{u_1, \dots, u_n\}$, $V = \{v_1, \dots, v_n\}$ und $W = \{w_1, \dots, w_n\}$. Für jedes $x \in U \dot{\cup} V \dot{\cup} W$ wählen wir ein $t_x \in T$ mit $(x, t_x) \in S$. Für jedes $t = (u_i, v_j, w_k) \in T$ definieren wir die Objektgrößen wie folgt:

t	hat die Größe	$\frac{1}{C}(10N^4 + 8 - iN - jN^2 - kN^3)$
(u_i, t)	hat die Größe	$\begin{cases} \frac{1}{C}(10N^4 + iN + 1) & \text{für } t = t_{u_i} \\ \frac{1}{C}(11N^4 + iN + 1) & \text{für } t \neq t_{u_i} \end{cases}$
(v_j, t)	hat die Größe	$\begin{cases} \frac{1}{C}(10N^4 + jN^2 + 2) & \text{für } t = t_{v_j} \\ \frac{1}{C}(11N^4 + jN^2 + 2) & \text{für } t \neq t_{v_j} \end{cases}$
(w_k, t)	hat die Größe	$\begin{cases} \frac{1}{C}(10N^4 + kN^3 + 4) & \text{für } t = t_{w_k} \\ \frac{1}{C}(8N^4 + kN^3 + 4) & \text{für } t \neq t_{w_k} \end{cases}$

wobei $N := 100n$ und $C := 40N^4 + 15$. Dies ergibt eine Instanz $I = (a_1, \dots, a_{4|T|})$ des BIN-PACKING-PROBLEMS. Nun setzen wir $B := |T|$ und behaupten, dass I genau dann eine Lösung mit höchstens B Behältern hat, wenn die ursprüngliche 3DM-Instanz eine Ja-Instanz ist, d. h. es gibt eine Teilmenge M von T mit $|M| = n$, so dass für zwei verschiedene $(u, v, w), (u', v', w') \in M$ gilt: $u \neq u'$, $v \neq v'$ und $w \neq w'$.

Zunächst nehmen wir an, dass es eine solche Lösung M der 3DM-Instanz gibt. Da die Lösbarkeit von I mit B Behältern unabhängig von der Wahl der t_x ($x \in U \cup V \cup W$) ist, können wir Letztere so umdefinieren, dass $t_x \in M$ für alle x . Nun packen wir für jedes $t = (u, v, w) \in T$ die Objekte $t, (u, t), (v, t), (w, t)$ in einen Behälter. Damit erhalten wir eine Lösung mit $|T|$ Behältern.

Sei nun umgekehrt f eine Lösung für I mit $B = |T|$ Behältern. Da $\text{SUM}(I) = |T|$, ist jeder Behälter restlos voll. Da sämtliche Objektgrößen streng zwischen $\frac{1}{5}$ und $\frac{1}{3}$ liegen, enthält jeder Behälter genau vier Objekte.

Betrachte den k -ten Behälter ($k \in \{1, \dots, B\}$). Da $C \sum_{i:f(i)=k} a_i = C \equiv 15 \pmod{N}$, enthält er ein $t = (u, v, w) \in T$, ein $(u', t') \in U \times T$, ein $(v', t'') \in V \times T$ und ein $(w', t''') \in W \times T$. Da $C \sum_{i:f(i)=k} a_i = C \equiv 15 \pmod{N^2}$, folgt $u = u'$. Betrachten wir die Summen modulo N^3 und modulo N^4 , so folgt analog, dass $v = v'$ und $w = w'$. Ferner gilt entweder $t' = t_u$ und $t'' = t_v$ und $t''' = t_w$ (Fall 1), oder $t' \neq t_u$ und $t'' \neq t_v$ und $t''' \neq t_w$ (Fall 2).

Sei nun M die Menge derjenigen $t \in T$, die einem Fall-1-Behälter zugeordnet sind. Offensichtlich ist M eine Lösung für die 3DM-Instanz.

Beachte, dass alle in der konstruierten Bin-Pack-Instanz I vorkommenden Zahlen polynomiell groß, genauer: $O(n^4)$ sind. Da 3DM NP-vollständig ist (Satz 15.26), ist der Satz bewiesen. \square

Dieser Beweis stammt von Papadimitriou [1994]. Sogar unter der Annahme, dass $P \neq NP$, schließt der obige Satz die Möglichkeit eines absoluten Approximationsalgorithmus nicht aus, z. B. einen, der höchstens einen Behälter mehr als die optimale Anzahl benötigt. Es ist eine offene Frage, ob es einen solchen Algorithmus gibt.

Die einfachste Heuristik für das Bin-Pack-Problem ist wohl die folgende:

NEXT-FIT-ALGORITHMUS (NF)

Input: Eine Instanz a_1, \dots, a_n des BIN-PACKING-PROBLEMS.

Output: Eine Lösung (k, f) .

- ① Setze $k := 1$ und $S := 0$.
 - ② **For** $i := 1$ **to** n **do:**
 - If** $S + a_i > 1$ **then** setze $k := k + 1$ und $S := 0$.
 - Setze $f(i) := k$ und $S := S + a_i$.
-

Es bezeichne $\text{NF}(I)$ die von diesem Algorithmus für die Instanz I benötigte Anzahl k der Behälter.

Satz 18.4. Der NEXT-FIT-ALGORITHMUS läuft in $O(n)$ -Zeit. Für jede Instanz $I = a_1, \dots, a_n$ gilt

$$\text{NF}(I) \leq 2\lceil \text{SUM}(I) \rceil - 1 \leq 2\text{OPT}(I) - 1.$$

Beweis: Die Laufzeit ist klar. Setze $k := \text{NF}(I)$ und sei f die von dem NEXT-FIT-ALGORITHMUS gefundene Zuordnung. Für $j = 1, \dots, \lfloor \frac{k}{2} \rfloor$ folgt

$$\sum_{i:f(i) \in \{2j-1, 2j\}} a_i > 1.$$

Addieren wir all diese Ungleichungen, so bekommen wir

$$\left\lfloor \frac{k}{2} \right\rfloor < \text{SUM}(I).$$

Da die linke Seite eine ganze Zahl ist, folgern wir daraus, dass

$$\frac{k-1}{2} \leq \left\lfloor \frac{k}{2} \right\rfloor \leq \lceil \text{SUM}(I) \rceil - 1.$$

Damit ist $k \leq 2\lceil \text{SUM}(I) \rceil - 1$ bewiesen. Die zweite Ungleichung ist trivial. \square

Die Instanzen $2\epsilon, 1 - \epsilon, 2\epsilon, 1 - \epsilon, \dots, 2\epsilon$ mit sehr kleinem $\epsilon > 0$ zeigen, dass dies die bestmögliche Schranke ist. Somit ist der NEXT-FIT-ALGORITHMUS ein 2-Approximationsalgorithmus. Natürlich wird die Approximationsgüte besser, wenn die vorkommenden Zahlen kleiner sind:

Proposition 18.5. Sei $0 < \gamma < 1$. Für jede Instanz $I = a_1, \dots, a_n$ mit $a_i \leq \gamma$ für alle $i \in \{1, \dots, n\}$ gilt

$$\text{NF}(I) \leq \left\lceil \frac{\text{SUM}(I)}{1 - \gamma} \right\rceil.$$

Beweis: Wir haben $\sum_{i:f(i)=j} a_i > 1 - \gamma$ für $j = 1, \dots, \text{NF}(I) - 1$. Addieren wir all diese Ungleichungen, so erhalten wir $(\text{NF}(I) - 1)(1 - \gamma) < \text{SUM}(I)$ und somit

$$\text{NF}(I) - 1 \leq \left\lceil \frac{\text{SUM}(I)}{1 - \gamma} \right\rceil - 1.$$

□

Ein zweiter Ansatz zur Konstruktion eines effizienten Approximationsalgorithmus könnte folgendermaßen aussehen:

FIRST-FIT-ALGORITHMUS (FF)

Input: Eine Instanz a_1, \dots, a_n des BIN-PACKING-PROBLEMS.

Output: Eine Lösung (k, f) .

① **For** $i := 1$ **to** n **do:**

$$\text{Setze } f(i) := \min \left\{ j \in \mathbb{N} : \sum_{h < i: f(h)=j} a_h + a_i \leq 1 \right\}.$$

② Setze $k := \max_{i \in \{1, \dots, n\}} f(i)$.

Natürlich kann der FIRST-FIT-ALGORITHMUS nicht schlechter als der NEXT-FIT-ALGORITHMUS sein. Also ist FIRST-FIT ein weiterer 2-Approximationsalgorithmus. Er ist sogar besser:

Satz 18.6. (Dósa and Sgall [2013]) *Für alle Instanzen I des BIN-PACKING-PROBLEMS gilt*

$$\text{FF}(I) \leq \left\lfloor \frac{17}{10} \text{OPT}(I) \right\rfloor.$$

Ferner gibt es Instanzen I mit beliebig großem $\text{OPT}(I)$, bei denen die Ungleichung mit Gleichheit erfüllt ist.

Dies ist eine leichte Verschärfung früherer Ergebnisse von Johnson et al. [1974] und Garey et al. [1976]. Wir werden den sehr komplizierten Beweis hier nicht führen.

Nach Proposition 18.5 verhält sich der NEXT-FIT (also auch der FIRST-FIT) ALGORITHMUS gut bei kleinen Objekten. Somit liegt es nahe, die größeren Objekte zuerst zu verpacken. Die folgende Modifizierung des FIRST-FIT-ALGORITHMUS scannt die n Zahlen in abnehmender Reihenfolge:

FIRST-FIT-DECREASING-ALGORITHMUS (FFD)

Input: Eine Instanz a_1, \dots, a_n des BIN-PACKING-PROBLEMS.

Output: Eine Lösung (k, f) .

① Sortiere die Zahlen so, dass $a_1 \geq a_2 \geq \dots \geq a_n$.

② Wende den FIRST-FIT-ALGORITHMUS an.

Satz 18.7. (Simchi-Levi [1994]) *Der FIRST-FIT-DECREASING-ALGORITHMUS ist ein $\frac{3}{2}$ -Approximationsalgorithmus für das BIN-PACKING-PROBLEM.*

Beweis: Sei I eine Instanz und $k := \text{FFD}(I)$. Setze $j := \lceil \frac{2}{3}k \rceil$ und betrachte den j -ten Behälter. Enthält er ein Objekt der Größe $> \frac{1}{2}$, so hatte jeder Behälter mit kleinerem Index nicht genügend Platz für dieses Objekt und enthielt somit bereits zugeteilte Objekte. Da die Objekte in abnehmender Gewichtsreihenfolge betrachtet werden, gibt es somit mindestens j Objekte der Größe $> \frac{1}{2}$. Damit folgt $\text{OPT}(I) \geq j \geq \frac{2}{3}k$.

Andernfalls enthält der j -te Behälter, also auch jeder Behälter mit größerem Index, kein Objekt der Größe $> \frac{1}{2}$. Also enthalten die Behälter $j, j+1, \dots, k$ mindestens $2(k-j)+1$ Objekte, von denen keins in die Behälter mit den Indizes $1, \dots, j-1$ passt. Beachte, dass $2(k-j)+1 \geq 2(k-(\frac{2}{3}k + \frac{2}{3}))+1 = \frac{2}{3}k - \frac{1}{3} \geq j-1$. Somit ist $\text{OPT}(I) \geq \text{SUM}(I) > j-1$, d.h. $\text{OPT}(I) \geq j \geq \frac{2}{3}k$. \square

Nach Korollar 18.2 ist dies das bestmögliche Ergebnis (betrachte für FFD die Instanz 0,4, 0,4, 0,3, 0,3, 0,3, 0,3). Die asymptotische Approximationsgüte ist jedoch besser: Johnson [1973] hat bewiesen, dass $\text{FFD}(I) \leq \frac{11}{9} \text{OPT}(I) + 4$ für alle Instanzen I (siehe auch Johnson [1974]). Mit einem einfacheren Beweis hat Baker [1985] bewiesen, dass $\text{FFD}(I) \leq \frac{11}{9} \text{OPT}(I) + 3$. Yue [1991] hat dies auf $\text{FFD}(I) \leq \frac{11}{9} \text{OPT}(I) + 1$ verbessert. Diese Frage wurde schließlich entschieden, indem Dósa et al. [2013] Folgendes bewies:

Satz 18.8. (Dósa et al. [2013]) *Für alle Instanzen I des BIN-PACKING-PROBLEMS gilt*

$$\text{FFD}(I) \leq \frac{11}{9} \text{OPT}(I) + \frac{2}{3},$$

und diese Schranke ist bestmöglich.

Der Beweis ist zu kompliziert, um ihn hier zu führen. Wir werden jedoch eine Klasse von Instanzen I mit beliebig großem $\text{OPT}(I)$ und $\text{FFD}(I) = \frac{11}{9} \text{OPT}(I)$ angeben. (Dieses Beispiel ist der Arbeit von Garey und Johnson [1979] entnommen.)

Dazu nehmen wir ein genügend kleines $\epsilon > 0$ und setzen $I := \{a_1, \dots, a_{30m}\}$ mit

$$a_i = \begin{cases} \frac{1}{2} + \epsilon & \text{für } 1 \leq i \leq 6m, \\ \frac{1}{4} + 2\epsilon & \text{für } 6m < i \leq 12m, \\ \frac{1}{4} + \epsilon & \text{für } 12m < i \leq 18m, \\ \frac{1}{4} - 2\epsilon & \text{für } 18m < i \leq 30m. \end{cases}$$

Die optimale Lösung lautet:

$$\begin{array}{ll} 6m \text{ Behälter mit den Objekten} & \frac{1}{2} + \epsilon, \frac{1}{4} + \epsilon, \frac{1}{4} - 2\epsilon, \\ & \frac{1}{4} + 2\epsilon, \frac{1}{4} + 2\epsilon, \frac{1}{4} - 2\epsilon, \frac{1}{4} - 2\epsilon. \\ 3m \text{ Behälter mit den Objekten} & \end{array}$$

Die FFD-Lösung lautet:

$$\begin{array}{ll} 6m \text{ Behälter mit den Objekten} & \frac{1}{2} + \epsilon, \frac{1}{4} + 2\epsilon, \\ & \frac{1}{4} + \epsilon, \frac{1}{4} + \epsilon, \frac{1}{4} + \epsilon, \\ 2m \text{ Behälter mit den Objekten} & \\ 3m \text{ Behälter mit den Objekten} & \frac{1}{4} - 2\epsilon, \frac{1}{4} - 2\epsilon, \frac{1}{4} - 2\epsilon, \frac{1}{4} - 2\epsilon. \end{array}$$

Somit gilt $\text{OPT}(I) = 9m$ und $\text{FFD}(I) = 11m$.

Es gibt einige weitere Algorithmen für das BIN-PACKING-PROBLEM, manche mit besserer asymptotischer Approximationsgüte als $\frac{11}{9}$. Im nächsten Abschnitt zeigen wir, dass man eine beliebig nahe an 1 liegende asymptotische Approximationsgüte erreichen kann.

In manchen Anwendungen muss man die Objekte in der Reihenfolge ihrer Ankunft verpacken, ohne jegliche Kenntnis über die noch erwarteten Objekte zu haben. Algorithmen, denen keine Informationen über die weiteren Objekte zur Verfügung stehen, heißen Online-Algorithmen. Beispiele hierzu sind NEXT-FIT und FIRST-FIT, aber nicht der FIRST-FIT-DECREASING-ALGORITHMUS. Der bislang beste Online-Algorithmus für das BIN-PACKING-PROBLEM hat die asymptotische Approximationsgüte 1,59 (Seiden [2002]). Andererseits hat van Vliet [1992] bewiesen, dass es keinen asymptotischen Online-1,54-Approximationsalgorithmus für das BIN-PACKING-PROBLEM gibt. Eine schwächere untere Schranke wird in Aufgabe 6 besprochen.

18.2 Ein asymptotisches Approximationsschema

In diesem Abschnitt zeigen wir, dass es für jedes $\epsilon > 0$ einen Algorithmus mit linearer Laufzeit gibt, der immer eine Lösung mit höchstens $(1 + \epsilon) \text{OPT}(I) + \frac{1}{\epsilon^2}$ Behältern findet.

Zunächst betrachten wir Instanzen mit nicht allzu vielen verschiedenen Zahlen. Die verschiedenen Zahlen unserer Instanz I bezeichnen wir mit s_1, \dots, s_m . Es enthalte I genau b_i Kopien von s_i ($i = 1, \dots, m$).

Mit T_1, \dots, T_N bezeichnen wir die verschiedenen Packmöglichkeiten eines Behälters:

$$\{T_1, \dots, T_N\} := \left\{ (k_1, \dots, k_m) \in \mathbb{Z}_+^m : \sum_{i=1}^m k_i s_i \leq 1 \right\}.$$

Wir schreiben $T_j = (t_{j1}, \dots, t_{jm})$. Dann kann unser BIN-PACKING-PROBLEM äquivalent als das folgende ganzzahlige LP formuliert werden (nach Eisemann [1957]):

$$\begin{aligned} \min \quad & \sum_{j=1}^N x_j \\ \text{bzgl.} \quad & \sum_{j=1}^N t_{ji} x_j \geq b_i \quad (i = 1, \dots, m) \\ & x_j \in \mathbb{Z}_+ \quad (j = 1, \dots, N). \end{aligned} \tag{18.1}$$

Eigentlich wollen wir $\sum_{j=1}^N t_{ji} x_j = b_i$ haben, aber die Relaxierung dieser Nebenbedingung ist ohne Wirkung. Die LP-Relaxierung von (18.1) lautet:

$$\begin{aligned} \min \quad & \sum_{j=1}^N x_j \\ \text{bzgl.} \quad & \sum_{j=1}^N t_{ji} x_j \geq b_i \quad (i = 1, \dots, m) \\ & x_j \geq 0 \quad (j = 1, \dots, N). \end{aligned} \tag{18.2}$$

Dieses LP wird manchmal Konfigurations-LP genannt, da es für jede Behälterkonfiguration eine Variable enthält. Der folgende Satz besagt, dass man durch Rundung einer Lösung der LP-Relaxierung (18.2) eine Lösung von (18.1), d. h. vom BIN-PACKING-PROBLEM, erhält, die nicht viel schlechter ausfällt:

Satz 18.9. (Fernandez de la Vega und Lueker [1981]) *Sei I eine Instanz des BIN-PACKING-PROBLEMS mit nur m verschiedenen Zahlen. Sei x eine zulässige (aber nicht notwendigerweise optimale) Lösung von (18.2) mit höchstens m nichtverschwindenden Komponenten. Dann kann eine Lösung des BIN-PACKING-PROBLEMS mit höchstens $\lceil \sum_{j=1}^N x_j \rceil + \left\lfloor \frac{m-1}{2} \right\rfloor$ Behältern in $O(|I|)$ -Zeit gefunden werden.*

Beweis: Betrachte $\lfloor x \rfloor$, das durch Abrunden aller Komponenten von x entsteht. Der Vektor $\lfloor x \rfloor$ verpackt I i. A. nicht vollständig (vielleicht verpackt er manche Zahlen öfter als notwendig, was aber ohne Wirkung bleibt). Die übrig gebliebenen Objekte bilden eine Instanz I' . Beachte, dass

$$\text{SUM}(I') \leq \sum_{j=1}^N (x_j - \lfloor x_j \rfloor) \sum_{i=1}^m t_{ji} s_i \leq \sum_{j=1}^N x_j - \sum_{j=1}^N \lfloor x_j \rfloor.$$

Somit genügt es, I' in höchstens $\lceil \text{SUM}(I') \rceil + \left\lfloor \frac{m-1}{2} \right\rfloor$ Behältern zu verpacken, weil dann die Gesamtanzahl der benutzten Behälter nicht größer ist, als

$$\sum_{j=1}^N \lfloor x_j \rfloor + \lceil \text{SUM}(I') \rceil + \left\lfloor \frac{m-1}{2} \right\rfloor \leq \left\lceil \sum_{j=1}^N x_j \right\rceil + \left\lfloor \frac{m-1}{2} \right\rfloor.$$

Wir betrachten zwei Packverfahren für I' . Zunächst folgt: Der Vektor $\lceil x \rceil - \lfloor x \rfloor$ verpackt auf jeden Fall wenigstens die Elemente von I' . Die Anzahl der benutzten Behälter ist höchstens gleich m , da x höchstens m nichtverschwindende Komponenten hat. Zweitens können wir eine Verpackung von I' unter Benutzung von höchstens $2\lceil \text{SUM}(I') \rceil - 1$ Behältern bekommen, indem wir den NEXT-FIT-ALGORITHMUS (Satz 18.4) anwenden. Beide Verpackungen können in linearer Zeit bewerkstelligt werden.

Die bessere dieser beiden Verpackungen benutzt höchstens $\min\{m, 2\lceil \text{SUM}(I') \rceil - 1\} \leq \lceil \text{SUM}(I') \rceil + \frac{m-1}{2}$ Behälter. Damit ist der Satz bewiesen. \square

Korollar 18.10. (Fernandez de la Vega und Lueker [1981]) *Seien m und $\gamma > 0$ feste Konstanten. Sei I eine Instanz des BIN-PACKING-PROBLEMS mit nur m verschiedenen Zahlen, alle größer oder gleich γ . Dann können wir eine Lösung mit höchstens $\text{OPT}(I) + \left\lfloor \frac{m-1}{2} \right\rfloor$ Behältern in $O(|I|)$ -Zeit finden.*

Beweis: Mit dem SIMPLEXALGORITHMUS (Satz 3.14) können wir eine optimale Basislösung x^* von (18.2) finden, d. h. eine Ecke des Polyeders. Da jede Ecke N Nebenbedingungen mit Gleichheit erfüllt (Proposition 3.9), hat x^* höchstens m nichtverschwindende Komponenten.

Die zur Bestimmung von x^* benötigte Zeit hängt nur von m und N ab. Beachte, dass $N \leq (m+1)^{\frac{1}{\gamma}}$, da in jedem Behälter höchstens $\frac{1}{\gamma}$ Objekte sind. Somit kann x^* in konstanter Zeit gefunden werden.

Da $\left\lceil \sum_{j=1}^N x_j^* \right\rceil \leq \text{OPT}(I)$, folgt der Beweis mit Satz 18.9. \square

Eine Verwendung der ELLIPSOIDMETHODE (Satz 4.18) führt zu demselben Ergebnis. Dies ist jedoch nicht das bestmögliche: Für festes m und γ kann man sogar die exakte Optimallösung in polynomieller Zeit finden, da GANZZAHLIGE OPTIMIERUNG mit einer konstanten Anzahl von Variablen in polynomieller Zeit gelöst werden kann (Lenstra [1983]). Ein einfacherer exakter Algorithmus für feste m wird in Aufgabe 7 besprochen. Diese stärkeren Resultate werde hier jedoch nicht benötigt.

Im nächsten Abschnitt werden wir Satz 18.9 wieder benutzen, um dieselbe Approximationsgüte in polynomieller Zeit zu erreichen, und zwar auch, wenn m und γ nicht fest sind (im Beweis von Satz 18.14).

Wir sind nun in der Lage, den Algorithmus von Fernandez de la Vega und Lueker [1981] zu formulieren. Er läuft in groben Zügen folgendermaßen ab. Zunächst partitionieren wir die n Zahlen in $m+2$ Gruppen, ihrer Größe entsprechend. Die Gruppe mit den größten Zahlen verpacken wir, indem wir einen Behälter pro Zahl benutzen. Dann verpacken wir die m mittleren Gruppen, indem wir in jeder dieser Gruppen alle Zahlen der größten gleich setzen und dann Korollar 18.10 anwenden. Schließlich verpacken wir noch die Gruppe mit den kleinsten Zahlen.

FERNANDEZ-DE-LA-VEGA-LUEKER-ALGORITHMUS

Input: Eine Instanz $I = a_1, \dots, a_n$ des BIN-PACKING-PROBLEMS.

Eine Zahl $\epsilon > 0$.

Output: Eine Lösung (k, f) für I .

- ① Setze $\gamma := \frac{\epsilon}{\epsilon+1}$ und $h := \lceil \epsilon \text{ SUM}(I) \rceil$.
- ② Sei $I_1 = L, M, R$ eine Umordnung der Liste I , wobei
 $M = K_0, y_1, K_1, y_2, \dots, K_{m-1}, y_m$ und $L, K_0, K_1, \dots, K_{m-1}$ und
 R wieder Listen sind und die folgenden Eigenschaften gelten:
 - (a) Für alle $x \in L$: $x < \gamma$.
 - (b) Für alle $x \in K_0$: $\gamma \leq x \leq y_1$.
 - (c) Für alle $x \in K_i$: $y_i \leq x \leq y_{i+1}$ ($i = 1, \dots, m-1$).
 - (d) Für alle $x \in R$: $y_m \leq x$.
 - (e) $|K_1| = \dots = |K_{m-1}| = |R| = h-1$ und $|K_0| \leq h-1$.
 Es wird (k, f) nun durch die folgenden drei Verpackungsschritte bestimmt:
- ③ Bestimme eine Verpackung S_R von R , die $|R|$ Behälter benutzt.
- ④ Betrachte eine Instanz Q bestehend aus den Zahlen y_1, y_2, \dots, y_m , wobei
jede h mal vorkommt. Bestimme eine Verpackung S_Q von Q , die höchstens $\frac{m+1}{2}$ mehr Behälter als notwendig benutzt (unter Verwendung von Korollar 18.10). Verändere S_Q in eine Verpackung S_M von M .
- ⑤ So lange ein Behälter von S_R oder S_M noch mindestens Raum γ hat, fülle man ihn mit Objekten von L . Schließlich bestimme man eine Verpackung der restlichen Objekte von L mit dem NEXT-FIT-ALGORITHMUS.

In ④ haben wir eine etwas schwächere Schranke als die in Korollar 18.10 gewonnene benutzt. Dies macht hier nichts aus, und außerdem benötigen wir die obige Variante in Abschnitt 18.3. Der obige Algorithmus ist ein asymptotisches Approximationsschema. Genauer:

Satz 18.11. (Fernandez de la Vega und Lueker [1981]) *Für jedes $0 < \epsilon \leq \frac{1}{2}$ und jede Instanz I des BIN-PACKING-PROBLEMS liefert der FERNANDEZ-DE-LA-VEGA-LUEKER-ALGORITHMUS eine Lösung mit höchstens $(1 + \epsilon) \text{ OPT}(I) + \frac{1}{\epsilon^2}$ Behältern. Die Laufzeit ist $O(n \frac{1}{\epsilon^2})$ plus die zur Lösung von (18.2) benötigte Zeit. Für festes ϵ ist die Laufzeit $O(n)$.*

Beweis: In ② bestimmen wir zunächst L in $O(n)$ -Zeit. Dann setzen wir $m := \left\lfloor \frac{|I| - |L|}{h} \right\rfloor$. Da $\gamma(|I| - |L|) \leq \text{SUM}(I)$, folgt

$$m \leq \frac{|I| - |L|}{h} \leq \frac{|I| - |L|}{\epsilon \text{ SUM}(I)} \leq \frac{1}{\gamma \epsilon} = \frac{\epsilon + 1}{\epsilon^2}.$$

Wir wissen, dass y_i das $(|I| + 1 - (m - i + 1)h)$ -kleinste Objekt ist ($i = 1, \dots, m$). Somit können wir nach Korollar 17.4 jedes y_i in $O(n)$ -Zeit bestimmen. Schließlich bestimmen wir K_0, K_1, \dots, K_{m-1} , und R , jedes in $O(n)$ Zeit. Demnach benötigt ② $O(mn)$ -Zeit. Beachte, dass $m = O(\frac{1}{\epsilon^2})$.

Die Schritte ③, ④ und ⑤ – mit Ausnahme der Lösung von (18.2) – können leicht so implementiert werden, dass sie in $O(n)$ Zeit ablaufen. Für festes ϵ kann (18.2) auch in $O(n)$ -Zeit optimal gelöst werden (Korollar 18.10).

Nun beweisen wir die Approximationsgüte. Sei k die Anzahl der vom Algorithmus benutzten Behälter. Bezeichnen wir mit $|S_R|$ bzw. $|S_M|$ die Anzahl der für die Verpackung von R bzw. M benutzten Behälter, so haben wir

$$|S_R| \leq |R| = h - 1 < \epsilon \text{SUM}(I) \leq \epsilon \text{OPT}(I).$$

Beachte ferner, dass $\text{OPT}(Q) \leq \text{OPT}(I)$: Die Größe des i -größten Objektes von I ist größer oder gleich der Größe des i -größten Objektes von Q für alle $i = 1, \dots, hm$. Daraus folgt mit ④ (Korollar 18.10):

$$|S_M| = |S_Q| \leq \text{OPT}(Q) + \frac{m+1}{2} \leq \text{OPT}(I) + \frac{m+1}{2}.$$

In ⑤ können wir einige Objekte aus L in Behälter von S_R und S_M packen. Sei L' die Liste der restlichen Objekte von L .

Fall 1: Es ist L' nicht leer. Dann ist die Gesamtgröße der Objekte in jedem Behälter, außer eventuell dem letzten, größer als $1 - \gamma$, demnach ist $(1 - \gamma)(k - 1) < \text{SUM}(I) \leq \text{OPT}(I)$. Daraus folgern wir, dass

$$k \leq \frac{1}{1 - \gamma} \text{OPT}(I) + 1 = (1 + \epsilon) \text{OPT}(I) + 1.$$

Fall 2: L' ist leer. Dann folgt, da $\epsilon \leq \frac{1}{2}$:

$$\begin{aligned} k &\leq |S_R| + |S_M| \\ &< \epsilon \text{OPT}(I) + \text{OPT}(I) + \frac{m+1}{2} \\ &\leq (1 + \epsilon) \text{OPT}(I) + \frac{\epsilon + 1 + \epsilon^2}{2\epsilon^2} \\ &\leq (1 + \epsilon) \text{OPT}(I) + \frac{1}{\epsilon^2}. \end{aligned}$$

□

Natürlich wächst die Laufzeit exponentiell mit $\frac{1}{\epsilon}$. Karmarkar und Karp haben jedoch gezeigt, wie man ein voll-polynomielles asymptotisches Approximationsschema bekommen kann. Dies werden wir im nächsten Abschnitt betrachten.

18.3 Der Karmarkar-Karp-Algorithmus

Der Algorithmus von Karmarkar und Karp [1982] funktioniert wie der Algorithmus im vorigen Abschnitt, nur dass er die LP-Relaxierung (18.2) mit einem konstanten absoluten Fehler löst, anstatt optimal wie in Korollar 18.10.

Die Tatsache, dass die Anzahl der Variablen exponentiell mit $\frac{1}{\epsilon}$ wächst, hält uns nicht unbedingt davon ab, das LP zu lösen: Gilmore und Gomory [1961] haben die Spaltenerzeugungsmethode entwickelt und eine Variante des SIMPLEXALGORITHMUS konstruiert, die (18.2) in der Praxis recht effizient löst. Ähnliche Ideen führen zu einem theoretisch effizienten Algorithmus, wenn man stattdessen den GRÖTSCHEL-LOVÁSZ-SCHRIJVER-ALGORITHMUS benutzt.

In beiden obigen Ansätzen spielt das duale LP eine herausragende Rolle. Das duale LP von (18.2) ist:

$$\begin{aligned} \max \quad & yb \\ \text{bzgl.} \quad & \sum_{i=1}^m t_{ji} y_i \leq 1 \quad (j = 1, \dots, N) \\ & y_i \geq 0 \quad (i = 1, \dots, m). \end{aligned} \tag{18.3}$$

Dieses LP hat nur m Variablen, aber eine exponentielle Anzahl von Nebenbedingungen. Die Anzahl der Nebenbedingungen spielt jedoch keine Rolle, so lange wir das SEPARATIONS-PROBLEM in polynomieller Zeit lösen können. Es wird sich herausstellen, dass das SEPARATIONS-PROBLEM mit einem KNAPSACK-PROBLEM äquivalent ist. Da wir KNAPSACK-PROBLEME mit beliebig kleinem Fehler lösen können, können wir auch das SCHWACHE SEPARATIONS-PROBLEM in polynomieller Zeit lösen. Diese Idee führt zum Beweis des folgenden Resultats:

Lemma 18.12. (Karmarkar und Karp [1982]) *Sei I eine Instanz des BIN-PACKING-PROBLEMS mit nur m verschiedenen Zahlen, alle größer oder gleich γ . Sei $\delta > 0$. Dann kann man eine zulässige Lösung y^* des dualen LP (18.3) mit vom Optimum um höchstens δ abweichendem Zielfunktionswert in $O\left(m^6 \log^2 \frac{mn}{\gamma\delta} + \frac{m^5 n}{\delta} \log \frac{mn}{\gamma\delta}\right)$ -Zeit finden.*

Beweis: Wir können annehmen, dass $\delta = \frac{1}{p}$ für eine natürliche Zahl p . Wir wenden den GRÖTSCHEL-LOVÁSZ-SCHRIJVER-ALGORITHMUS (Satz 4.19) an. Sei \mathcal{D} das Polyeder von (18.3). Dann gilt

$$B\left(x_0, \frac{\gamma}{2}\right) \subseteq [0, \gamma]^m \subseteq \mathcal{D} \subseteq [0, 1]^m \subseteq B(x_0, \sqrt{m}),$$

wobei x_0 der Vektor mit allen Komponenten gleich $\frac{\gamma}{2}$ ist.

Wir werden beweisen, dass wir das SCHWACHE SEPARATIONS-PROBLEM für (18.3), d. h. für \mathcal{D} und b und $\frac{\delta}{2}$, in $O\left(\frac{nm}{\delta}\right)$ -Zeit lösen können, unabhängig von der Größe des Inputvektors y . Mit Satz 4.19 folgt daraus, dass das SCHWACHE OPTIMIERUNGSPROBLEM in $O\left(m^6 \log^2 \frac{m||b||}{\gamma\delta} + \frac{m^5 n}{\delta} \log \frac{m||b||}{\gamma\delta}\right)$ -Zeit gelöst werden kann. Da $||b|| \leq n$, folgt das Lemma.

Um zu zeigen, wie man das SCHWACHE SEPARATIONS-PROBLEM löst, sei $y \in \mathbb{Q}^m$ gegeben. Wir können annehmen, dass $0 \leq y \leq 1$, da die Aufgabe sonst trivial ist. Beachte nun, dass y genau dann zulässig ist, wenn

$$\max\{yx : x \in \mathbb{Z}_+^m, xs \leq 1\} \leq 1, \quad (18.4)$$

wobei $s = (s_1, \dots, s_m)$ der Vektor der Objektgrößen ist.

Es ist (18.4) eine Art KNAPSACK-PROBLEM, somit besteht keine Hoffnung, es exakt zu lösen. Das ist aber auch nicht nötig, da das SCHWACHE SEPARATIONS-PROBLEM nur eine approximative Lösung verlangt.

Es sei $y' := \lfloor \frac{2n}{\delta} y \rfloor$ (komponentenweise Abrundung). Das Problem

$$\max\{y'x : x \in \mathbb{Z}_+^m, xs \leq 1\} \quad (18.5)$$

kann mit dynamischer Optimierung optimal gelöst werden, wie bei dem EXAKTEN KNAPSACK-ALGORITHMUS in Abschnitt 17.2 (siehe Aufgabe 10, Kapitel 17): Sei $F(0) := 0$ und

$$F(k) := \min\{F(k - y'_i) + s_i : i \in \{1, \dots, m\}, y'_i \leq k\}$$

für $k = 1, \dots, \frac{4n}{\delta}$. Es ist $F(k)$ die minimale Gesamtgröße einer Objektmenge mit Gesamtkosten k (bezüglich y').

Das Maximum in (18.5) ist genau dann kleiner oder gleich $\frac{2n}{\delta}$, wenn $F(k) > 1$ für alle $k \in \{\frac{2n}{\delta} + 1, \dots, \frac{4n}{\delta}\}$. Die für diese Entscheidung benötigte Gesamtzeit ist $O(\frac{mn}{\delta})$. Wir unterscheiden zwei Fälle:

Fall 1: Das Maximum in (18.5) ist kleiner oder gleich $\frac{2n}{\delta}$. Dann ist $\frac{\delta}{2n}y'$ eine zulässige Lösung für (18.3). Ferner gilt $by - b\frac{\delta}{2n}y' \leq b\frac{\delta}{2n}\mathbf{1} = \frac{\delta}{2}$. Damit ist die Aufgabe bezüglich des SCHWACHEN SEPARATIONS-PROBLEMS erledigt.

Fall 2: Es gibt ein $x \in \mathbb{Z}_+^m$ mit $xs \leq 1$ und $y'x > \frac{2n}{\delta}$. Ein solches x können wir leicht aus den Zahlen $F(k)$ in $O(\frac{mn}{\delta})$ -Zeit berechnen. Es gilt $yx \geq \frac{\delta}{2n}y'x > 1$. Somit entspricht x einer Behälterkonfiguration, die die Unzulässigkeit von y beweist. Da $zx \leq 1$ für alle $z \in \mathcal{D}$, ist dies eine trennende Hyperebene. Damit sind wir fertig. \square

Lemma 18.13. (Karmarkar und Karp [1982]) *Sei I eine Instanz des BIN-PACKING-PROBLEMS mit nur m verschiedenen Zahlen, alle größer oder gleich γ . Sei $\delta > 0$. Dann kann man eine zulässige Lösung x des primalen LP (18.2) mit höchstens m nichtverschwindenden Komponenten und mit vom Optimum um höchstens δ abweichendem Zielfunktionswert in einer Zeit finden, die polynomiell in n , $\frac{1}{\delta}$ und $\frac{1}{\gamma}$ ist.*

Beweis: Zunächst lösen wir das duale LP (18.3) approximativ mittels Lemma 18.12. Damit erhalten wir einen Vektor y^* mit $y^*b \geq \text{OPT}(18.3) - \delta$. Es seien $T_{k_1}, \dots, T_{k_{N'}}$ diejenigen Behälterkonfigurationen, die in Fall 2 im vorigen Beweis als trennende Hyperebene aufgetreten waren, und die Einheitsvektoren (d. h. die nur ein Objekt enthaltenden Behälterkonfigurationen). Beachte, dass N' durch die Anzahl der Iterationen des GRÖTSCHEL-LOVÁSZ-SCHRIJVER-ALGORITHMUS (Satz 4.19) beschränkt ist. Somit ist $N' = O\left(m^2 \log \frac{mn}{\gamma \delta}\right)$.

Betrachte das LP

$$\begin{aligned} \max \quad & yb \\ \text{bzgl.} \quad & \sum_{i=1}^m t_{k_j i} y_i \leq 1 \quad (j = 1, \dots, N') \\ & y_i \geq 0 \quad (i = 1, \dots, m). \end{aligned} \tag{18.6}$$

Beachte, dass obiges Verfahren für (18.3) (im Beweis von Lemma 18.12) auch eine gültige Anwendung des GRÖTSCHEL-LOVÁSZ-SCHRIJVER-ALGORITHMUS für (18.6) ist: Das Orakel für das SCHWACHE SEPARATIONS-PROBLEM kann immer dieselbe Antwort wie oben geben. Somit folgt $y^* b \geq \text{OPT}(18.6) - \delta$. Betrachte das LP

$$\begin{aligned} \min \quad & \sum_{j=1}^{N'} x_{k_j} \\ \text{bzgl.} \quad & \sum_{j=1}^{N'} t_{k_j i} x_{k_j} \geq b_i \quad (i = 1, \dots, m) \\ & x_{k_j} \geq 0 \quad (j = 1, \dots, N'). \end{aligned} \tag{18.7}$$

Dies ist das duale LP von (18.6), und es geht aus (18.2) durch die Eliminierung der Variablen x_j für $j \in \{1, \dots, N\} \setminus \{k_1, \dots, k_{N'}\}$ hervor (d. h. diese Variablen werden gezwungen, den Wert Null anzunehmen). Mit anderen Worten, nur N' der N Behälterkonfigurationen können benutzt werden.

Es folgt

$$\text{OPT}(18.7) - \delta = \text{OPT}(18.6) - \delta \leq y^* b \leq \text{OPT}(18.3) = \text{OPT}(18.2).$$

Somit genügt es, (18.7) zu lösen. Aber (18.7) ist ein LP von polynomieller Größe: Es hat N' Variablen und m Nebenbedingungen, kein Element der Matrix ist größer als $\frac{1}{\gamma}$, und keine Komponente der rechten Seite ist größer als n . Mit dem Satz von Khachiyan (Satz 4.18) folgt dann, dass (18.7) in polynomieller Zeit gelöst werden kann. Damit erhalten wir eine optimale Basislösung x (es ist x eine Ecke des Polyeders, somit hat x höchstens m nichtverschwindende Komponenten). \square

Nun wenden wir den FERNANDEZ-DE-LA-VEGA-LUEKER-ALGORITHMUS mit nur einer Modifizierung an: Wir ersetzen die exakte Lösung von (18.2) durch eine Anwendung von Lemma 18.13. Wir fassen zusammen:

Satz 18.14. (Karmarkar und Karp [1982]) *Es gibt ein voll-polynomielles asymptotisches Approximationsschema für das BIN-PACKING-PROBLEM.*

Beweis: Wir wenden Lemma 18.13 mit $\delta = 1$ an und erhalten eine optimale Lösung x von (18.7) mit höchstens m nichtverschwindenden Komponenten. Es

gilt $\|x\| \leq \text{OPT}(18.2) + 1$. Eine Anwendung von Satz 18.9 ergibt eine ganzzahlige Lösung, die höchstens $\lceil \text{OPT}(18.2) \rceil + 1 + \frac{m-1}{2}$ Behälter benutzt, wie in ④ des FERNANDEZ-DE-LA-VEGA-LUEKER-ALGORITHMUS verlangt.

Somit bleibt die Aussage von Satz 18.11 gültig. Da $m \leq \frac{2}{\epsilon^2}$ und $\frac{1}{\gamma} \leq \frac{2}{\epsilon}$ (wir können annehmen, dass $\epsilon \leq 1$), ist die Laufzeit für die Bestimmung von x polynomiell in n und $\frac{1}{\epsilon}$. \square

Die auf diese Weise erzielte Laufzeit ist schlechter als $O(\epsilon^{-40})$ und für praktische Zwecke völlig indiskutabel. Karmarkar und Karp [1982] haben gezeigt, wie man die Anzahl der Variablen in (18.7) auf m reduziert (wobei sich der Optimalwert nur geringfügig ändert) und somit die Laufzeit verbessert (siehe Aufgabe 13). Plotkin, Shmoys und Tardos [1995] haben eine Laufzeit von $O(n \log \epsilon^{-1} + \epsilon^{-6} \log \epsilon^{-1})$ erreicht.

Es sind viele Verallgemeinerungen betrachtet worden. Das zweidimensionale Bin-Packing-Problem, bei dem es um das Platzieren (ohne Rotation) einer gegebenen Menge achsenparalleler Rechtecke in einer möglichst kleinen Anzahl von Einheitsquadranten geht, hat kein asymptotisches Approximationsschema, sofern $P \neq NP$ (Bansal et al. [2006]). Für weitere verwandte Resultate siehe Jansen, Prädel und Schwarz [2009], Bansal und Khan [2014] und die dort zitierten Arbeiten.

Aufgaben

1. Sei k fest. Man beschreibe einen pseudopolynomiellen Algorithmus, welcher für eine gegebene Instanz I des BIN-PACKING-PROBLEMS eine Lösung findet, die höchstens k Behälter benutzt, oder entscheide, dass es keine solche Lösung gibt.
2. Man betrachte das BIN-PACKING-PROBLEM beschränkt auf Instanzen a_1, \dots, a_n mit $a_i > \frac{1}{3}$ für $i = 1, \dots, n$.
 - (a) Man reduziere dieses Problem auf das KARDINALITÄTS-MATCHING-PROBLEM.
 - (b) Man zeige, wie man das Problem in $O(n \log n)$ Laufzeit lösen kann.
3. Man betrachte das QUADRATISCHE ZUORDNUNGSPROBLEM: Für gegebene Matrizen $A, B \in \mathbb{R}_+^{n \times n}$ bestimme man eine Permutation π von $\{1, \dots, n\}$, so dass $\sum_{i,j=1}^n a_{i,j} b_{\pi(i),\pi(j)}$ minimal ist. Man zeige, dass es für dieses Problem keinen k -Approximationsalgorithmus mit k konstant gibt, sofern $P \neq NP$, sogar wenn A eine 0-1-Matrix ist und die Elemente von B eine Metrik definieren.
Hinweis: Man benutze Satz 18.3.
(Queyranne [1986])
4. Man finde eine Instanz I des BIN-PACKING-PROBLEMS, für welche $\text{FF}(I) = 17$, aber $\text{OPT}(I) = 10$.
5. Man implementiere den FIRST-FIT-ALGORITHMUS und den FIRST-FIT-DECREASING-ALGORITHMUS mit $O(n \log n)$ -Laufzeit.

6. Man zeige, dass es keinen Online-Algorithmus für das BIN-PACKING-PROBLEM mit Approximationsgüte kleiner als $\frac{4}{3}$ gibt.

Hinweis: Man beachte, dass wir nicht $P \neq NP$ voraussetzen. Es gibt keinen solchen Algorithmus, ungeachtet der Laufzeit. Man betrachte die Liste bestehend aus n Elementen der Größe $\frac{1}{2} - \epsilon$, gefolgt von n Elementen der Größe $\frac{1}{2} + \epsilon$.

7. Man zeige, dass das BIN-PACKING-PROBLEM mit einer festen Anzahl von unterschiedlichen Objektgrößen in Zeit, die polynomiell in der Zahl der Objekte ist, gelöst werden kann.

Hinweis: Man berechne mittels Dynamischer Optimierung, welche Teilmengen der Objekte i Behältern zugeordnet werden können, wobei $i = 1, 2, \dots$

Bemerkung: Dies ist kein polynomieller Algorithmus, da die Instanz repräsentiert werden kann, indem man nur die Objektgrößen und (in binärer Kodierung) die Zahl der Objekte für jede Größe abspeichert. Allerdings haben Goemans und Rothvoß [2014] einen polynomiellen Algorithmus gefunden.

8. Man zeige, dass jede Bin-Pack-Instanz mit nur m verschiedenen Objektgrößen eine Lösung mit höchstens 2^m verschiedenen Behälterkonfigurationen hat.
Hinweis: In einer Lösung mit mehr Konfigurationen gibt es zwei Behälter, für welche die Quantitäten der hineingepackten Objekte dieselben Paritäten haben.

(Eisenbrand und Shmonin [2006])

9. Man zeige, dass Schritt ② des FERNANDEZ-DE-LA-VEGA-LUEKER-ALGORITHMUS mit $O\left(n \log \frac{1}{\epsilon}\right)$ -Laufzeit implementiert werden kann.
10. Man betrachte das LP (18.3), mit einer Variable y_i für jedes $i = 1, \dots, m$ (d. h. für jede Objektgröße). Es gelte $s_1 > \dots > s_m$. Man zeige, dass es dann eine optimale Lösung mit $y_1 \geq \dots \geq y_m$ gibt.

(Caprara [2008])

- * 11. Man beweise, dass es für jedes $\epsilon > 0$ einen polynomiellen Algorithmus gibt, der für jede Instanz $I = (a_1, \dots, a_n)$ des BIN-PACKING-PROBLEMS eine Verpackung mit der optimalen Anzahl von Behältern findet, die aber die Kapazitätsgrenzen um bis zu ϵ verletzen darf, d. h. ein $f : \{1, \dots, n\} \rightarrow \{1, \dots, \text{OPT}(I)\}$ mit $\sum_{f(i)=j} a_i \leq 1 + \epsilon$ für alle $j \in \{1, \dots, \text{OPT}(I)\}$.

Hinweis: Man benutze Ideen aus Abschnitt 18.2.

(Hochbaum und Shmoys [1987])

12. Man betrachte das folgende MULTIPROCESSOR-SCHEDULING-PROBLEM: Gegeben sei eine endliche Menge A von Jobs, eine positive Zahl $t(a)$ für jedes $a \in A$ (die Ausführungsduer) und eine Zahl m von Maschinen. Man bestimme eine Partition $A = A_1 \dot{\cup} A_2 \dot{\cup} \dots \dot{\cup} A_m$ von A in m paarweise disjunkte Mengen, die $\max_{i=1}^m \sum_{a \in A_i} t(a)$ minimiert.
- Man zeige, dass dieses Problem stark NP -schwer ist.
 - Man zeige, dass ein Greedy-Algorithmus, der Jobs (in beliebiger Reihenfolge) schrittweise jeweils der zur Zeit am wenigsten benutzten Maschine zuteilt, ein 2-Approximationsalgorithmus ist.

- (c) Man zeige, dass das Problem für jedes feste m ein voll-polynomielles Approximationsschema hat.
(Horowitz und Sahni [1976])
 - * (d) Man benutze Aufgabe 11 um zu zeigen, dass das MULTIPROCESSOR-SCHEDULING-PROBLEM ein Approximationsschema hat.
(Hochbaum und Shmoys [1987])
- Bemerkung:* Dieses Problem war das Thema der ersten Arbeit über Approximationsalgorithmen (Graham [1966]). Scheduling-Probleme sind in vielen Variationen studiert worden; siehe z. B. Graham et al. [1979] oder Lawler et al. [1993].
- * 13. Man betrachte das LP (18.6) im Beweis von Lemma 18.13. Alle außer m Nebenbedingungen können entfernt werden, ohne den optimalen Zielfunktionswert zu verändern. Wir können diese m Nebenbedingungen nicht in polynomieller Zeit finden, aber wir können m Nebenbedingungen mit der Eigenschaft finden, dass das Entfernen der restlichen den optimalen Zielfunktionswert nicht allzu sehr erhöht (zum Beispiel um höchstens 1). Wie beweist man dies?
Hinweis: Sei $D^{(0)}$ das LP (18.6). Man konstruiere LPs $D^{(1)}, D^{(2)}, \dots$, indem man schrittweise mehr und mehr Nebenbedingungen entfernt. Bei jeder Iteration erhalten wir eine Lösung $y^{(i)}$ von $D^{(i)}$ mit $b y^{(i)} \geq \text{OPT}(D^{(i)}) - \delta$. Die Menge der Nebenbedingungen wird in $m + 1$ ungefähr gleich große Mengen partitioniert und für jede dieser Mengen prüfen wir, ob sie entfernt werden kann. Diese Prüfung besteht aus der Betrachtung des LP nach dem Entfernen der Nebenbedingungen, nennen wir es \bar{D} , und der Anwendung des GRÖTSCHEL-LOVÁSZ-SCHRIJVER-ALGORITHMUS. Sei \bar{y} eine Lösung von \bar{D} mit $b \bar{y} \geq \text{OPT}(\bar{D}) - \delta$. Gilt $b \bar{y} \leq b y^{(i)} + \delta$, so war die Prüfung erfolgreich und wir setzen $D^{(i+1)} := \bar{D}$ und $y^{(i+1)} := \bar{y}$. Man wähle δ auf geeignete Weise. (Karmarkar und Karp [1982])
 - 14. Man finde eine geeignete Wahl von ϵ als Funktion von $\text{SUM}(I)$, so dass die resultierende Modifizierung des KARMAKAR-KARP-ALGORITHMUS ein polynomieller Algorithmus ist, der stets eine Lösung findet, welche höchstens $\text{OPT}(I) + O((\text{OPT}(I))^{2/3})$ Behälter benutzt.
(Johnson [1982])
- Bemerkung:* Hoberg und Rothvoß [2017] haben gezeigt, wie man eine Lösung mit höchstens $\text{OPT}(I) + O(\log \text{OPT}(I))$ Behältern findet.

Literatur

Allgemeine Literatur:

Coffman, E.G., Garey, M.R., und Johnson, D.S. [1996]: Approximation algorithms for bin-packing; a survey. In: Approximation Algorithms for NP-Hard Problems (D.S. Hochbaum, Hrsg.), PWS, Boston, 1996

Zitierte Literatur:

- Baker, B.S. [1985]: A new proof for the First-Fit Decreasing bin-packing algorithm. *Journal of Algorithms* 6 (1985), 49–70
- Bansal, N., Correa, J.R., Kenyon, C., und Sviridenko, M. [2006]: Bin packing in multiple dimensions: inapproximability results and approximation schemes. *Mathematics of Operations Research* 31 (2006), 31–49
- Bansal, N. und Khan, A. [2014]: Improved approximation algorithm for two-dimensional bin packing. *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms* (2014), 13–25
- Caprara, A. [2008]: Packing d -dimensional bins in d stages. *Mathematics of Operations Research* 33 (2008), 203–215
- Dósa, G., Li, R., Han, X., und Tuza, Z. [2013]: Tight absolute bound for First Fit Descreasing bin-packing: $FFD(L) \leq 11/9 OPT(L) + 6/9$. *Theoretical Computer Science* 510 (2013), 13–61
- Dósa, G., und Sgall, J. [2013]: First fit bin packing: a tight analysis. *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science* (2013), 538–549
- Eisenbrand, F., und Shmonin, G. [2006]: Carathéodory bounds for integer cones. *Operations Research Letters* 34 (2006), 564–568
- Eisemann, K. [1957]: The trim problem. *Management Science* 3 (1957), 279–284
- Fernandez de la Vega, W., und Lueker, G.S. [1981]: Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica* 1 (1981), 349–355
- Garey, M.R., Graham, R.L., Johnson, D.S., und Yao, A.C. [1976]: Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory A* 21 (1976), 257–298
- Garey, M.R., und Johnson, D.S. [1975]: Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing* 4 (1975), 397–411
- Garey, M.R., und Johnson, D.S. [1979]: Computers and Intractability; A Guide to the Theory of NP-Completeness. Freeman, San Francisco 1979, S. 127
- Gilmore, P.C., und Gomory, R.E. [1961]: A linear programming approach to the cutting-stock problem. *Operations Research* 9 (1961), 849–859
- Goemans, M.X., und Rothvoß, T. [2014]: Polynomiality for bin packing with a constant number of item types. *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms* (2014), 830–839
- Graham, R.L. [1966]: Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal* 45 (1966), 1563–1581
- Graham, R.L., Lawler, E.L., Lenstra, J.K., und Rinnooy Kan, A.H.G. [1979]: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: *Discrete Optimization II; Annals of Discrete Mathematics* 5 (P.L. Hammer, E.L. Johnson, B.H. Korte, Hrsg.), North-Holland, Amsterdam 1979, S. 287–326
- Hoberg, R., und Rothvoß, T. [2017]: A logarithmic additive integrality gap for bin packing. *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms* (2017), 2616–2625
- Hochbaum, D.S., und Shmoys, D.B. [1987]: Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM* 34 (1987), 144–162
- Horowitz, E., und Sahni, S.K. [1976]: Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM* 23 (1976), 317–327

- Jansen, K., Prädel, L., und Schwarz, U.M. [2009]: Two for one: tight approximation of 2D bin packing. In: Algorithms and Data Structures – Proceedings of the 11th Algorithms and Data Structures Symposium; LNCS 5664 (F. Dehne, M. Gavrilova, J.-R. Sack, C.D. Tóth, Hrsg.), Springer, Berlin 2009, pp. 399–410
- Johnson, D.S. [1973]: Near-Optimal Bin Packing Algorithms. Doctoral Thesis, Dept. of Mathematics, MIT, Cambridge, MA, 1973
- Johnson, D.S. [1974]: Fast algorithms for bin-packing. Journal of Computer and System Sciences 8 (1974), 272–314
- Johnson, D.S. [1982]: The *NP*-completeness column; an ongoing guide. Journal of Algorithms 3 (1982), 288–300, Abschnitt 3
- Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., und Graham, R.L. [1974]: Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM Journal on Computing 3 (1974), 299–325
- Karmarkar, N., und Karp, R.M. [1982]: An efficient approximation scheme for the one-dimensional bin-packing problem. Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (1982), 312–320
- Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., und Shmoys, D.B. [1993]: Sequencing and scheduling: algorithms and complexity. In: Handbooks in Operations Research and Management Science; Vol. 4 (S.C. Graves, A.H.G. Rinnooy Kan, P.H. Zipkin, Hrsg.), Elsevier, Amsterdam 1993
- Lenstra, H.W. [1983]: Integer Programming with a fixed number of variables. Mathematics of Operations Research 8 (1983), 538–548
- Papadimitriou, C.H. [1994]: Computational Complexity. Addison-Wesley, Reading 1994, S. 204–205
- Plotkin, S.A., Shmoys, D.B., und Tardos, É. [1995]: Fast approximation algorithms for fractional packing and covering problems. Mathematics of Operations Research 20 (1995), 257–301
- Queyranne, M. [1986]: Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. Operations Research Letters 4 (1986), 231–234
- Seiden, S.S. [2002]: On the online bin packing problem. Journal of the ACM 49 (2002), 640–671
- Simchi-Levi, D. [1994]: New worst-case results for the bin-packing problem. Naval Research Logistics 41 (1994), 579–585
- van Vliet, A. [1992]: An improved lower bound for on-line bin packing algorithms. Information Processing Letters 43 (1992), 277–284
- Yue, M. [1991]: A simple proof of the inequality $FFD(L) \leq \frac{11}{9} OPT(L) + 1$, $\forall L$, for the FFD bin-packing algorithm. Acta Mathematicae Applicatae Sinica 7 (1991), 321–331



19 Mehrgüterflüsse und kantendisjunkte Wege

Das MULTICOMMODITY-FLOW-PROBLEM ist eine Verallgemeinerung des MAXIMUM-FLOW-PROBLEMS. In einem gegebenen Digraphen mit Kantenkapazitäten möchten wir nun einen $s-t$ -Fluss für mehrere Paare (s, t) finden (wir sprechen hier von mehreren Gütern), so dass der Gesamtfluss durch jede Kante deren Kapazität nicht übersteigt. Die Paare (s, t) stellen wir in einem zweiten Graphen dar; technisch bedingt repräsentieren wir einen $s-t$ -Fluss durch eine Kante von t nach s . Formal definieren wir:

GERICHTETES MULTICOMMODITY-FLOW-PROBLEM

Instanz: Ein Paar (G, H) von Digraphen mit derselben Knotenmenge. Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und Nachfragen $b : E(H) \rightarrow \mathbb{R}_+$.

Aufgabe: Bestimme eine Familie $(x^f)_{f \in E(H)}$, wobei x^f ein $s-t$ -Fluss mit Wert $b(f)$ in G für jedes $f = (t, s) \in E(H)$ ist und

$$\sum_{f \in E(H)} x^f(e) \leq u(e) \quad \text{für alle } e \in E(G) \text{ gilt.}$$

Es gibt auch eine ungerichtete Version, die wir später betrachten werden. Die Kanten von G heißen **Angebotskanten** und diejenigen von H **Nachfragekanten**. Die Endknoten der Nachfragekanten heißen **Terminale**. Ist $u \equiv 1$ und $b \equiv 1$ und beschränken wir uns auf ganzzahlige x , so haben wir das KANTENDISJUNKTE-WEGE-PROBLEM. Manchmal hat man auch Kantenkosten und möchte dann einen kostenminimalen Mehrgüterfluss finden. Hier sind wir jedoch nur an zulässigen Lösungen interessiert.

Natürlich kann das obige Problem mittels LINEARER OPTIMIERUNG (siehe Satz 4.18) in polynomieller Zeit gelöst werden. Die LP-Formulierungen sind aber recht groß, somit ist es durchaus interessant, dass wir auch einen kombinatorischen Algorithmus zur approximativen Lösung des Problems haben; siehe Abschnitt 19.2. Dieser Algorithmus wird durch eine LP-Formulierung motiviert. Ferner liefert die LP-Dualität eine nützliche gute Charakterisierung unseres Problems, wie wir in Abschnitt 19.1 zeigen werden. Im Gegensatz zu Einzelgut-Flüssen entspricht das duale LP jedoch nicht einem minimalen Schnitt-Problem. In den Abschnitten 19.3 und 19.4 werden wir den sogenannten Max-Flow-Min-Cut-Quotienten ausführlich besprechen.

In vielen Anwendungen ist man an ganzzahligen Flüssen oder an Wegen interessiert, und hierfür ist das KANTENDISJUNKTE-WEGE-PROBLEM die richtige Formulierung. In Abschnitt 8.2 haben wir bereits einen Spezialfall dieses Problems betrachtet; dort ging es um eine notwendige und hinreichende Bedingung für die Existenz von k paarweise kantendisjunkten (oder intern disjunkten) Wegen von s nach t für zwei gegebene Knoten s und t (siehe die Sätze von Menger (Sätze 8.9 und 8.10)). Wir werden beweisen, dass das allgemeine KANTENDISJUNKTE-WEGE-PROBLEM NP -schwer ist, sowohl im gerichteten als auch im ungerichteten Fall. Es gibt jedoch einige interessante Spezialfälle, die in polynomieller Zeit gelöst werden können, wie wir in den Abschnitten 19.5 und 19.6 sehen werden.

19.1 Mehrgüterflüsse

Wir werden primär das Gerichtete Multicommodity-Flow-Problem bearbeiten, erwähnen hier jedoch, dass alle Resultate in diesem Abschnitt auch für den ungerichteten Fall gelten:

UNGERICHTETES MULTICOMMODITY-FLOW-PROBLEM

- Instanz:* Ein Paar (G, H) von ungerichteten Graphen auf derselben Knotenmenge.
 Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$ und Nachfragen $b : E(H) \rightarrow \mathbb{R}_+$.
- Aufgabe:* Bestimme eine Familie $(x^f)_{f \in E(H)}$, wobei x^f ein $s-t$ -Fluss mit Wert $b(f)$ in $(V(G), \{(v, w), (w, v) : \{v, w\} \in E(G)\})$ für jedes $f = \{t, s\} \in E(H)$ ist, und

$$\sum_{f \in E(H)} \left(x^f((v, w)) + x^f((w, v)) \right) \leq u(e)$$

für alle $e = \{v, w\} \in E(G)$ gilt.

Beide Versionen des MULTICOMMODITY-FLOW-PROBLEMS haben eine nahe liegende Formulierung als LP (siehe die LP-Formulierung des MAXIMUM-FLOW-PROBLEMS in Abschnitt 8.1). Somit können beide in polynomieller Zeit gelöst werden (Satz 4.18). Bis heute sind exakte polynomielle Algorithmen, die nicht LINEARE OPTIMIERUNG benutzen, nur für einige wenige Spezialfälle bekannt.

Wir werden nun eine andere LP-Formulierung für das MULTICOMMODITY-FLOW-PROBLEM angeben, die sich als nützlich erweisen wird:

Lemma 19.1. *Sei (G, H, u, b) eine Instanz des (GERICHTETEN oder UNGERICHTETEN) MULTICOMMODITY-FLOW-PROBLEMS. Sei \mathcal{C} die Menge derjenigen Kreise von $G+H$, die genau eine Nachfragekante enthalten. Sei M eine 0-1-Matrix, deren Spalten bzw. Zeilen den Elementen von \mathcal{C} bzw. den Kanten von G entsprechen, wobei $M_{e,C} = 1$ genau dann, wenn $e \in C$. Sei N auf analoge Weise eine 0-1-Matrix, deren Spalten bzw. Zeilen den Elementen von \mathcal{C} bzw. den Kanten von H entsprechen, wobei $N_{f,C} = 1$ genau dann, wenn $f \in C$.*

Dann entspricht jede Lösung des MULTICOMMODITY-FLOW-PROBLEMS mindestens einem Punkt des Polytops

$$\left\{ y \in \mathbb{R}^{\mathcal{C}} : y \geq 0, My \leq u, Ny = b \right\}, \quad (19.1)$$

und jeder Punkt dieses Polytops entspricht einer eindeutig bestimmten Lösung des MULTICOMMODITY-FLOW-PROBLEMS.

Beweis: Um die Notation zu vereinfachen, beweisen wir nur den gerichteten Fall; der ungerichtete Fall folgt dann durch das Ersetzen einer jeden ungerichteten Kante durch den in Abb. 8.2 dargestellten Teilgraphen.

Sei $(x^f)_{f \in E(H)}$ eine Lösung des MULTICOMMODITY-FLOW-PROBLEMS. Für jedes $f = (t, s) \in E(H)$ kann der s - t -Fluss x^f in eine Menge \mathcal{P} von s - t -Wegen und eine Menge \mathcal{Q} von Kreisen zerlegt werden (Satz 8.8): Für jede Nachfragekante f können wir

$$x^f(e) = \sum_{P \in \mathcal{P} \cup \mathcal{Q}: e \in E(P)} w(P)$$

für $e \in E(G)$ schreiben, wobei $w : \mathcal{P} \cup \mathcal{Q} \rightarrow \mathbb{R}_+$. Nun setzen wir $y_{P+f} := w(P)$ für $P \in \mathcal{P}$ und $y_C := 0$ für $f \in C \in \mathcal{C}$ mit $C - f \notin \mathcal{P}$. Offensichtlich ergibt dies einen Vektor $y \geq 0$ mit $My \leq u$ und $Ny = b$.

Sei nun umgekehrt $y \geq 0$ mit $My \leq u$ und $Ny = b$. Dann liefert

$$x^f(e) := \sum_{C \in \mathcal{C}: e, f \in E(C)} y_C$$

eine Lösung des MULTICOMMODITY-FLOW-PROBLEMS. \square

Mittels LP-Dualität können wir nun eine notwendige und hinreichende Bedingung für die Lösbarkeit des MULTICOMMODITY-FLOW-PROBLEMS ableiten. Ferner werden wir auch auf die Verbindung mit dem KANTENDISJUNKTE-WEGE-PROBLEM eingehen.

Definition 19.2. Eine Instanz (G, H) des (GERICHTETEN oder UNGERICHTETEN) KANTENDISJUNKTE-WEGE-PROBLEMS erfüllt das **Distanzkriterium**, falls für jedes $z : E(G) \rightarrow \mathbb{R}_+$ folgende Ungleichung gilt:

$$\sum_{f=(t,s) \in E(H)} \text{dist}_{(G,z)}(s, t) \leq \sum_{e \in E(G)} z(e). \quad (19.2)$$

Eine Instanz (G, H, u, b) des MULTICOMMODITY-FLOW-PROBLEMS erfüllt das **Distanzkriterium**, falls für jedes $z : E(G) \rightarrow \mathbb{R}_+$ folgende Ungleichung gilt:

$$\sum_{f=(t,s) \in E(H)} b(f) \text{dist}_{(G,z)}(s, t) \leq \sum_{e \in E(G)} u(e)z(e).$$

(Im ungerichteten Fall ersetze man (t, s) durch $\{t, s\}$.)

Die linke Seite des Distanzkriteriums kann als eine untere Schranke für die Kosten einer Lösung (bezüglich der Kantenkosten z) angesehen werden und die rechte Seite als eine obere Schranke für die höchstmöglichen Kosten.

Satz 19.3. *Das Distanzkriterium ist notwendig und hinreichend für die Lösbarkeit des MULTICOMMODITY-FLOW-PROBLEMS (sowohl im gerichteten als auch im ungerichteten Fall).*

Beweis: Wiederum betrachten wir nur den gerichteten Fall; der ungerichtete folgt wie früher durch das Ersetzen einer jeden ungerichteten Kante durch den in Abb. 8.2 dargestellten Teilgraphen. Nach Lemma 19.1 hat das MULTICOMMODITY-FLOW-PROBLEM genau dann eine Lösung, wenn das Polyeder $\{y \in \mathbb{R}_+^C : My \leq u, Ny = b\}$ nicht leer ist. Nach Korollar 3.25 ist dieses Polyeder genau dann leer, wenn es Vektoren z, w mit $z \geq 0$, $zM + wN \geq 0$ und $zu + wb < 0$ gibt. (M und N wurden in Lemma 19.1 definiert.)

Aus der Ungleichung $zM + wN \geq 0$ folgt

$$-w_f \leq \sum_{e \in P} z_e$$

für jede Nachfragekante $f = (t, s)$ und jeden s - t -Weg P in G ; somit gilt $-w_f \leq \text{dist}_{(G,z)}(s, t)$. Also gibt es Vektoren z, w mit $z \geq 0$, $zM + wN \geq 0$ und $zu + wb < 0$ genau dann, wenn es einen Vektor $z \geq 0$ gibt mit

$$zu - \sum_{f=(t,s) \in E(H)} \text{dist}_{(G,z)}(s, t) b(f) < 0.$$

Damit ist der Beweis abgeschlossen. \square

In Abschnitt 19.2 werden wir zeigen, wie die LP-Formulierung von Lemma 19.1 und das dazu duale LP zur Konstruktion eines Algorithmus für das MULTICOMMODITY-FLOW-PROBLEM verwendet werden können.

Aus Satz 19.3 folgt, dass das Distanzkriterium für die Lösbarkeit des KANTENDISJUNKTE-WEGE-PROBLEMS notwendig ist, da dieses als ein MULTICOMMODITY-FLOW-PROBLEM mit $b \equiv 1$, $u \equiv 1$ und ganzzahligen Nebenbedingungen betrachtet werden kann. Eine weitere notwendige Bedingung ist die folgende:

Definition 19.4. Eine Instanz (G, H) des (GERICHTETEN oder UNGERICHTETEN) KANTENDISJUNKTE-WEGE-PROBLEMS erfüllt das **Schnittkriterium**, falls für jedes $X \subseteq V(G)$ gilt:

- $|\delta_G^+(X)| \geq |\delta_H^-(X)| \quad \text{im gerichteten Fall, bzw.}$
- $|\delta_G(X)| \geq |\delta_H(X)| \quad \text{im ungerichteten Fall.}$

Eine Instanz (G, H, u, b) des (GERICHTETEN oder UNGERICHTETEN) MULTICOMMODITY-FLOW-PROBLEMS erfüllt das Schnittkriterium, falls für jedes $X \subseteq V(G)$ gilt:

- $u(\delta_G^+(X)) \geq b(\delta_H^-(X)) \quad \text{im gerichteten Fall, bzw.}$
- $u(\delta_G(X)) \geq b(\delta_H(X)) \quad \text{im ungerichteten Fall.}$

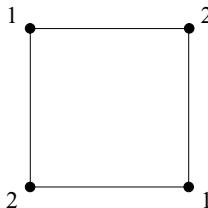
Korollar 19.5. Für eine Instanz (G, H) des (GERICHTETEN oder UNGERICHTETEN) KANTENDISJUNKTE-WEGE-PROBLEMS gelten die folgenden Implikationen: (G, H) hat eine Lösung $\Rightarrow (G, H)$ erfüllt das Distanzkriterium $\Rightarrow (G, H)$ erfüllt das Schnittkriterium.

Beweis: Die erste Implikation folgt aus Satz 19.3. Zum Beweis der zweiten Implikation beachten wir, dass das Schnittkriterium ein Spezialfall des Distanzkriteriums ist, mit Gewichtsfunktionen der folgenden Art für $X \subseteq V(G)$:

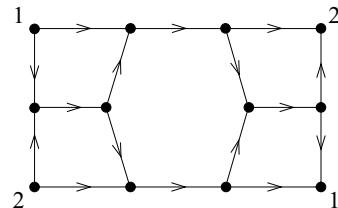
$$z(e) := \begin{cases} 1 & \text{für } e \in \delta^+(X) \text{ (gerichteter Fall) oder } e \in \delta(X) \text{ (ungerichteter Fall)} \\ 0 & \text{sonst.} \end{cases}$$

□

(a)



(b)



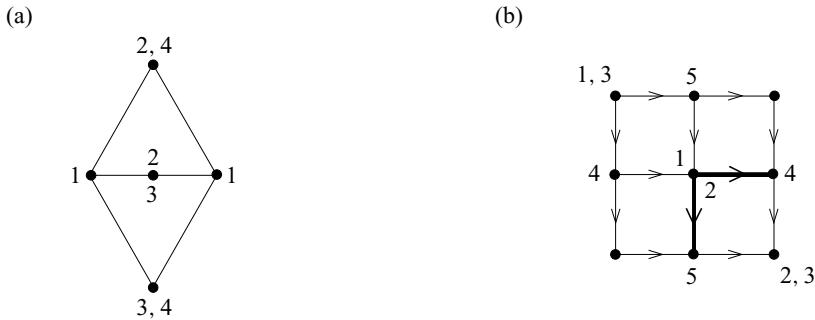


Abbildung 19.2.

kriterium erfüllt (Aufgabe 2), im Gegensatz zum Distanzkriterium, das mittels linearer Optimierung in polynomieller Laufzeit überprüft werden kann.

Für manche Instanzklassen des KANTENDISJUNKTE-WEGE-PROBLEMS ist das Schnittkriterium hinreichend dafür, dass es eine Lösung gibt. Einige Beispiele hierfür werden wir in diesem Kapitel noch kennenlernen. Ein erstes Beispiel ist der Satz von Menger. Um dieses etwas zu verallgemeinern, machen wir von der folgenden Konstruktion Gebrauch.

Lemma 19.6. *Sei (G, H) eine Instanz des (GERICHTETEN oder UNGERICHTETEN) KANTENDISJUNKTE-WEGE-PROBLEMS und $v \in V(G)$.*

Es gehe (G', H') aus (G, H) hervor durch Hinzufügung eines neuen Knotens x und Ausführung des Folgenden für jede Nachfragekante f mit der v inzident ist: Ist $f = \{v, w\}$, so ersetze f durch $\{v, x\}$ und füge eine Angebotskante $\{w, x\}$ hinzu. Ist $f = (v, w)$, so ersetze f durch (v, x) und füge eine Angebotskante (x, w) hinzu. Ist $f = (w, v)$, so ersetze f durch (x, v) , und füge eine Angebotskante (w, x) hinzu.

Dann folgt: (G', H') hat genau dann eine Lösung, wenn (G, H) eine hat. Ferner erfüllt (G', H') genau dann das Schnittkriterium, wenn (G, H) es erfüllt.

Beweis: Jede Lösung für (G', H') liefert eine Lösung für (G, H) , indem man die neuen Angebotskanten weglässt. Umgekehrt gilt: Jede Lösung für (G, H) lässt sich durch Hinzufügung der neuen Angebotskanten in eine Lösung für (G', H') transformieren.

Für das Schnittkriterium haben wir Folgendes: Für jede Menge $X \subseteq V(G') = V(G) \cup \{x\}$ gilt

$$|\delta_{G'}^+(X)| - |\delta_{H'}^-(X)| \geq |\delta_G^+(X \setminus \{x\})| - |\delta_H^-(X \setminus \{x\})|$$

bzw.

$$|\delta_{G'}(X)| - |\delta_{H'}(X)| \geq |\delta_G(X \setminus \{x\})| - |\delta_H(X \setminus \{x\})|,$$

mit Gleichheit falls $|X \cap \{v, x\}| = 1$. □

Satz 19.7. Sei (G, H) eine Instanz des (GERICHTETEN oder UNGERICHTETEN) KANTENDISJUNKTE-WEGE-PROBLEMS mit einem Knoten v für den Folgendes gilt:

- (a) $f \in \delta^+(v)$ für alle $f \in E(H)$, oder
- (b) $f \in \delta^-(v)$ für alle $f \in E(H)$ im gerichteten Fall, oder
- (c) $f \in \delta(v)$ für alle $f \in E(H)$ im ungerichteten Fall.

Dann folgt: (G, H) hat genau dann eine Lösung, wenn das Schnittkriterium erfüllt ist.

Beweis: Nach Lemma 19.6 können wir uns beschränken auf den Fall, dass H nur eine Menge paralleler Kanten hat. Dann ist die Aussage, dass das Schnittkriterium hinreichend ist, eine Umformulierung des Satzes von Menger (Satz 8.9). \square

19.2 Algorithmen für Mehrgüterflüsse

Die Definition des MULTICOMMODITY-FLOW-PROBLEMS führt sofort zu einer LP-Formulierung polynomieller Größe. Obwohl wir auf diesem Weg einen polynomiellen Algorithmus erhalten, kann dieser nicht zur Lösung großer Instanzen verwendet werden: Die Anzahl der Variablen ist enorm. Die mittels Lemma 19.1 gegebene LP-Formulierung (19.1) sieht noch schlimmer aus, da sie eine exponentiell wachsende Anzahl von Variablen hat. Trotzdem ist Letztere in der Praxis von viel größerem Nutzen. Dies werden wir nun erläutern.

Da wir nur an einer zulässigen Lösung interessiert sind, betrachten wir das LP

$$\max\{0y : y \geq 0, My \leq u, Ny = b\}$$

und das dazu duale LP $\min\{zu + wb : z \geq 0, zM + wN \geq 0\}$, welches wir folgendermaßen schreiben können:

$$\min\{zu + wb : z \geq 0, \text{dist}_{(G,z)}(s, t) \geq -w(f) \text{ für alle } f = (t, s) \in E(H)\}.$$

(Im ungerichteten Fall ersetzen wir (t, s) durch $\{t, s\}$.) Das duale LP hat nur $|E(G)| + |E(H)|$ Variablen, aber eine exponentielle Anzahl von Nebenbedingungen. Dies ist jedoch hier nicht wichtig, da das SEPARATIONS-PROBLEM mittels $|E(H)|$ Kürzeste-Wege-Berechnungen gelöst werden kann; da man nur nichtnegative Vektoren z zu betrachten braucht, kann man hier DIJKSTRAS ALGORITHMUS benutzen. Ist das duale LP unbeschränkt, so ist die Unzulässigkeit des primalen LP bewiesen. Andernfalls können wir das duale LP lösen, erhalten dadurch aber i. A. keine primale Lösung.

Ford und Fulkerson [1958] haben vorgeschlagen, den oben beschriebenen Weg zur direkten Lösung des primalen LP zu benutzen, zusammen mit dem SIMPLEX-ALGORITHMUS. Da die meisten Variablen bei jeder Iteration des SIMPLEXALGORITHMUS verschwinden, braucht man nur diejenigen Variablen zu speichern, für die die Nichtnegativitätsnebenbedingung $y_C \geq 0$ nicht zur aktuellen Menge J der

aktiven Zeilen gehört. Die restlichen Variablen werden nicht explizit gespeichert, sondern bei Bedarf „generiert“ (d. h., wenn die Nichtnegativitätsnebenbedingung inaktiv wird). Die Entscheidung, welche Variable bei jedem Schritt generiert werden muss, ist äquivalent zum SEPARATIONS-PROBLEM für das duale LP und lässt sich im vorliegenden Fall auf ein KÜRZESTE-WEGE-PROBLEM zurückführen. Diese sogenannte Spaltenerzeugungsmethode kann in der Praxis recht effizient sein.

Sogar bei diesen Verfahren bleiben viele praktische Instanzen übrig, die nicht optimal gelöst werden können. Obiges Schema führt aber auch zu einem Approximationsalgorithmus. Zunächst formulieren wir unser Problem als ein Optimierungsproblem:

MAXIMUM-MULTICOMMODITY-FLOW-PROBLEM

Instanz: Ein Paar (G, H) von Digraphen auf derselben Knotenmenge.
Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$.

Aufgabe: Bestimme eine Familie $(x^f)_{f \in E(H)}$, wobei x^f ein $s-t$ -Fluss in G für jedes $f = (t, s) \in E(H)$ ist und $\sum_{f \in E(H)} x^f(e) \leq u(e)$ für alle $e \in E(G)$ gilt, so dass der Wert des Gesamtflusses $\sum_{f \in E(H)} \text{value}(x^f)$ maximiert wird.

Es gibt auch andere interessante Formulierungen; z. B. kann man einen Fluss suchen, der den größtmöglichen Anteil der gegebenen Nachfragen erfüllt (das CONCURRENT-FLOW-PROBLEM), oder der alle Nachfragen mit den kleinstmöglichen Kapazitätsverletzungen erfüllt. Ferner kann man auch Kantenkosten einführen. Hier betrachten wir nur obiges MAXIMUM-MULTICOMMODITY-FLOW-PROBLEM; die oben erwähnten und andere ähnliche Probleme können auf ähnliche Weise angegangen werden.

Wir betrachten wieder unsere LP-Formulierung

$$\max \left\{ \sum_{P \in \mathcal{P}} y(P) : y \geq 0, \sum_{P \in \mathcal{P}: e \in E(P)} y(P) \leq u(e) \text{ für alle } e \in E(G) \right\},$$

wobei \mathcal{P} die Familie der $s-t$ -Wege in G für alle $(t, s) \in E(H)$ ist, und auch das dazu duale LP

$$\min \left\{ zu : z \geq 0, \sum_{e \in E(P)} z(e) \geq 1 \text{ für alle } P \in \mathcal{P} \right\}.$$

Wir werden einen primal-dualen Algorithmus beschreiben, der auf diesen LP-Formulierungen basiert und werden sehen, dass dieser ein voll-polynomielles Approximationsschema ist. Dieser Algorithmus arbeitet schrittweise mit einem primalen Vektor $y \geq 0$, der nicht notwendigerweise eine zulässige primale Lösung ist, da es verletzte Kapazitätsschranken geben kann. Am Anfang ist $y = 0$. Am Ende werden wir y mit einer geeigneten Konstante multiplizieren, damit alle Nebenbedingungen erfüllt sind. Um y effizient zu speichern, speichern wir die Familie

$\mathcal{P}' \subseteq \mathcal{P}$ der Wege P mit $y(P) > 0$; im Gegensatz zu \mathcal{P} ist die Kardinalität von \mathcal{P}' polynomiell beschränkt.

Der Algorithmus arbeitet auch mit einem dualen Vektor $z \geq 0$. Am Anfang ist $z(e) = \delta$ für alle $e \in E(G)$, wobei δ von n und dem Fehlerparameter ϵ abhängt. In jeder Iteration findet er eine maximal verletzte duale Nebenbedingung (entsprechend einem kürzesten s - t -Weg für $(t, s) \in E(H)$ bezüglich der Kantenlängen z) und vergrößert z und y entlang diesem Weg:

MULTICOMMODITY-FLOW-APPROXIMATIONSSCHEMA

Input: Ein Paar (G, H) von Digraphen mit derselben Knotenmenge.
 Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+ \setminus \{0\}$. Eine Zahl ϵ mit $0 < \epsilon \leq \frac{1}{2}$.
Output: Zahlen $y : \mathcal{P} \rightarrow \mathbb{R}_+$ mit $\sum_{P \in \mathcal{P}: e \in E(P)} y(P) \leq u(e)$ für alle $e \in E(G)$.

- ① Setze $y(P) := 0$ für alle $P \in \mathcal{P}$.
 Setze $\delta := (n(1 + \epsilon))^{-\lceil \frac{5}{\epsilon} \rceil}(1 + \epsilon)$ und $z(e) := \delta$ für alle $e \in E(G)$.
- ② Sei $P \in \mathcal{P}$ mit $z(E(P))$ minimal.
If $z(E(P)) \geq 1$ **then go to** ④.
 Sei $\gamma := \min_{e \in E(P)} u(e)$.
 Setze $y(P) := y(P) + \gamma$.
 Setze $z(e) := z(e) \left(1 + \frac{\epsilon \gamma}{u(e)}\right)$ für alle $e \in E(P)$.
Go to ②.
- ④ Sei $\xi := \max_{e \in E(G)} \frac{1}{u(e)}$
 $\sum_{P \in \mathcal{P}: e \in E(P)} y(P)$.
 Setze $y(P) := \frac{y(P)}{\xi}$ für alle $P \in \mathcal{P}$.

Dieser Algorithmus geht auf Young [1995] und auf Garg und Könemann [2007] zurück und basiert auf früheren Arbeiten von Shahrokh und Matula [1990], Shmoys [1996] und anderen.

Satz 19.8. (Garg und Könemann [2007]) *Das MULTICOMMODITY-FLOW-APPROXIMATIONSSCHEMA liefert eine zulässige Lösung, deren Gesamtfluss mindestens den Wert $\frac{1}{1+\epsilon} \text{OPT}(G, H, u)$ hat. Die Laufzeit ist $O\left(\frac{1}{\epsilon^2} km(m + n \log n) \log n\right)$, wobei $k = |E(H)|$, $n = |V(G)|$ und $m = |E(G)|$; somit ist dies ein voll-polynomiales Approximationsschema.*

Beweis: In jeder Iteration wird der Wert $z(e)$ für mindestens eine Kante e (die Bottleneck-Kante) um den Faktor $1 + \epsilon$ vergrößert. Da eine Kante e mit $z(e) \geq 1$ nie wieder in irgendeinem Weg benutzt wird, ist die Gesamtanzahl der Iterationen gleich $t \leq m \lceil \log_{1+\epsilon}(\frac{1}{\delta}) \rceil$. In jeder Iteration müssen wir k Instanzen des KÜRZESTE-WEGE-PROBLEMS mit nichtnegativen Gewichten lösen, um P zu bestimmen. Mit DIJKSTRAS ALGORITHMUS (Satz 7.4) erhalten wir eine Gesamlauf-

zeit von $O(tk(m + n \log n)) = O\left(km(m + n \log n) \log_{1+\epsilon}\left(\frac{1}{\delta}\right)\right)$. Die angegebene Laufzeit folgt hieraus mittels

$$\log_{1+\epsilon}\left(\frac{1}{\delta}\right) = \frac{\log\left(\frac{1}{\delta}\right)}{\log(1+\epsilon)} \leq \frac{\left\lceil \frac{5}{\epsilon} \right\rceil \log(2n)}{\frac{\epsilon}{2}} = O\left(\frac{\log n}{\epsilon^2}\right)$$

für $0 < \epsilon \leq 1$; hier haben wir $\log(1+\epsilon) \geq \frac{\epsilon}{2}$ gebraucht.

Ferner müssen wir prüfen, dass die maximale Anzahl der zum Speichern aller in den Berechnungen vorkommenden Zahlen benötigten Bits durch ein Polynom in $\log n + \text{size}(u) + \text{size}(\epsilon) + \frac{1}{\epsilon}$ beschränkt ist. Dies ist klar für die y -Variablen. Die Zahl δ kann mit $O\left(\frac{1}{\epsilon} \text{size}(n(1+\epsilon)) + \text{size}(\epsilon)\right) = O\left(\frac{1}{\epsilon}(\log n + \text{size}(\epsilon))\right)$ Bits gespeichert werden. Zur Betrachtung der z -Variablen nehmen wir an, dass u ganzzahlig ist; anderenfalls multiplizieren wir alle Kapazitäten vorab mit dem Produkt der Nenner (siehe Proposition 4.1). Damit sind die Nenner der z -Variablen zu jeder Zeit beschränkt durch das Produkt aller Kapazitäten und des Nenners von δ . Da der Zähler höchstens das Doppelte des Nenners ist, haben wir gezeigt, dass die Größe aller Zahlen tatsächlich polynomiell in der Inputgröße und $\frac{1}{\epsilon}$ ist.

Die Zulässigkeit der Lösung wird durch ④ gewährleistet.

Beachte, dass wir bei jeder Addition von γ Flusseinheiten in Kante e das Gewicht $z(e)$ um den Faktor $\left(1 + \frac{\epsilon\gamma}{u(e)}\right)$ vergrößern. Dieser Wert ist mindestens $(1+\epsilon)^{\frac{\gamma}{u(e)}}$, da $1+\epsilon a \geq (1+\epsilon)^a$ für alle $0 \leq a \leq 1$ (für $a \in \{0, 1\}$ wird die Ungleichung mit Gleichheit erfüllt und die linke Seite ist linear in a , während die rechte Seite konvex ist). Sobald $z(e) \geq 1$ ist, wird e nicht wieder benutzt, somit können wir höchstens $u(e)(1 + \log_{1+\epsilon}\left(\frac{1}{\delta}\right))$ Flusseinheiten in Kante e addieren. Damit folgt

$$\xi \leq 1 + \log_{1+\epsilon}\left(\frac{1}{\delta}\right) = \log_{1+\epsilon}\left(\frac{1+\epsilon}{\delta}\right). \quad (19.3)$$

Sei $z^{(i)}$ der Vektor z nach Iteration i . Sei ferner P_i bzw. γ_i der Weg P bzw. die Zahl γ in Iteration i . Es gilt $z^{(i)}u = z^{(i-1)}u + \epsilon\gamma_i \sum_{e \in E(P_i)} z^{(i-1)}(e)$, somit folgt $(z^{(i)} - z^{(0)})u = \epsilon \sum_{j=1}^i \gamma_j \alpha(z^{(j-1)})$, wobei $\alpha(z) := \min_{P \in \mathcal{P}} z(E(P))$. Setze $\beta := \min \left\{ zu : z \in \mathbb{R}_+^{E(G)}, \alpha(z) \geq 1 \right\}$. Dann folgt $(z^{(i)} - z^{(0)})u \geq \beta \alpha(z^{(i)} - z^{(0)})$ und somit $(\alpha(z^{(i)}) - \delta n)\beta \leq \alpha(z^{(i)} - z^{(0)})\beta \leq (z^{(i)} - z^{(0)})u$. Damit bekommen wir

$$\alpha(z^{(i)}) \leq \delta n + \frac{\epsilon}{\beta} \sum_{j=1}^i \gamma_j \alpha(z^{(j-1)}). \quad (19.4)$$

Nun beweisen wir

$$\delta n + \frac{\epsilon}{\beta} \sum_{j=1}^i \gamma_j \alpha(z^{(j-1)}) \leq \delta n e^{\left(\frac{\epsilon}{\beta} \sum_{j=1}^i \gamma_j\right)} \quad (19.5)$$

mittels Induktion über i (hier bezeichnet e die Basis des natürlichen Logarithmus). Der Fall $i = 0$ ist trivial. Für $i > 0$ haben wir

$$\begin{aligned}\delta n + \frac{\epsilon}{\beta} \sum_{j=1}^i \gamma_j \alpha(z^{(j-1)}) &= \delta n + \frac{\epsilon}{\beta} \sum_{j=1}^{i-1} \gamma_j \alpha(z^{(j-1)}) + \frac{\epsilon}{\beta} \gamma_i \alpha(z^{(i-1)}) \\ &\leq \left(1 + \frac{\epsilon}{\beta} \gamma_i\right) \delta n e^{\left(\frac{\epsilon}{\beta} \sum_{j=1}^{i-1} \gamma_j\right)}\end{aligned}$$

mittels (19.4) und der Induktionsvoraussetzung. Mit $1+x < e^x$ für alle $x > 0$ ist (19.5) bewiesen.

Insbesondere folgt mit (19.4), (19.5) und dem Terminierungskriterium, dass

$$1 \leq \alpha(z^{(t)}) \leq \delta n e^{\left(\frac{\epsilon}{\beta} \sum_{j=1}^t \gamma_j\right)},$$

also ist $\sum_{j=1}^t \gamma_j \geq \frac{\beta}{\epsilon} \ln\left(\frac{1}{\delta n}\right)$. Beachte nun, dass der vom Algorithmus berechnete Wert des Gesamtflusses gleich $\sum_{P \in \mathcal{P}} y(P) = \frac{1}{\xi} \sum_{j=1}^t \gamma_j$ ist. Mit Obigem und (19.3), sowie der Wahl von δ , ist dies mindestens gleich

$$\begin{aligned}\frac{\beta \ln\left(\frac{1}{\delta n}\right)}{\epsilon \log_{1+\epsilon}\left(\frac{1+\epsilon}{\delta}\right)} &= \frac{\beta \ln(1+\epsilon)}{\epsilon} \cdot \frac{\ln\left(\frac{1}{\delta n}\right)}{\ln\left(\frac{1+\epsilon}{\delta}\right)} \\ &= \frac{\beta \ln(1+\epsilon)}{\epsilon} \cdot \frac{\left(\lceil \frac{5}{\epsilon} \rceil - 1\right) \ln(n(1+\epsilon))}{\lceil \frac{5}{\epsilon} \rceil \ln(n(1+\epsilon))} \\ &\geq \frac{\beta(1 - \frac{\epsilon}{5}) \ln(1+\epsilon)}{\epsilon}.\end{aligned}$$

Beachte ferner, dass β der Optimalwert des dualen LP ist, also auch des primalen LP nach dem Dualitätssatz (Satz 3.20). Ferner gilt $\ln(1+\epsilon) \geq \epsilon - \frac{\epsilon^2}{2}$ (diese Ungleichung ist trivial für $\epsilon = 0$ und die Ableitung der linken Seite ist für jedes $\epsilon > 0$ größer als die der rechten). Demnach folgt für $\epsilon \leq \frac{1}{2}$:

$$\frac{(1 - \frac{\epsilon}{5}) \ln(1+\epsilon)}{\epsilon} \geq \left(1 - \frac{\epsilon}{5}\right) \left(1 - \frac{\epsilon}{2}\right) = \frac{1 + \frac{3}{10}\epsilon - \frac{6}{10}\epsilon^2 + \frac{1}{10}\epsilon^3}{1 + \epsilon} \geq \frac{1}{1 + \epsilon}.$$

Daraus folgern wir, dass der Algorithmus eine Lösung findet, für die der Wert des Gesamtflusses mindestens gleich $\frac{1}{1+\epsilon} \text{OPT}(G, H, u)$ ist. \square

Ein anderer Algorithmus, der dieselbe Laufzeit liefert (mittels einer komplizierteren Analyse), stammt von Grigoriadis und Khachiyan [1996] und wurde früher veröffentlicht. Fleischer [2000] verbesserte die Laufzeit des obigen Algorithmus um den Faktor k . Sie bemerkte, dass es in ② genügt, einen approximativen kürzesten Weg zu berechnen, und benutzte diese Tatsache um zu zeigen, dass es nicht notwendig ist, eine Kürzeste-Wege-Berechnung für jedes $(t, s) \in E(H)$ in jeder Iteration durchzuführen. Siehe auch Karakostas [2008], Müller, Radke und Vygen [2011], Bienstock und Iyengar [2006] und Chudak und Eleutério [2005].

19.3 Das Sparsest-Cut-Problem und der Max-Flow-Min-Cut-Quotient

Wir betrachten das folgende Problem:

ALLGEMEINES SPARSEST-CUT-PROBLEM

Instanz: Eine Instanz (G, H, u, b) des UNGERICHTETEN MULTICOMMODITY-FLOW-PROBLEMS.

Aufgabe: Bestimme eine Menge $X \subset V(G)$ mit $b(\delta_H(X)) > 0$ und $\frac{u(\delta_G(X))}{b(\delta_H(X))}$ minimal.

Der Spezialfall wo H der vollständige ungerichtete Graph ist und $b(f) = 1$ für alle $f \in E(H)$, heißt SPARSEST-CUT-PROBLEM: Hier suchen wir eine nichtleere echte Teilmenge X der Knotenmenge mit $\frac{u(\delta_G(X))}{|X||V(G)\setminus X|}$ minimal. Der entsprechende Spezialfall des UNGERICHTETEN MULTICOMMODITY-FLOW-PROBLEMS heißt UNIFORMES MULTICOMMODITY-FLOW-PROBLEM.

Der kleinste Wert von $\frac{u(\delta_G(X))}{b(\delta_H(X))}$ ist offensichtlich eine obere Schranke des Optimalwertes der Instanz (G, H, u, b) des CONCURRENT-FLOW-PROBLEMS, welches folgendermaßen geschrieben werden kann:

$$\max \left\{ \lambda : y(P) \geq 0 \ (P \in \mathcal{P}), \sum_{P \in \mathcal{P}_f} y(P) \geq \lambda b(f) \ (f \in E(H)), \sum_{P \in \mathcal{P}: e \in E(P)} y(P) \leq u(e) \ (e \in E(G)) \right\}, \quad (19.6)$$

wobei \mathcal{P}_f die Familie der $s-t$ -Wege in G mit $f = \{t, s\} \in E(H)$ ist, und $\mathcal{P} = \bigcup_{f \in E(H)} \mathcal{P}_f$.

Führen wir die Entscheidungsvariablen $z(e) \in \{0, 1\}$ für $e \in E(G)$ ein, wobei z der Inzidenzvektor von $\delta_G(X)$ sei, so können wir das ALLGEMEINE SPARSEST-CUT-PROBLEM als ganzzahliges nichtlineares Programm darstellen:

$$\min \left\{ \frac{u^\top z}{b^\top w} : z \in \{0, 1\}^{E(G)}, w \in \{0, 1\}^{E(H)}, b^\top w > 0, \sum_{e \in E(P)} z(e) \geq w(f) \ (P \in \mathcal{P}_f, f \in E(H)) \right\}. \quad (19.7)$$

Proposition 19.9. Das ALLGEMEINE SPARSEST-CUT-PROBLEM ist äquivalent mit (19.7): Die Optimalwerte sind gleich, und man kann in linearer Zeit aus einer Lösung des Letzteren eine wenigstens genauso gute Lösung des Ersten berechnen, und umgekehrt.

Beweis: Für eine gegebene Menge $X \subset V(G)$ mit $b(\delta_H(X)) > 0$ sei z bzw. w der Inzidenzvektor von $\delta_G(X)$ bzw. $\delta_H(X)$. Dann ist (z, w) eine zulässige Lösung von (19.7), und es gilt $\frac{u^\top z}{b^\top w} = \frac{u(\delta_G(X))}{b(\delta_H(X))}$.

Zum Beweis der umgekehrten Richtung, sei (z, w) eine zulässige Lösung von (19.7) und seien X_1, \dots, X_p die Zusammenhangskomponenten von $(V(G), \{e \in E(G) : z(e) = 0\})$. Dann gilt $\min_{i=1}^p \frac{u(\delta_G(X_i))}{b(\delta_H(X_i))} \leq \frac{\sum_{i=1}^p u(\delta_G(X_i))}{\sum_{i=1}^p b(\delta_H(X_i))} \leq \frac{2u^\top z}{2b^\top w} = \frac{u^\top z}{b^\top w}$. \square

Jede zulässige Lösung von (19.7) kann man zu einer zulässigen Lösung des folgenden LP (mittels Multiplikation sämtlicher Variablen mit einer geeigneten positiven Konstante) skalieren:

$$\begin{aligned} \min \left\{ u^\top z : z \in \mathbb{R}_+^{E(G)}, w \in \mathbb{R}_+^{E(H)}, b^\top w = 1, \right. \\ \left. \sum_{e \in E(P)} z(e) \geq w(f) \quad (P \in \mathcal{P}_f, f \in E(H)) \right\}, \end{aligned} \quad (19.8)$$

natürlich mit demselben Zielfunktionswert. Somit kann (19.8) als eine LP-Relaxierung betrachtet werden.

Lemma 19.10. *Das LP (19.8) kann in polynomieller Zeit gelöst werden.*

Beweis: Ist t für irgendein $f = \{t, s\} \in E(H)$ mit $b(f) > 0$ nicht erreichbar von s in G aus, so ist das LP leicht zu lösen und der Optimalwert ist Null. Andernfalls gibt es eine optimale Lösung (z, w) mit $w(f) = \text{dist}_{(G,z)}(s, t)$ für alle $f = \{t, s\} \in E(H)$. Nun fügen wir neue Variablen $d(\{a, b\})$ für $a, b \in V(G)$ hinzu und ersetzen die Nebenbedingungen $\sum_{e \in E(P)} z(e) \geq w(f)$ für $P \in \mathcal{P}_f$ und $f \in E(H)$ durch die äquivalenten Nebenbedingungen $d(e) \leq z(e)$ für $e \in E(G)$, $w(f) \leq d(f)$ für $f \in E(H)$ und $d(\{a, c\}) \leq d(\{a, b\}) + d(\{b, c\})$ für $a, b, c \in V(G)$. Damit bekommen wir ein LP polynomieller Größe und können nun Satz 4.18 anwenden. \square

Es gibt jedoch auch ein kombinatorisches voll-polynomielles Approximationsschema analog dem in Abschnitt 19.2, da (19.8) das duale LP von (19.6), dem CONCURRENT-FLOW-PROBLEM ist (siehe Aufgabe 5).

Der größte Wert des folgenden Quotienten: Optimalwert von (19.7) durch Optimalwert von (19.8), heißt **Max-Flow-Min-Cut-Quotient** der Multicommodity-Flow-Instanz. Sein Wert kann bis $\Theta(\log n)$ groß sein, wo $n = |V(G)|$, sogar im uniformen Fall (d. h. H ist der vollständige Graph und $b \equiv 1$). Ein Beispiel hierfür ist ein Expander-Graph mit Einheitskapazitäten (siehe Aufgabe 6). Im nächsten Abschnitt zeigen wir, dass er niemals schlechter ist.

19.4 Der Satz von Leighton und Rao

Wir werden eine Lösung (z, w) des LP (19.8) benutzen, um einen „sparse“ Schnitt zu berechnen. Ist (z, w) eine optimale Lösung, so ist $W := u^\top z$ eine untere Schranke für die sogenannte „sparsity“ $\min_{X \subset V(G)} \frac{u(\delta_G(X))}{|X||V(G) \setminus X|}$. Im Folgenden werden wir w nicht gebrauchen. Nach Leighton und Rao [1999] partitionieren wir zunächst unseren Graphen in Mengen kleineren Durchmessers, so dass die Kapazität des Multischritts recht klein ist.

Lemma 19.11. (Leighton und Rao [1999]) Sei G ein ungerichteter Graph mit Kantengewichten $z : E(G) \rightarrow \mathbb{R}_+$ und Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$. Wir definieren den **Durchmesser** einer nichtleeren Menge $X \subseteq V(G)$ als $\max_{v, v' \in X} \text{dist}_{(G, z)}(v, v')$. Ferner sei d eine positive Zahl.

Es kann $V(G)$ in Mengen V_1, \dots, V_k partitioniert werden, jede mit Durchmesser höchstens gleich d , so dass die Gesamtkapazität der Kanten im Multischnitt höchstens gleich $\frac{8W \log n}{d}$ ist, wobei $W := \sum_{e \in E(G)} u(e)z(e)$.

Ferner kann man eine solche Partitionierung in $O(m \log n)$ -Zeit berechnen, wobei $n := |V(G)|$ und $m := |E(G)|$.

Beweis: Zunächst besprechen wir einige triviale Fälle. Sei $U := u(E(G))$. Gilt $U = 0$ oder $d \leq \frac{8W \log n}{U}$, so genügt die Partitionierung in einelementige Knotenmengen (singletons). Gilt $W = 0$, so nehme man die Zusammenhangskomponenten von $(V(G), \{e \in E(G) : z(e) = 0\})$.

Andernfalls haben wir $W > 0$ und es gilt $\epsilon := \frac{2W \log n}{Ud} < \frac{1}{4}$. Setze $z'(e) := \lceil Uz(e)/W \rceil$ für $e \in E(G)$. Beachte, dass $U \leq \sum_{e \in E(G)} z'(e)u(e) < 2U$.

Nun wenden wir das folgende Verfahren an, beginnend mit $i := 1$ und $G_1 := G$. Wähle $v_i \in V(G_i)$. Für $x \in \mathbb{Z}_+$ sei $U_x^i := \frac{2U}{n-1} + \sum_{\xi=0}^{x-1} u(\delta_{G_i}(B_\xi^i))$, wobei B_ξ^i die Menge derjenigen Knoten sei, deren Distanz von v_i in (G_i, z') höchstens ξ beträgt. Wähle $r_i \geq 0$ so klein wie möglich, so dass $U_{r_i+1}^i < 2^{\epsilon} U_{r_i}^i$ gilt. Setze $V_i := B_{r_i}^i$. Setze ferner $G_{i+1} := G_i - V_i$, erhöhe i um Eins, und iteriere bis keine Knoten mehr übrig sind.

Damit erreichen wir eine Partitionierung $V(G) = V_1 \dot{\cup} \dots \dot{\cup} V_k$. Für jedes i gilt $U_{r_i}^i \geq 2^{\epsilon r_i} U_0^i = 2^{\epsilon r_i} \frac{2U}{n-1}$ und damit folgt $2^{\epsilon r_i} \frac{2U}{n-1} \leq U_{r_i}^i \leq \frac{2U}{n-1} + \sum_{e \in E(G)} z'(e)u(e) \leq \frac{2Un}{n-1}$. Somit ist $r_i \leq \frac{\log n}{\epsilon}$. Für je zwei Knoten $v, v' \in V_i$ gilt $\text{dist}_{(G, z)}(v, v') \leq \frac{W}{U} \text{dist}_{(G, z')}(v, v') \leq \frac{W}{U} (\text{dist}_{(G, z')}(v_i, v) + \text{dist}_{(G, z')}(v_i, v')) \leq \frac{W}{U} 2r_i \leq \frac{2W \log n}{U\epsilon} = d$.

Wir haben $u(\delta_{G_i}(V_i)) = U_{r_i+1}^i - U_{r_i}^i < (2^\epsilon - 1)U_{r_i}^i \leq \epsilon U_{r_i}^i$ für $i = 1, \dots, k$. Durch Summieren bekommen wir

$$\begin{aligned} u(\delta_G(V_1, \dots, V_k)) &= \sum_{i=1}^k u(\delta_{G_i}(V_i)) \\ &\leq \sum_{i=1}^k \epsilon U_{r_i}^i \\ &= \epsilon \left(\frac{2Uk}{n-1} + \sum_{i=1}^k \sum_{\xi=0}^{r_i-1} u(\delta_{G_i}(B_\xi^i)) \right) \\ &\leq \epsilon \left(2U + \sum_{e \in E(G)} z'(e)u(e) \right) \\ &< 4\epsilon U. \end{aligned}$$

Dieses Verfahren kann wie DIJKSTRAS ALGORITHMUS implementiert werden. In jeder Iteration brauchen wir nur $x = 0$ und die Werte von $x > 0$ mit $\delta_{G_i}(B_x^i) \neq$

$\delta_{G_i}(B_{x-1}^i)$ zu betrachten. Nach jedem dieser Schritte prüfen wir, ob $r_i = x$. Sonst berechnen wir die Zahl $x' = x + 1 + \lfloor 1/(2^\epsilon - 1) - U_x^i/u(\delta_{G_i}(B_x^i)) \rfloor$. Bleibt $\delta_{G_i}(B_\xi^i)$ konstant für $\xi = x, x+1, \dots, x'$, so ist x' die kleinste ganze Zahl für die $u(\delta_{G_i}(B_{x'}^i)) = u(\delta_{G_i}(B_x^i)) < (2^\epsilon - 1)(U_x^i + (x' - x)u(\delta_{G_i}(B_x^i))) = (2^\epsilon - 1)U_{x'}^i$, und damit die gesuchte Zahl r_i .

Also brauchen wir $O(\sum_{w \in B_{r_i}^i} (1 + |\delta(w)|))$ Operationen in Iteration i und $O(m + n \log n)$ -Gesamtzeit für Fibonacci-Heaps. \square

Wir sind nun in der Lage, den Hauptsatz von Leighton und Rao [1999] zu beweisen:

Satz 19.12. *Sei G ein Graph mit Kantengewichten $z : E(G) \rightarrow \mathbb{R}_+$ und Kapazitäten $u : E(G) \rightarrow \mathbb{R}_+$. Angenommen, es gelte $\sum_{\{v, v'\} \in \binom{V(G)}{2}} \text{dist}_{(G,z)}(v, v') = 1$. Sei $n := |V(G)|$. Dann kann man eine nichtleere echte Teilmenge $X \subseteq V(G)$ mit*

$$\frac{u(\delta(X))}{|X||V(G) \setminus X|} \leq 36W \log n,$$

in polynomieller Zeit bestimmen, wobei (wie früher) $W := \sum_{e \in E(G)} u(e)z(e)$.

Beweis: Zunächst wenden wir Lemma 19.11 mit $d = \frac{1}{n^2}$ an. Wir bekommen eine Partitionierung $V(G) = V_1 \dot{\cup} \dots \dot{\cup} V_k$.

Hat keine der Mengen V_1, \dots, V_k mehr als $\frac{2n}{3}$ Knoten, so vereinen wir schrittweise die zwei kleinsten Mengen bis nur zwei übrig sind. Es seien X und $V(G) \setminus X$ die so resultierenden Mengen. Dann haben wir $|X||V(G) \setminus X| \geq \frac{n}{3} \cdot \frac{2n}{3}$ und $u(\delta(X)) \leq u(\delta(V_1, \dots, V_k)) \leq 8W \log n/d = 8Wn^2 \log n$, und somit $\frac{u(\delta(X))}{|X||V(G) \setminus X|} \leq 36W \log n$, wie gewünscht.

Andernfalls hat eine der Mengen, etwa V_1 , mehr als $\frac{2n}{3}$ Knoten und Durchmesser höchstens $\frac{1}{n^2}$.

Sei B_ξ die Menge derjenigen Knoten, deren Distanz von V_1 höchstens ξ beträgt. Dann gilt

$$\begin{aligned} \int_{\xi \geq 0} |B_\xi| |V(G) \setminus B_\xi| d\xi &\geq |V_1| \int_{\xi \geq 0} |V(G) \setminus B_\xi| d\xi \\ &= |V_1| \sum_{v \in V(G)} \text{dist}_{(G,z)}(v, V_1) \\ &= \frac{|V_1|}{n-1} \sum_{\{v, v'\} \in \binom{V(G)}{2}} (\text{dist}_{(G,z)}(v, V_1) + \text{dist}_{(G,z)}(v', V_1)) \\ &\geq \frac{|V_1|}{n-1} \sum_{\{v, v'\} \in \binom{V(G)}{2}} (\text{dist}_{(G,z)}(v, v') - d) \\ &= \frac{|V_1|}{n-1} \left(1 - \binom{n}{2}d\right) \\ &= \frac{n+1}{2n(n-1)} |V_1| \\ &> \frac{1}{3}. \end{aligned}$$

Damit erhalten wir

$$\int_{\xi \geq 0} u(\delta_G(B_\xi)) d\xi \leq \sum_{e \in E(G)} u(e)z(e) = W < 3W \int_{\xi \geq 0} |B_\xi| |V(G) \setminus B_\xi| d\xi.$$

Somit gibt es ein $\xi \geq 0$ mit $B_\xi \subset V(G)$ und $\frac{u(\delta_G(B_\xi))}{|B_\xi| |V(G) \setminus B_\xi|} \leq 3W$. \square

Hiermit folgt nun:

Satz 19.13. (Leighton und Rao [1999]) *Es gibt einen $O(\log n)$ -Approximationsalgorithmus für das SPARSEST-CUT-PROBLEM, wobei $n = |V(G)|$. Der Max-Flow-Min-Cut-Quotient für das UNIFORME MULTICOMMODITY-FLOW-PROBLEM ist $O(\log n)$.*

Beweis: Gegeben sei eine Instanz (G, u) . Mit Lemma 19.10 erhalten wir eine LP-Lösung (z, w) . O. B. d. A. gilt $w(f) = \text{dist}_{(G,z)}(s, t)$ für alle $f = \{t, s\} \in E(H)$. Nun wenden wir Satz 19.12 auf (G, u) und z an. Da W der LP-Wert ist und somit eine untere Schranke für den Optimalwert darstellt, folgen beide Aussagen. \square

Dieses Resultat wurde von Linial, London und Rabinovich [1995] und Aumann und Rabani [1998] auf beliebige Instanzen des UNGERICHTETEN MULTICOMMODITY-FLOW-PROBLEMS verallgemeinert; im Allgemeinen ist der Max-Flow-Min-Cut-Quotient $O(\log |E(H)|)$.

Die Approximationsszahl in der ersten Aussage ist von Arora, Rao und Vazirani [2009] mittels Relaxierung zu einem semidefiniten Programm auf $O(\sqrt{\log n})$ verbessert worden. Schnellere Algorithmen mit derselben Approximationsgüte sind von Arora, Hazan und Kale [2004] und Sherman [2009] vorgeschlagen worden.

Sparsest-Cut-Algorithmen können beim Entwurf von „divide and conquer“ Approximationsalgorithmen für viele Optimierungsprobleme auf Graphen eingesetzt werden. Ein Beispiel wird in Aufgabe 8 erläutert. Siehe auch Leighton und Rao [1999] und Shmoys [1996], wo viele weitere Beispiele und Verweise angegeben werden.

19.5 Das gerichtete Kantendisjunkte-Wege-Problem

Zunächst bemerken wir, dass dieses Problem bereits in einer recht eingeschränkten Version *NP*-schwer ist:

Satz 19.14. (Even, Itai und Shamir [1976]) *Das GERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM ist sogar dann *NP*-schwer, wenn G azyklisch ist und H nur aus zwei Mengen paralleler Kanten besteht.*

Beweis: Wir werden SATISFIABILITY polynomiell in unser Problem transformieren. Gegeben sei eine Familie $\mathcal{Z} = \{Z_1, \dots, Z_m\}$ von Klauseln über $X =$

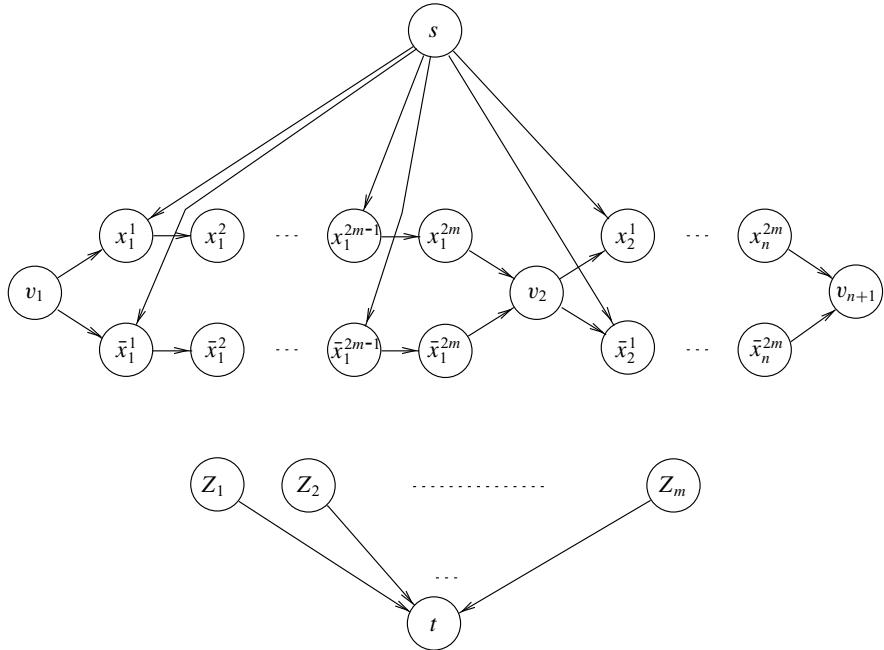


Abbildung 19.3.

$\{x_1, \dots, x_n\}$. Wir konstruieren nun eine Instanz (G, H) des GERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, so dass G azyklisch ist, H nur aus zwei Mengen paralleler Kanten besteht und (G, H) genau dann eine Lösung hat, wenn \mathcal{Z} erfüllbar ist.

Der Digraph G enthält $2m$ Knoten $\lambda^1, \dots, \lambda^{2m}$ für jedes Literal λ und die zusätzlichen Knoten s und t , v_1, \dots, v_{n+1} und Z_1, \dots, Z_m . Es gibt die Kanten (v_i, x_i^1) , (v_i, \bar{x}_i^1) , (x_i^{2m}, v_{i+1}) , $(\bar{x}_i^{2m}, v_{i+1})$, (x_i^j, x_i^{j+1}) und $(\bar{x}_i^j, \bar{x}_i^{j+1})$ für $i = 1, \dots, n$ und $j = 1, \dots, 2m-1$. Ferner gibt es die Kanten (s, x_i^{2j-1}) und (s, \bar{x}_i^{2j-1}) für $i = 1, \dots, n$ und $j = 1, \dots, m$ und auch die Kanten (Z_j, t) und (λ^{2j}, Z_j) für $j = 1, \dots, m$ und alle Literale λ der Klausel Z_j . Ein Beispiel ist in Abb. 19.3 gegeben.

Es bestehe H aus einer Kante (v_{n+1}, v_1) und m parallelen Kanten (t, s) . Wir zeigen nun, dass jede Lösung von (G, H) einer Wahrheitsbelegung entspricht, die alle Klauseln erfüllt, und umgekehrt. Der v_1-v_{n+1} -Weg verläuft nämlich für jedes i entweder durch alle x_i^j (d. h. x_i ist false), oder durch alle \bar{x}_i^j (d. h. x_i ist true). Durch jedes Z_j verläuft ein $s-t$ -Weg. Dies ist aber genau dann möglich, wenn Z_j von der oben definierten Wahrheitsbelegung erfüllt wird. \square

Fortune, Hopcroft und Wyllie [1980] haben gezeigt, dass das GERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM in polynomieller Zeit gelöst werden kann, wenn G azyklisch ist und $|E(H)| = k$ für ein festes k . Für den Fall, dass G nicht

azyklisch ist, haben sie gezeigt, dass das Problem sogar schon dann *NP*-schwer ist, wenn $|E(H)| = 2$. Andererseits haben wir das positive Resultat:

Satz 19.15. (Nash-Williams [1969]) *Sei (G, H) eine Instanz des GERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, wobei $G + H$ eulersch ist und H nur aus zwei Mengen paralleler Kanten besteht. Dann hat (G, H) eine Lösung genau dann, wenn das Schnittkriterium erfüllt ist.*

Beweis: Zunächst bestimmen wir mittels des Satzes von Menger (Satz 8.9) eine Menge von Wegen, die die erste Menge von parallelen Kanten in H realisiert. Nach dem Entfernen dieser Wege (und der entsprechenden Nachfragekanten) erfüllt die restliche Instanz die Voraussetzungen von Proposition 8.12 und hat somit eine Lösung. \square

Mittels Lemma 19.6 kann man dies auf den Fall erweitern, dass es eine Partition von $E(H)$ in zwei Mengen gibt, wobei jede dieser beiden Mengen nur einen Knoten mit nichtverschwindendem Eingangsgrad oder nur einen Knoten mit nichtverschwindendem Ausgangsgrad hat. Ist $G + H$ eulersch und $|E(H)| = 3$, so gibt es auch einen polynomiellen Algorithmus (Ibaraki und Poljak [1991]). Andererseits haben wir folgendes negative Resultat:

Satz 19.16. (Vygen [1995]) *Das GERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM ist sogar dann *NP*-schwer, wenn G azyklisch und $G + H$ eulersch ist und H nur aus drei Mengen paralleler Kanten besteht.*

Beweis: Wir reduzieren das Problem in Satz 19.14 auf dieses. Dazu sei (G, H) eine Instanz des GERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, wobei G azyklisch ist und H nur aus zwei Mengen paralleler Kanten besteht.

Für jedes $v \in V(G)$ definieren wir

$$\begin{aligned}\alpha(v) &:= \max(0, |\delta_{G+H}^+(v)| - |\delta_{G+H}^-(v)|) \text{ und} \\ \beta(v) &:= \max(0, |\delta_{G+H}^-(v)| - |\delta_{G+H}^+(v)|).\end{aligned}$$

Es gilt

$$\sum_{v \in V(G)} (\alpha(v) - \beta(v)) = \sum_{v \in V(G)} (|\delta_{G+H}^+(v)| - |\delta_{G+H}^-(v)|) = 0,$$

und daraus folgt

$$\sum_{v \in V(G)} \alpha(v) = \sum_{v \in V(G)} \beta(v) =: q.$$

Nun konstruieren wir eine Instanz (G', H') des GERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS. Es gehe G' aus G hervor durch Hinzufügen zweier Knoten s und t , sowie $\alpha(v)$ paralleler Kanten (s, v) und $\beta(v)$ paralleler Kanten (v, t) für jeden Knoten v . Es bestehe H' aus allen Kanten von H , sowie q parallelen Kanten (t, s) .

Diese Konstruktion kann offensichtlich in polynomieller Zeit bewerkstelligt werden. Insbesondere wird die Anzahl der Kanten von $G + H$ höchstens vervierfacht. Ferner ist G' azyklisch, $G' + H'$ eulersch und H' besteht nur aus drei Mengen paralleler Kanten. Somit bleibt nur noch zu zeigen, dass (G, H) genau dann eine Lösung hat, wenn (G', H') eine hat.

Jede Lösung von (G', H') ergibt eine Lösung von (G, H) , indem man einfach die $s-t$ -Wege weglässt. Sei also \mathcal{P} eine Lösung von (G, H) und G'' der aus G' durch Entfernen aller von \mathcal{P} benutzten Kanten hervorgehende Graph. Sei H'' der nur aus den q Kanten von t nach s bestehende Teilgraph von H' . Es erfüllt (G'', H'') die Voraussetzungen von Proposition 8.12 und hat somit eine Lösung. Kombinieren wir \mathcal{P} mit einer Lösung von (G'', H'') , so erhalten wir eine Lösung von (G', H') . \square

Da eine Lösung einer Instanz des GERICHTETEN KANTENDISJUNKTE-WEGEPROBLEMS aus paarweise kantendisjunkten Kreisen besteht, liegt es nahe zu fragen, wie viele paarweise kantendisjunkte Kreise ein Digraph hat. Wenigstens für planare Digraphen haben wir eine gute Charakterisierung: Wir betrachten den planaren dualen Graphen und fragen nach der maximalen Anzahl paarweise kantendisjunkter gerichteter Schnitte. Dazu haben wir den folgenden bekannten Min-Max-Satz, dessen Beweis dem von Satz 6.14 sehr ähnlich ist:

Satz 19.17. (Lucchesi und Younger [1978]) *Sei G ein zusammenhängender aber nicht stark zusammenhängender Digraph. Dann ist die maximale Anzahl von paarweise kantendisjunkten gerichteten Schnitten in G gleich der minimalen Kardinalität einer Kantenmenge, die mindestens ein Element aus jedem gerichteten Schnitt enthält.*

Beweis: Sei A die Matrix, deren Spaltenindizes den Kanten entsprechen und deren Zeilen die Inzidenzvektoren der gerichteten Schnitte sind. Betrachte das LP

$$\min\{\mathbb{1}x : Ax \geq \mathbb{1}, x \geq 0\}$$

und das dazu duale LP

$$\max\{\mathbb{1}y : yA \leq \mathbb{1}, y \geq 0\}.$$

Wir müssen beweisen, dass sowohl das primale als auch das duale LP eine ganzzahlige Lösung hat. Nach Korollar 5.16 genügt es zu zeigen, dass das System $Ax \geq \mathbb{1}, x \geq 0$ TDI ist. Dazu verwenden wir Lemma 5.24.

Sei $c : E(G) \rightarrow \mathbb{Z}_+$ und y eine optimale Lösung von $\max\{\mathbb{1}y : yA \leq c, y \geq 0\}$, für die

$$\sum_X y_{\delta^+(X)} |X|^2 \tag{19.9}$$

größtmöglich ist, wobei die Summe über alle Zeilen von A läuft. Wir behaupten, dass das Mengensystem $(V(G), \mathcal{F})$ mit $\mathcal{F} := \{X : y_{\delta^+(X)} > 0\}$ kreuzungsfrei ist. Dazu seien $X, Y \in \mathcal{F}$ mit $X \cap Y \neq \emptyset, X \setminus Y \neq \emptyset, Y \setminus X \neq \emptyset$ und $X \cup Y \neq V(G)$.

Dann sind $\delta^+(X \cap Y)$ und $\delta^+(X \cup Y)$ nach Lemma 2.1(b) auch gerichtete Schnitte. Sei $\epsilon := \min\{y_{\delta^+(X)}, y_{\delta^+(Y)}\}$. Setze $y'_{\delta^+(X)} := y_{\delta^+(X)} - \epsilon$, $y'_{\delta^+(Y)} := y_{\delta^+(Y)} - \epsilon$, $y'_{\delta^+(X \cap Y)} := y_{\delta^+(X \cap Y)} + \epsilon$, $y'_{\delta^+(X \cup Y)} := y_{\delta^+(X \cup Y)} + \epsilon$ und $y'_S := y_S$ für alle weiteren gerichteten Schnitte S . Da y' eine zulässige duale Lösung ist, ist sie auch optimal. Dies widerspricht aber der Wahl von y , da (19.9) für y' größer ist.

Nun sei A' die Untermatrix von A , deren Zeilen den Elementen von \mathcal{F} entsprechen. Es ist A' die Zweiweg-Schnitt-Inzidenzmatrix einer kreuzungsfreien Familie. Mit Satz 5.29 folgt sodann, dass A' vollständig-unimodular ist, wie erwünscht. \square

Ein kombinatorischer Beweis wurde von Lovász [1976] gegeben und ein algorithmischer von Frank [1981].

Beachte, dass die Kantenmengen, die alle gerichteten Schnitte schneiden, genau diejenigen Kantenmengen sind, deren Kontraktion den Digraphen stark zusammenhängend macht. Im planaren dualen Digraphen entsprechen diese Mengen denjenigen Kantenmengen, die mit allen gerichteten Kreisen nichtleeren Durchschnitt haben. Solche Mengen heißen **Feedback-Kantenmengen** und die minimale Kardinalität einer Feedback-Kantenmenge ist die **Feedback-Zahl** des Graphen. Das Problem der Bestimmung der Feedback-Zahl ist i. A. NP-schwer (Karp [1972]) und wahrscheinlich auch schwer zu approximieren (Guruswami et al. [2011]), ist aber polynomiell lösbar für planare Graphen.

Korollar 19.18. *In einem planaren Digraphen ist die maximale Anzahl der paarweise kantendisjunkten Kreise gleich der minimalen Kardinalität einer Kantenmenge, die jeden Kreis schneidet.*

Beweis: Sei G ein Digraph. O. B. d. A. können wir annehmen, dass G zusammenhängend ist und keinen Artikulationsknoten enthält. Betrachte den planaren dualen Graphen von G und Korollar 2.44 und wende den Satz von Lucchesi und Younger (Satz 19.17) an. \square

Ein polynomieller Algorithmus zur Bestimmung der Feedback-Zahl für planare Graphen kann durch Zusammenfügung des Planaritäts-Algorithmus (Satz 2.40), des GRÖTSCHEL-LOVÁSZ-SCHRIJVER-ALGORITHMUS (Satz 4.21) und eines Algorithmus für das MAXIMUM-FLOW-PROBLEM zur Lösung des SEPARATIONS-PROBLEMS (Aufgabe 10) gebildet werden. Das folgende Korollar beinhaltet eine Anwendung auf das KANTENDISJUNKTE-WEGE-PROBLEM:

Korollar 19.19. *Sei (G, H) eine Instanz des GERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, wobei G azyklisch und $G + H$ planar ist. Es hat (G, H) genau dann eine Lösung, wenn $G + H$ durch das Entfernen von $|E(H)| - 1$ beliebigen Kanten von $G + H$ nicht azyklisch wird.* \square

Insbesondere ist das Distanzkriterium hier eine notwendige und hinreichende Bedingung und das Problem kann in polynomieller Zeit gelöst werden.

19.6 Das ungerichtete Kantendisjunkte-Wege-Problem

Das folgende Lemma zeigt eine Verbindung zwischen dem gerichteten und dem ungerichteten Problem auf.

Lemma 19.20. Sei (G, H) eine Instanz des GERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, wobei G azyklisch und $G + H$ eulersch ist. Betrachte die Instanz (G', H') des UNGERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, welche aus (G, H) durch Vernachlässigung der Kantenorientierungen entsteht. Dann ist jede Lösung von (G', H') auch eine Lösung von (G, H) und umgekehrt.

Beweis: Jede Lösung von (G, H) ist trivialerweise auch eine Lösung von (G', H') . Die umgekehrte Aussage beweisen wir mittels Induktion über $|E(G)|$. Hat G keine Kanten, so sind wir fertig.

Nun sei \mathcal{P} eine Lösung von (G', H') . Da G azyklisch ist, enthält G einen Knoten v mit $\delta_G^-(v) = \emptyset$ und $\delta_G^+(v) \neq \emptyset$. Da $G + H$ eulersch ist, gilt $|\delta_H^-(v)| = |\delta_G^+(v)| + |\delta_H^+(v)|$.

Für jede mit v inzidente Nachfragekante gibt es einen von v ausgehenden ungerichteten Weg in \mathcal{P} . Somit ist $|\delta_G^+(v)| \geq |\delta_H^-(v)| + |\delta_H^+(v)|$. Daraus folgt $|\delta_H^+(v)| = 0$ und $|\delta_G^+(v)| = |\delta_H^-(v)|$. Demnach benutzt \mathcal{P} jede mit v inzidente Kante mit der richtigen Orientierung.

Nun sei G_1 der aus G durch Entfernen der mit v inzidenten Kanten hervorgehende Graph. Ferner gehe H_1 aus H dadurch hervor, dass wir jede mit v inzidente Kante $f = (t, v)$ durch (t, w) ersetzen und sie entfernen falls $t = w$, wobei w der erste innere Knoten des f realisierenden Weges in \mathcal{P} ist.

Offensichtlich ist G_1 azyklisch und $G_1 + H_1$ eulersch. Es gehe \mathcal{P}_1 aus \mathcal{P} durch Entfernen aller mit v inzidenten Kanten hervor. Es ist \mathcal{P}_1 eine Lösung von (G'_1, H'_1) , dem (G_1, H_1) entsprechenden ungerichteten Problem.

Mit der Induktionsvoraussetzung ist \mathcal{P}_1 auch eine Lösung von (G_1, H_1) . Durch das Hinzufügen der anfänglichen Kanten folgt, dass \mathcal{P} eine Lösung von (G, H) ist. \square

Daraus folgern wir:

Satz 19.21. (Vygen [1995]) Das UNGERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM ist sogar dann NP-schwer, wenn $G + H$ eulersch ist und H nur aus drei Mengen paralleler Kanten besteht.

Beweis: Wir reduzieren das Problem von Satz 19.16 auf den ungerichteten Fall mittels Lemma 19.20. \square

Ein weiterer NP-schwerer Spezialfall des UNGERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS ist der mit $G + H$ planar (Middendorf und Pfeiffer [1993]). Ist jedoch $G + H$ planar und eulersch, so wird das Problem zugänglich:

Satz 19.22. (Seymour [1981]) Sei (G, H) eine Instanz des UNGERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, wobei $G + H$ planar und eulersch ist. Es hat (G, H) genau dann eine Lösung, wenn das Schnittkriterium erfüllt ist.

Beweis: Wir brauchen nur zu zeigen, dass das Schnittkriterium hinreichend ist. Wir können annehmen, dass $G + H$ zusammenhängend ist. Sei D der planare duale Graph von $G + H$ und $F \subseteq E(D)$ die Menge der dualen Kanten, die den

Nachfragekanten entsprechen. Dann folgt aus dem Schnittkriterium, zusammen mit Satz 2.43, dass $|F \cap E(C)| \leq |E(C) \setminus F|$ für jeden Kreis C in D . Mit Proposition 12.8 folgt, dass F ein T -Join minimalen Gewichtes ist, wobei $T := \{x \in V(D) : |F \cap \delta(x)| \text{ ist ungerade}\}$.

Da $G + H$ eulersch ist, ist D nach Korollar 2.45 bipartit. Nach Satz 12.16 gibt es somit $|F|$ paarweise kantendisjunkte T -Schnitte $C_1, \dots, C_{|F|}$. Da nach Proposition 12.15 jeder T -Schnitt F schneidet, enthält jeder T -Schnitt $C_1, \dots, C_{|F|}$ genau eine Kante aus F .

In $G + H$ wiederum, sind die dualen Mengen der T -Schnitte $C_1, \dots, C_{|F|}$ paarweise kantendisjunkte Kreise, jeder mit genau einer Nachfragekante. Dies bedeutet aber, dass wir eine Lösung des KANTENDISJUNKTE-WEGE-PROBLEMS haben. \square

Mit diesem Satz folgt die Existenz eines polynomiellen Algorithmus (Aufgabe 15). In der Tat haben Matsumoto, Nishizeki und Saito [1986] bewiesen, dass das UNGERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM mit $G + H$ planar und eulersch in $O(n^{\frac{5}{2}} \log n)$ -Zeit gelöst werden kann. Sebő [1993] hat gezeigt, dass das UNGERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM für jede feste ganze Zahl k in polynomieller Zeit gelöst werden kann, falls $G + H$ planar ist und H aus nur k Mengen paralleler Kanten besteht. Ein weiteres bekanntes Resultat ist der Satz von Okamura und Seymour:

Satz 19.23. (Okamura und Seymour [1981]) *Sei (G, H) eine Instanz des UNGERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, wobei $G + H$ eulersch und G planar ist und alle Terminalen am äußeren Gebiet liegen. Es hat (G, H) genau dann eine Lösung, wenn das Schnittkriterium erfüllt ist.*

Beweis: Wir werden mittels Induktion über $|V(G)| + |E(G)|$ zeigen, dass das Schnittkriterium hinreichend ist. Ist $|V(G)| \leq 2$, so ist dies klar.

Hier können wir annehmen, dass G 2-fach zusammenhängend ist, da wir sonst die Induktionsvoraussetzung auf die Blöcke von G anwenden können (dabei müssen Nachfragekanten, die verschiedene Blöcke verbinden, an Artikulationsknoten aufgespalten werden). Wir setzen eine feste planare Einbettung von G voraus; nach Proposition 2.31 ist das äußere Gebiet durch einen Kreis C berandet.

Es heiße eine Menge $X \subset V(G)$ **kritisch**, falls $\emptyset \neq X \cap V(C) \neq V(C)$ und $|\delta_G(X)| = |\delta_H(X)|$.

Gibt es keine kritischen Mengen, so erfüllt die Instanz $(G - e, H + e)$ für jede Kante $e \in E(C)$ das Schnittkriterium: Es ist $|\delta_G(X)| - |\delta_H(X)|$ gerade für alle $X \subseteq V(G)$ (da $G + H$ eulersch ist). Mit der Induktionsvoraussetzung hat $(G - e, H + e)$ eine Lösung und daraus folgt sofort eine Lösung für (G, H) .

Somit nehmen wir an, dass es eine kritische Menge gibt. Wir brauchen das folgende Resultat:

Behauptung: Sei G' ein zusammenhängender Teilgraph von G und H' ein beliebiger Graph mit der Eigenschaft, dass alle Endknoten seiner Kanten in $V(C)$ liegen. Setze $k := \min\{|\delta_{G'}(Y)| - |\delta_{H'}(Y)| : \emptyset \neq Y \subset V(G)\} \in \{-2, 0\}$. Dann

gibt es eine Menge $X \subset V(G)$ für die $C[X]$ ein Weg und $|\delta_{G'}(X)| - |\delta_{H'}(X)| = k$ ist.

Zum Beweis der Behauptung sei $\emptyset \neq X \subset V(G)$ mit $|\delta_{G'}(X)| - |\delta_{H'}(X)| = k$, so dass die gesamte Anzahl der Zusammenhangskomponenten von $G'[X]$ und $G'[V(G) \setminus X]$ minimal ist. Zunächst zeigen wir, dass dann $G'[X]$ und $G'[V(G) \setminus X]$ beide zusammenhängend sind.

Angenommen, dies wäre nicht der Fall, es sei etwa $G'[X]$ unzusammenhängend und habe die Zusammenhangskomponenten X_1, \dots, X_l (die andere Möglichkeit ist symmetrisch). Dann gilt $k = |\delta_{G'}(X)| - |\delta_{H'}(X)| \geq \sum_{i=1}^l (|\delta_{G'}(X_i)| - |\delta_{H'}(X_i)|)$ und damit folgt, dass $|\delta_G(X_i)| - |\delta_H(X_i)| = k$ für irgendwelche $i \in \{1, \dots, l\}$. Ersetzen wir nun X durch X_i , so wird die Anzahl der Zusammenhangskomponenten von $G'[X]$ vermindert, ohne dass die Anzahl der Zusammenhangskomponenten von $G'[V(G) \setminus X]$ erhöht wird. Dies widerspricht jedoch der Wahl von X .

Somit sind $G'[X]$ und $G'[V(G) \setminus X]$ zusammenhängend, und damit auch $G[X]$ und $G[V(G) \setminus X]$. Beachte ferner, dass $\emptyset \neq X \cap V(C) \neq V(C)$, da $|\delta_{H'}(X)| = |\delta_{G'}(X)| - k \geq |\delta_{G'}(X)| > 0$. Da G planar ist, ist $C[X]$ ein Weg. Damit ist die Behauptung bewiesen.

Nun wenden wir die Behauptung auf G und H an. Sei X eine kritische Menge mit der Eigenschaft, dass $C[X]$ ein Weg minimaler Länge ist. Es sei v_1, \dots, v_l eine zyklische Nummerierung der Knoten von C , wobei $V(C) \cap X = \{v_1, \dots, v_j\}$. Setze $e := \{v_l, v_1\}$.

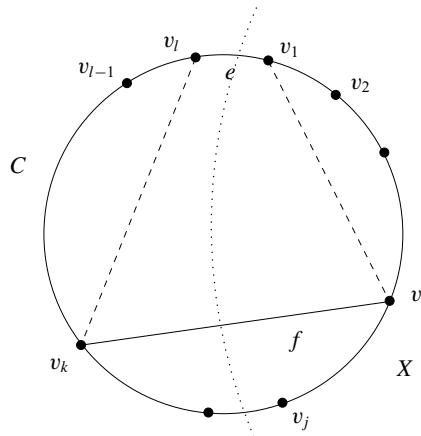


Abbildung 19.4.

Wähle $f = \{v_i, v_k\} \in E(H)$, so dass $1 \leq i \leq j < k \leq l$ (d. h. $v_i \in X$ und $v_k \notin X$) und k größtmöglich ist (siehe Abb. 19.4). Nun betrachten wir $G' := G - e$ und $H' := (V(H), (E(H) \setminus \{f\}) \cup \{\{v_i, v_1\}, \{v_l, v_k\}\})$. (Die Fälle $i = 1$ oder $k = l$ sind nicht ausgeschlossen; hier sollen aber keine Schleifen hinzugefügt werden.)

Wir behaupten, dass (G', H') das Schnittkriterium erfüllt. Dann folgt mittels Induktion, dass (G', H') eine Lösung hat, und diese kann leicht in eine Lösung von (G, H) umgeformt werden.

Angenommen, es erfülle (G', H') nicht das Schnittkriterium, d.h. $|\delta_{G'}(Y)| < |\delta_{H'}(Y)|$ für ein $Y \subseteq V(G)$. Wegen der Behauptung können wir annehmen, dass $C[Y]$ ein Weg ist. Durch eventuellen Austausch von Y und $V(G) \setminus Y$ können wir ferner annehmen, dass $v_i \notin Y$. Da $|\delta_{H'}(Y)| - |\delta_{G'}(Y)| > 0 = |\delta_H(Y)| - |\delta_G(Y)|$, gibt es drei Fälle:

- (a) $v_1 \in Y, v_i, v_k, v_l \notin Y$;
- (b) $v_1, v_l \in Y, v_i, v_k \notin Y$;
- (c) $v_l \in Y, v_1, v_i, v_k \notin Y$.

In jedem dieser drei Fälle haben wir $Y \cap V(C) \subseteq \{v_{k+1}, \dots, v_{i-1}\}$, demnach folgt mit der Wahl von f , dass $E_H(X, Y) = \emptyset$. Ferner gilt $|\delta_G(Y)| = |\delta_H(Y)|$. Mit zweimaliger Anwendung von Lemma 2.1(c) folgt

$$\begin{aligned} |\delta_H(X)| + |\delta_H(Y)| &= |\delta_G(X)| + |\delta_G(Y)| \\ &= |\delta_G(X \cap Y)| + |\delta_G(X \cup Y)| + 2|E_G(X, Y)| \\ &\geq |\delta_H(X \cap Y)| + |\delta_H(X \cup Y)| + 2|E_G(X, Y)| \\ &= |\delta_H(X)| + |\delta_H(Y)| - 2|E_H(X, Y)| + 2|E_G(X, Y)| \\ &= |\delta_H(X)| + |\delta_H(Y)| + 2|E_G(X, Y)| \\ &\geq |\delta_H(X)| + |\delta_H(Y)|. \end{aligned}$$

Somit gilt Gleichheit durchweg. Daraus folgt $|\delta_G(X \cap Y)| = |\delta_H(X \cap Y)|$ und $E_G(X, Y) = \emptyset$.

Also ist Fall (c) unmöglich (da hier $e \in E_G(X, Y)$); d.h. $v_1 \in Y$. Demnach ist $X \cap Y$ nicht leer und $C[X \cap Y]$ ist ein kürzerer Weg als $C[X]$, im Widerspruch zur Wahl von X . \square

Dieser Beweis liefert einen polynomiellen Algorithmus (Aufgabe 16) für das UNGERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM in diesem Spezialfall. Er kann mit $O(n^2)$ -Laufzeit implementiert werden (Becker und Mehlhorn [1986]) und sogar mit linearer Laufzeit (Wagner und Weihe [1995]). Frank [1985] hat sogar einen polynomiellen Algorithmus beschrieben für den Fall, dass die Bedingung ' $G + H$ ist eulersch' durch die schwächere Bedingung ' $|\delta_{G+H}(v)|$ ist gerade für alle Knoten v die nicht am äußeren Gebiet liegen' ersetzt wird. Die Bedingung ' $G + H$ ist eulersch' kann jedoch nicht komplett gestrichen werden:

Satz 19.24. (Schwärzler [2009]) *Das UNGERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM ist sogar NP-schwer wenn G planar ist und alle Terminale am äußeren Gebiet liegen.*

Naves [2009] hat dieses Resultat noch verstärkt indem er annimmt, dass H aus nur zwei Mengen paralleler Kanten besteht. Diese NP-Vollständigkeitsresultate gelten auch für einen planaren azyklischen Digraphen G .

Als Vorbereitung für ein weiteres großes Resultat dieses Abschnitts beweisen wir den folgenden Satz über Orientierungen von gemischten Graphen, d. h. Graphen mit gerichteten und ungerichteten Kanten. Recently, Können wir für einen gegebenen gemischten Graphen G die ungerichteten Kanten so orientieren, dass der resultierende Digraph eulersch ist? Der folgende Satz beantwortet diese Frage.

Satz 19.25. (Ford und Fulkerson [1962]) *Sei G ein ungerichteter Graph und H ein Digraph mit $V(G) = V(H)$. Dann gilt: G hat genau dann eine Orientierung G' , so dass der Digraph $G' + H$ eulersch ist, wenn Folgendes gilt:*

- $|\delta_H^+(v)| + |\delta_H^-(v)| + |\delta_G(v)|$ ist gerade für alle $v \in V(G)$ und
- $|\delta_H^+(X)| - |\delta_H^-(X)| \leq |\delta_G(X)|$ für alle $X \subseteq V(G)$.

Beweis: Die Notwendigkeit der Bedingungen ist klar. Dass sie auch hinreichend sind, beweisen wir mittels Induktion über $|E(G)|$. Die Aussage ist trivial, falls $E(G) = \emptyset$.

Wir nehmen also an, dass G und H die Bedingungen erfüllen und dass $E(G) \neq \emptyset$. Nun wählen wir eine ungerichtete Kante e und orientieren sie so zu einer gerichteten Kante e' , dass $G - e$ und $H + e'$ die Bedingungen erfüllen. Da jeder Grad gerade ist, ist $|\delta_G(X)| - (|\delta_H^+(X)| - |\delta_H^-(X)|)$ für alle $X \subseteq V(G)$ gerade. Diese Zahl verringert sich um 2 falls e' die Menge X verlässt, sonst bleibt sie gleich.

Also werden wir eine Menge X kritisch nennen, falls $|\delta_G(X)| = |\delta_H^+(X)| - |\delta_H^-(X)| > 0$. Gibt es keine kritische Menge, so orientieren wir irgendeine ungerichtete Kante beliebig und wenden Induktion an. Andernfalls sei X eine kritische Menge und $e \in \delta_G(X)$. Wir orientieren e so, dass die resultierende gerichtete Kante e' in einem Knoten von X endet und behaupten, dass die Bedingungen weiterhin gelten.

Hierzu brauchen wir nur die zweite Bedingung zu betrachten. Angenommen, sie gelte nicht, d. h. es gibt ein $Y \subseteq V(G)$ mit $|\delta_{H+e'}^+(Y)| - |\delta_{H+e'}^-(Y)| > |\delta_{G-e}(Y)|$. Dies bedeutet, dass Y vor der Orientierung von e kritisch war und e' verlässt nun Y .

Wenden wir Lemma 2.1(a) bzw. (b) auf $|\delta_H^+|$ bzw. $|\delta_H^-|$ und Lemma 2.1(c) auf $|\delta_G|$ an, so haben wir (vor der Orientierung von e):

$$\begin{aligned} 0 + 0 &= |\delta_G(X)| - |\delta_H^+(X)| + |\delta_H^-(X)| + |\delta_G(Y)| - |\delta_H^+(Y)| + |\delta_H^-(Y)| \\ &= |\delta_G(X \cap Y)| - |\delta_H^+(X \cap Y)| + |\delta_H^-(X \cap Y)| \\ &\quad + |\delta_G(X \cup Y)| - |\delta_H^+(X \cup Y)| + |\delta_H^-(X \cup Y)| + 2|E_G(X, Y)| \\ &\geq 0 + 0 + 2|E_G(X, Y)| \geq 0. \end{aligned}$$

Somit gilt Gleichheit durchweg und daraus folgern wir, dass $E_G(X, Y) = \emptyset$, im Widerspruch zur Existenz von e . \square

Korollar 19.26. *Ein ungerichteter eulerscher Graph kann so orientiert werden, dass ein gerichteter eulerscher Graph resultiert.* \square

Natürlich kann dieses Korollar leichter dadurch bewiesen werden, dass man die Kanten ihrem Auftreten in einem eulerschen Spaziergang entsprechend orientiert.

Wir kehren nun zum KANTENDISJUNKTE-WEGE-PROBLEM zurück.

Satz 19.27. (Rothschild und Whinston [1966]) *Sei (G, H) eine Instanz des UNGERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, wobei $G + H$ eulersch ist und H die Vereinigung von zwei Sternen ist (d.h. es gibt zwei Knoten, so dass jede Nachfragekante mindestens einen von diesen als Endknoten hat). Es hat (G, H) genau dann eine Lösung, wenn das Schnittkriterium erfüllt ist.*

Beweis: Wir zeigen, dass das Schnittkriterium hinreichend ist. Nach Lemma 19.6 können wir annehmen, dass H nur zwei Mengen paralleler Kanten enthält. Beachte, dass die Konstruktion die Eigenschaft ' $G + H$ its eulersch' nicht zerstört.

Nun orientieren wir die Kanten von H beliebig, aber so, dass parallele Kanten dieselbe Orientierung haben. Es sei H' der resultierende Graph. Die beiden Graphen H' und G erfüllen die Voraussetzungen von Satz 19.25, da mit dem Schnittkriterium folgt: $|\delta_{H'}^+(X)| - |\delta_{H'}^-(X)| \leq |\delta_G(X)|$ für alle $X \subseteq V(G)$. Demnach können wir die Kanten von G so orientieren, dass für den resultierenden Digraphen G' gilt: $G' + H'$ ist eulersch.

Wir betrachten (G', H') als eine Instanz des GERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS. Es erfüllt (G', H') das (gerichtete) Schnittkriterium, da für $X \subseteq V(G)$ Folgendes gilt: $2|\delta_{G'}^+(X)| = |\delta_G(X)| + (|\delta_{G'}^+(X)| - |\delta_{G'}^-(X)|) = |\delta_G(X)| + (|\delta_{H'}^-(X)| - |\delta_{H'}^+(X)|) \geq |\delta_H(X)| + (|\delta_{H'}^-(X)| - |\delta_{H'}^+(X)|) = 2|\delta_{H'}^-(X)|$. Nun gewährleistet Satz 19.15 aber eine Lösung, die – bei Vernachlässigung der Orientierungen – eine Lösung von (G, H) ergibt. \square

Hieraus folgt auch die Existenz einer halbganzzahligen Lösung für jede Instanz (G, H) , bei der H die Vereinigung von zwei Sternen ist und das Schnittkriterium erfüllt ist (man verdopple einfach alle Kanten).

Es gilt derselbe Satz, falls H (bei Vernachlässigung paralleler Kanten) K_4 oder C_5 (der Kreis mit Länge 5) ist (Lomonosov [1979], Seymour [1980]). Im Falle K_5 ist das Distanzkriterium hinreichend (Karzanov [1987]). Siehe auch Hirai [2010]. Hat H jedoch drei Mengen paralleler Kanten mit paarweise verschiedenen Endknoten, so ist das Problem NP-schwer, wie wir in Satz 19.21 gesehen haben.

Andererseits haben Robertson und Seymour einen polynomiellen Algorithmus für eine feste Anzahl von Nachfragekanten gefunden:

Satz 19.28. (Robertson und Seymour [1995]) *Für festes k gibt es polynomielle Algorithmen für das auf Instanzen mit $|E(H)| \leq k$ beschränkte UNGERICHTETE KNOTENDISJUNKTE-WEGE-PROBLEM und UNGERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM.*

Beachte, dass das UNGERICHTETE KNOTENDISJUNKTE-WEGE-PROBLEM auch NP-schwer ist; siehe Aufgabe 27. Satz 19.28 ist eines der Resultate aus den bedeutenden Arbeiten von Robertson und Seymour über Minoren von Graphen, die weit über den Rahmen dieses Buches hinausgehen. Der obige Satz wurde

für den knotendisjunkten Fall bewiesen: Robertson und Seymour haben gezeigt, dass es entweder einen nicht relevanten Knoten gibt (dessen Entfernen die Lösbarkeit nicht beeinflusst), oder dass der Graph eine Baumzerlegung mit geringer Baumweite hat (in diesem Fall gibt es einen einfachen polynomiellen Algorithmus; siehe Aufgabe 26). Dies ist ein sehr tiefes Resultat; aber ein wesentlich kürzerer Beweis stammt von Kawarabayashi und Wollan [2010]. Der kantendisjunkte Fall folgt dann leicht; siehe Aufgabe 27. Obwohl die Laufzeit $O(n^2m)$ ist, wächst die von k abhängige Konstante extrem schnell und ist bereits für $k = 3$ in der Praxis unbrauchbar. Vor Kurzem haben Kawarabayashi, Kobayashi und Reed [2012] die Laufzeit auf $O(n^2)$ verbessert.

Aufgaben

1. Sei (G, H) eine gerichtete oder ungerichtete Instanz des KANTENDISJUNKTE-WEGE-PROBLEMS, die das Distanzkriterium (19.2) für ein $z : E(G) \rightarrow \mathbb{R}_+$ verletzt. Man beweise, dass es dann auch ein $z : E(G) \rightarrow \mathbb{Z}_+$ gibt, für welches (19.2) verletzt wird. Ferner finde man Beispiele von Instanzen, die (19.2) für kein $z : E(G) \rightarrow \{0, 1\}$ verletzen.
2. Man zeige, dass das folgende Problem *NP*-schwer ist: Man entscheide ob eine gegebene Instanz (G, H) des UNGERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS das Schnittkriterium erfüllt.
Hinweis: Man gebe eine polynomielle Reduktion von dem folgenden Problem an: Für einen gegebenen ungerichteten Graphen G' und eine Menge $X \subseteq V(G')$ bestimme man eine Menge $Y \subseteq V(G')$ mit $|\delta_{G'}(Y)| > |\delta_{G'}(X)|$ oder entscheide, dass es eine solche nicht gibt. Nach Satz 16.6 ist dieses Problem *NP*-schwer. Man setze $E(H) := \delta_{G'}(X)$ und $E(G) := E(G') \setminus \delta(X)$.
Bemerkung: Dieses Problem ist offensichtlich in *coNP*, aber es ist unbekannt ob es *coNP*-vollständig ist.
 (Sebő [unveröffentlicht])
3. Sei $k \in \mathbb{N}$ gegeben. Man zeige für eine gegebene Instanz (G, H, u, b) des (GERICHTETEN oder UNGERICHTETEN) MULTICOMMODITY-FLOW-PROBLEMS
 - (a) mit höchstens k Terminalen,
 - (b) wobei H eine Knotenüberdeckung der Größe k hat,
 dass man in polynomieller Laufzeit entscheiden kann, ob das Schnittkriterium erfüllt wird.
- * 4. Man betrachte für eine gegebene Instanz (G, H) des UNGERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS die Mehrgüterfluss-Relaxierung und löse

$$\min \{\lambda : \lambda \in \mathbb{R}, y \geq 0, My \leq \lambda \mathbb{1}, Ny = \mathbb{1}\},$$

wobei M und N wie in Lemma 19.1 definiert sind. Sei (y^*, λ^*) eine optimale Lösung. Ziel ist es, eine ganzzahlige Lösung zu finden, d.h. einen $s-t$ -Weg P_f für jede Nachfragekante $f = \{t, s\} \in E(H)$, so dass die maximale Auslastung einer Angebotskante minimiert wird (die Auslastung einer Kante ist

die Anzahl der sie benutzenden Wege). Wir erreichen dies mittels zufälligen Rundens: Für jede Nachfragekante unabhängig wählen wir einen Weg P mit Wahrscheinlichkeit y_P .

Sei $0 < \epsilon \leq 1$. Angenommen, es sei $\lambda^* \geq 3 \ln \frac{|E(G)|}{\epsilon}$. Man beweise, dass das obige zufällige Runden mit Wahrscheinlichkeit mindestens $1 - \epsilon$ eine ganzzahlig Lösung mit maximaler Auslastung höchstens gleich $\lambda^* + \sqrt{3\lambda^* \ln \frac{|E(G)|}{\epsilon}}$ liefert.

Hinweis: Man benutze die folgenden Tatsachen aus der Wahrscheinlichkeitstheorie: Ist $B(m, N, p)$ die Wahrscheinlichkeit von mindestens m Erfolgen in N unabhängigen Bernoulli-Versuchen, jeder mit Erfolgswahrscheinlichkeit p , so gilt

$$B((1 + \beta)Np, N, p) < e^{-\frac{1}{3}\beta^2 Np}$$

für alle $0 < \beta \leq 1$. Ferner ist die Wahrscheinlichkeit von mindestens m Erfolgen in N unabhängigen Bernoulli-Versuchen mit den Erfolgswahrscheinlichkeiten p_1, \dots, p_N höchstens gleich $B\left(m, N, \frac{1}{N}(p_1 + \dots + p_N)\right)$.

(Raghavan und Thompson [1987])

5. Man beschreibe ein Approximationsschema für das CONCURRENT-FLOW-PROBLEM mittels des MULTICOMMODITY-FLOW-APPROXIMATIONSSCHEMAS als Subroutine.

Hinweis: Man benutze binäre Suche um bessere Schranken zur Eingrenzung des Optimalwertes zu finden und um die Instanz so zu modifizieren, dass der mögliche Fluss für jedes Gut proportional zur Nachfrage beschränkt wird.

Bemerkung: Man kann auch das MULTICOMMODITY-FLOW-APPROXIMATIONSSCHEMA so modifizieren, dass man das CONCURRENT-FLOW-PROBLEM direkt lösen kann. Siehe die Literaturangaben in Abschnitt 19.2.

6. Sei G ein Expandergraph wie in Satz 16.42, $n := |V(G)|$, und $u(e) := 1$ für alle $e \in E(G)$. Man zeige, dass der Max-Flow-Min-Cut-Quotient dieser Instanz des UNIFORMEN MULTICOMMODITY-FLOW-PROBLEMS $\Omega(\log n)$ ist.

Hinweis: Man vergleiche die linke mit der rechten Seite in den Ungleichungen, die dem Distanzkriterium und dem Schnittkriterium entsprechen.

7. Sei G ein ungerichteter Graph, $n := V(G)$ und $u : E(G) \rightarrow \mathbb{R}_+$. Für $0 < b \leq \frac{1}{2}$ ist ein b -balancierter Schnitt ein Schnitt $\delta_G(X)$ mit $bn \leq |X| \leq (1 - b)n$. Man bezeichne mit OPT_b die minimale Kapazität eines b -balancierten Schnitts. Nun sei $b' \leq \frac{1}{3}$. Man betrachte den folgenden Algorithmus. Sei $G_0 := G$ und $i := 0$. Finde einen „approximate sparsest“ Schnitt in G_i , und es sei X_i die kleinere der beiden Knotenmengen. Gilt $|V(G_i) \setminus X_i| > (1 - b')n$, so setze $G_{i+1} := G - X_i$, vergrößere i um 1, und iteriere. Sonst gebe $\delta(X_0 \cup \dots \cup X_i)$ als Output an.

- (a) Man zeige, dass $\delta(X_0 \cup \dots \cup X_i)$ ein b' -balancierter Schnitt ist.
(b) Man zeige, dass G_i einen Schnitt $\delta(X)$ hat, für den Folgendes gilt:

$$\frac{u(\delta(X))}{|X||V(G_i) \setminus X|} \leq \frac{2\text{OPT}_b}{(b - b')n^2}$$
 für jedes b mit $b' < b \leq \frac{1}{2}$.

- (c) Man zeige: Wendet man einen $O(\log n)$ -Approximationsalgorithmus für das Sparsest-Cut-Problem als Subroutine an (siehe Satz 19.13), so bekommt man $|\delta(X_0 \cup \dots \cup X_i)| \leq O\left(\frac{\text{OPT}_b \log n}{b-b'}\right)$.

(Leighton und Rao [1999])

Bemerkung: Dies ergibt keinen Approximationsalgorithmus zur Bestimmung eines kapazitätsminimalen b -balancierten Schnittes. Für den Spezialfall $b = \frac{1}{2}$ (der Bisektion heißtt), hat Räcke [2008] einen $O(\log n)$ -Approximationsalgorithmus gefunden.

8. Bei dem OPTIMALEN LINEAR-ARRANGEMENT-PROBLEM geht es um Folgendes: Man nummeriere die n Knoten eines gegebenen ungerichteten Graphen G mit v_1, \dots, v_n , so dass die Gesamtkantenlänge $\sum_{\{v_i, v_j\} \in E(G)} |i - j|$ minimiert wird. Wir wenden Aufgabe 7 an, um einen Approximationsalgorithmus zu bekommen.

- (a) Man zeige, dass die optimalen Gesamtkosten einer Lösung mindestens $(1 - 2b)n \text{OPT}_b$ für beliebiges $0 < b < \frac{1}{2}$ mit $bn \in \mathbb{N}$ betragen.
 (b) Man betrachte den folgenden Algorithmus. Gibt es nur einen Knoten, so ist das Problem trivial. Andernfalls finde man einen $\frac{1}{3}$ -balancierten Schnitt $\delta(X)$ wie in Aufgabe 7, wende den Algorithmus rekursiv auf die beiden durch X und $V(G) \setminus X$ induzierten Teilgraphen an und verkette die beiden linearen Ordnungen. Man zeige, dass die Gesamtkosten der resultierenden Lösung höchstens $O(\log^2 n)$ mal den Optimalwert betragen.

(Hansen [1989])

Bemerkung: Das Problem ist NP -schwer. Der momentan beste Approximationsalgorithmus, mit Approximationsgüte $O(\sqrt{\log n} \log \log n)$, ist der von Charikar et al. [2010] und Feige und Lee [2007].

9. Man beweise, dass es einen polynomiellen Algorithmus für das GERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM mit $G + H$ eulersch und H bestehend aus nur zwei Mengen paralleler Kanten gibt.
10. Man zeige, dass man in einem gegebenen Digraphen eine kardinalitätsminimale alle gerichteten Schnitte schneidende Kantenmenge in polynomieller Zeit finden kann. Man zeige, dass man für planare Graphen die Feedback-Zahl in polynomieller Zeit bestimmen kann.
11. Man zeige, dass man in einem gegebenen Digraphen eine kardinalitätsminimale Menge von Kanten, deren Kontraktion den Digraphen stark zusammenhängend macht, in polynomieller Zeit finden kann.
12. Man beweise, dass die Aussage von Korollar 19.18 nicht für allgemeine (nicht planare) Digraphen gilt.
13. Man zeige, dass die Aussage von Korollar 19.19 ohne die Bedingung „ G ist azyklisch“ falsch ist.
- Bemerkung:* In diesem Fall ist das GERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM NP -schwer (Vygen [1995]).
14. Man betrachte den folgenden Greedy-Algorithmus für das GERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM: Gegeben sei eine Instanz (G, H) . Man wähle eine Kante $f = (t, s) \in E(H)$ mit $\text{dist}_G(s, t)$ minimal. Sei P_f ein

kürzester s - t -Weg in G . Man entferne f aus $E(H)$ und $E(P_f)$ aus $E(G)$. Nun iteriere man bis es keine realisierbare Nachfragekante mehr gibt. Dies liefert eine Lösung für (G, H') mit einem Teilgraphen H' von H .

Sei H^* irgendein Teilgraph von H , für den (G, H^*) eine Lösung besitzt. Man zeige, dass dann $|E(H')| \geq \frac{|E(H^*)|}{\sqrt{m}}$ ist, wobei $m := |E(G)|$.

Hinweis: Man betrachte den Fall, dass keine weitere Nachfragekante mittels eines Weges der Länge höchstens gleich \sqrt{m} realisiert werden kann.

(Kleinberg [1996]; siehe Chekuri und Khanna [2007] für eine verbesserte Analyse.)

15. Man beweise, dass das UNGERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM in polynomieller Zeit gelöst werden kann, falls $G + H$ planar und eulersch ist.
16. Man zeige, dass der Beweis des Satzes von Okamura und Seymour zu einem polynomiellen Algorithmus führt.
17. Sei (G, H) eine Instanz des UNGERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS. Man nehme an, dass G planar ist, dass alle Terminale am äußeren Gebiet liegen und dass jeder nicht am äußeren Gebiet liegende Knoten geraden Grad hat. Man nehme ferner an, dass

$$|\delta_G(X)| > |\delta_H(X)| \quad \text{für alle } \emptyset \neq X \subset V(G).$$

Man beweise, dass es eine Lösung für (G, H) gibt.

Hinweis: Man verwende den Satz von Okamura und Seymour.

18. Man formuliere und beweise als Verallgemeinerung des Satzes von Robbins (Aufgabe 22(c), Kapitel 2) eine notwendige und hinreichende Bedingung für die Existenz einer Orientierung der ungerichteten Kanten eines gemischten Graphen, so dass der resultierende Digraph stark zusammenhängend ist.
(Boesch und Tindell [1980])
19. Sei (G, H) eine Instanz des GERICHTETEN KANTENDISJUNKTE-WEGE-PROBLEMS, wobei $G + H$ eulersch und G planar und azyklisch ist und alle Terminale am äußeren Gebiet liegen. Man beweise, dass (G, H) genau dann eine Lösung hat, wenn das Schnittkriterium erfüllt ist.
Hinweis: Man verwende Lemma 19.20 und den Satz von Okamura und Seymour (Satz 19.23).
20. Man beweise Satz 19.25 mittels Netzwerkflussverfahren (woraus auch die Existenz eines polynomiellen Verfahrens folgt).
21. Man zeige auf zwei verschiedene Arten, dass es möglich ist, in polynomieller Zeit zu entscheiden, ob ein Paar (G, H) die Bedingungen von Satz 19.25 erfüllt: Einmal mittels Minimierung submodularer Funktionen und der Äquivalenz von Separation und Optimierung, und das andere Mal mittels Netzwerkflussverfahren (siehe Aufgabe 20).
22. Man beweise den Orientierungssatz von Nash-Williams [1969], der den Satz von Robbins (Aufgabe 22(c), Kapitel 2) verallgemeinert:
Ein ungerichteter Graph G kann genau dann durch Orientierung zu einem stark k -fach kantenzusammenhängenden (d. h. es gibt k paarweise kantendisjunkte

s - t -Wege für jedes Paar $(s, t) \in V(G) \times V(G)$ gemacht werden, wenn G $2k$ -fach kantenzusammenhängend ist.

Hinweis: Um zu beweisen, dass die Bedingung hinreichend ist, sei G' irgend-eine Orientierung von G . Man beweise, dass das System

$$\begin{aligned} x_e &\leq 1 & (e \in E(G')), \\ x_e &\geq 0 & (e \in E(G')), \\ \sum_{e \in \delta_{G'}^-(X)} x_e - \sum_{e \in \delta_{G'}^+(X)} x_e &\leq |\delta_{G'}^-(X)| - k & (\emptyset \neq X \subset V(G')) \end{aligned}$$

TDI ist, wie im Beweis des Satzes von Lucchesi und Younger (Satz 19.17). (Frank [1980], Frank und Tardos [1984])

23. Man beweise den 2-Commodity-Flow-Satz von Hu: Eine Instanz (G, H, u, b) des UNGERICHTETEN MULTICOMMODITY-FLOW-PROBLEMS mit $|E(H)| = 2$ hat genau dann eine Lösung, wenn $\sum_{e \in \delta_G(X)} u(e) \geq \sum_{f \in \delta_H(X)} b(f)$ für alle $X \subseteq V(G)$ gilt, d. h. wenn das Schnittkriterium erfüllt ist.

Hinweis: Man verwende Satz 19.27.

(Hu [1963])

24. Man beweise, dass es einen polynomiellen Algorithmus für das UNGERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM mit $G + H$ eulersch und H bestehend aus nur zwei Mengen paralleler Kanten gibt.

- * 25. Man betrachte Instanzen (G, H) des UNGERICHTETEN KNOTENDISJUNKTE-WEGE-PROBLEMS, wobei G planar und alle Terminale paarweise verschieden sind (d. h. $e \cap f = \emptyset$ für je zwei Nachfragekanten e und f) und am äußeren Gebiet liegen. Sei (G, H) eine solche Instanz mit G 2-fach zusammenhängend; also wird das äußere Gebiet durch einen Kreis C berandet (siehe Proposition 2.31).

Man beweise, dass (G, H) genau dann eine Lösung hat, wenn die folgenden Bedingungen gelten:

- Es ist $G + H$ planar;
- keine Menge $X \subseteq V(G)$ trennt mehr als $|X|$ Nachfragekanten (man sagt: X trennt $\{v, w\}$, falls $\{v, w\} \cap X \neq \emptyset$ ist oder w nicht von v aus in $G - X$ erreichbar ist).

Man folgere hieraus, dass das UNGERICHTETE KNOTENDISJUNKTE-WEGE-PROBLEM in planaren Graphen mit paarweise verschiedenen Terminalen am äußeren Gebiet in polynomieller Zeit gelöst werden kann.

Hinweis: Um zu zeigen, dass (a) und (b) hinreichend sind, betrachte man den folgenden Induktionsschritt: Sei $f = \{v, w\}$ eine Nachfragekante mit der Eigenschaft, dass mindestens einer der beiden v - w -Wege auf C kein anderes Terminal enthält. Man realisiere f mit diesem Weg und entferne ihn dann.

Bemerkung: Robertson und Seymour [1986] haben dies zu einer notwendigen und hinreichenden Bedingung für die Lösbarkeit des UNGERICHTETEN KNOTENDISJUNKTE-WEGE-PROBLEMS mit zwei Nachfragekanten erweitert.

- * 26. Es sei $k \in \mathbb{N}$ fest. Man beweise, dass es einen polynomiellen Algorithmus für das auf Graphen mit Baumweite höchstens gleich k beschränkte KNOTENDISJUNKTE-WEGE-PROBLEM gibt (siehe Aufgabe 28, Kapitel 2).

- Bemerkung:* Scheffler [1994] hat bewiesen, dass es tatsächlich einen Algorithmus mit linearer Laufzeit gibt. Im Gegensatz dazu ist das UNGERICHTETE KANTENDISJUNKTE-WEGE-PROBLEM sogar für Graphen mit Baumweite 2 NP-schwer (Nishizeki, Vygen und Zhou [2001]).
27. Man beweise, dass sowohl das GERICHTETE als auch das UNGERICHTETE KNOTENDISJUNKTE-WEGE-PROBLEM NP-schwer ist. Man beweise ferner, dass der kantendisjunkte Teil von Satz 19.28 aus dem knotendisjunkten Teil folgt.

Literatur

Allgemeine Literatur:

- Frank, A. [1990]: Packing paths, circuits and cuts – a survey. In: Paths, Flows, and VLSI-Layout (B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, Hrsg.), Springer, Berlin 1990, S. 47–100
- Naves, G., und Sebő, A. [2009]: Multiflow feasibility: an annotated tableau. In: Research Trends in Combinatorial Optimization (W.J. Cook, L. Lovász, J. Vygen, Hrsg.), Springer, Berlin 2009, pp. 261–283
- Rippelhausen-Lipa, H., Wagner, D., und Weihe, K. [1995]: Efficient algorithms for disjoint paths in planar graphs. In: Combinatorial Optimization; DIMACS Series in Discrete Mathematics and Theoretical Computer Science 20 (W.J. Cook, L. Lovász, P. Seymour, Hrsg.), AMS, Providence 1995
- Schrijver, A. [2003]: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin 2003, Kapitel 70–76
- Shmoys, D.B. [1996]: Cut problems and their application to divide-and-conquer. In: Approximation Algorithms for NP-Hard Problems (D.S. Hochbaum, Hrsg.), PWS, Boston, 1996

Zitierte Literatur:

- Aumann, Y. und Rabani, Y. [1998]: An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. SIAM Journal on Computing 27 (1998), 291–301
- Arora, S., Rao, S., und Vazirani, U. [2009]: Expander flows, geometric embeddings and graph partitioning. Journal of the ACM 56 (2009), Article 5
- Arora, S., Hazan, E., und Kale, S. [2004]: $O(\sqrt{\log n})$ approximation to SPARSEST CUT in $\tilde{O}(n^2)$ time. Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (2004), 238–247
- Becker, M., und Mehlhorn, K. [1986]: Algorithms for routing in planar graphs. Acta Informatica 23 (1986), 163–176
- Bienstock, D., und Iyengar, G. [2006]: Solving fractional packing problems in $O^*(\frac{1}{\epsilon})$ iterations. SIAM Journal on Computing 35 (2006), 825–854
- Boesch, F., und Tindell, R. [1980]: Robbins's theorem for mixed multigraphs. American Mathematical Monthly 87 (1980), 716–719
- Charikar, M., Hajiaghayi, M.T., Karloff, H. und Rao, S. [2010]: ℓ_2^2 spreading metrics for vertex ordering problems. Algorithmica 56 (2010), 577–604
- Chekuri, C., und Khanna, S. [2007]: Edge-disjoint paths revisited. ACM Transactions on Algorithms 3 (2007), Article 46

- Chudak, F.A., und Eleutério, V. [2005]: Improved approximation schemes for linear programming relaxations of combinatorial optimization problems. In: Integer Programming and Combinatorial Optimization; Proceedings of the 11th International IPCO Conference; LNCS 3509 (M. Jünger, V. Kaibel, Hrsg.), Springer, Berlin 2005, S. 81–96
- Even, S., Itai, A., und Shamir, A. [1976]: On the complexity of timetable and multicommodity flow problems. SIAM Journal on Computing 5 (1976), 691–703
- Feige, U., und Lee, J.R. [2007]: An improved approximation ratio for the minimum linear arrangement problem. Information Processing Letters 101 (2007), 26–29
- Fleischer, L.K. [2000]: Approximating fractional multicommodity flow independent of the number of commodities. SIAM Journal on Discrete Mathematics 13 (2000), 505–520
- Ford, L.R., und Fulkerson, D.R. [1958]: A suggested computation for maximal multicommodity network flows. Management Science 5 (1958), 97–101
- Ford, L.R., und Fulkerson, D.R. [1962]: Flows in Networks. Princeton University Press, Princeton 1962
- Fortune, S., Hopcroft, J., und Wyllie, J. [1980]: The directed subgraph homeomorphism problem. Theoretical Computer Science 10 (1980), 111–121
- Frank, A. [1980]: On the orientation of graphs. Journal of Combinatorial Theory B 28 (1980), 251–261
- Frank, A. [1981]: How to make a digraph strongly connected. Combinatorica 1 (1981), 145–153
- Frank, A. [1985]: Edge-disjoint paths in planar graphs. Journal of Combinatorial Theory B 39 (1985), 164–178
- Frank, A., und Tardos, É. [1984]: Matroids from crossing families. In: Finite and Infinite Sets; Vol. I (A. Hajnal, L. Lovász, und V.T. Sós, Hrsg.), North-Holland, Amsterdam, 1984, S. 295–304
- Garg, N., und Könemann, J. [2007]: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. SIAM Journal on Computing 37 (2007), 630–652
- Grigoriadis, M.D., und Khachiyan, L.G. [1996]: Coordination complexity of parallel price-directive decomposition. Mathematics of Operations Research 21 (1996), 321–340
- Guruswami, V., Hästad, J., Manokaran, R., Raghavendra, P., und Charikar, M. [2011]: Beating the random ordering is hard: every ordering CSP is approximation resistant. SIAM Journal on Computing 40 (2011), 878–914
- Hansen, M.D. [1989]: Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (1989), 604–609
- Hirai, H. [2010]: Metric packing for $K_3 + K_3$. Combinatorica 30 (2010), 295–326
- Hu, T.C. [1963]: Multi-commodity network flows. Operations Research 11 (1963), 344–360
- Ibaraki, T., und Poljak, S. [1991]: Weak three-linking in Eulerian digraphs. SIAM Journal on Discrete Mathematics 4 (1991), 84–98
- Karakostas, G. [2008]: Faster approximation schemes for fractional multicommodity flow problems. ACM Transactions on Algorithms 4 (2008), Article 13
- Karp, R.M. [1972]: Reducibility among combinatorial problems. In: Complexity of Computer Computations (R.E. Miller, J.W. Thatcher, Hrsg.), Plenum Press, New York 1972, S. 85–103
- Karzanov, A.V. [1987]: Half-integral five-terminus flows. Discrete Applied Mathematics 18 (1987) 263–278
- Kawarabayashi, K., Kobayashi, Y., und Reed, B. [2012]: The disjoint paths problem in quadratic time. Journal of Combinatorial Theory B 102 (2012), 424–435

- Kawarabayashi, K., und Wollan, P. [2010]: A shorter proof of the graph minor algorithm: the unique linkage theorem. Proceedings of the 42th Annual ACM Symposium on Theory of Computing (2010), 687–694
- Kleinberg, J. [1996]: Approximation algorithms for disjoint paths problems. PhD thesis, MIT, Cambridge 1996
- Leighton, T., und Rao, S. [1999]: Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. Journal of the ACM 46 (1999), 787–832
- Linial, N., London, E., und Rabinovich, Y. [1995]: The geometry of graphs and some of its algorithmic applications. Combinatorica 15 (1995), 215–245
- Lomonosov, M. [1979]: Multiflow feasibility depending on cuts. Graph Theory Newsletter 9 (1979), 4
- Lovász, L. [1976]: On two minimax theorems in graph. Journal of Combinatorial Theory B 21 (1976), 96–103
- Lucchesi, C.L., und Younger, D.H. [1978]: A minimax relation for directed graphs. Journal of the London Mathematical Society II 17 (1978), 369–374
- Matsumoto, K., Nishizeki, T., und Saito, N. [1986]: Planar multicommodity flows, maximum matchings and negative cycles. SIAM Journal on Computing 15 (1986), 495–510
- Middendorf, M., und Pfeiffer, F. [1993]: On the complexity of the disjoint path problem. Combinatorica 13 (1993), 97–107
- Müller, D., Radke, K., und Vygen, J. [2011]: Faster min-max resource sharing in theory and practice. Mathematical Programming Computation 3 (2011), 1–35
- Nagamochi, H., und Ibaraki, T. [1989]: On max-flow min-cut and integral flow properties for multicommodity flows in directed networks. Information Processing Letters 31 (1989), 279–285
- Nash-Williams, C.S.J.A. [1969]: Well-balanced orientations of finite graphs and unobtrusive odd-vertex-pairings. In: Recent Progress in Combinatorics (W. Tutte, Hrsg.), Academic Press, New York 1969, S. 133–149
- Naves, G. [2009]: The hardness of routing two pairs on one face. Les cahiers Leibniz, Technical Report No. 177, Grenoble 2009
- Nishizeki, T., Vygen, J., und Zhou, X. [2001]: The edge-disjoint paths problem is NP-complete for series-parallel graphs. Discrete Applied Mathematics 115 (2001), 177–186
- Okamura, H., und Seymour, P.D. [1981]: Multicommodity flows in planar graphs. Journal of Combinatorial Theory B 31 (1981), 75–81
- Räcke, H. [2008]: Optimal hierarchical decompositions for congestion minimization in networks. Proceedings of the 40th Annual ACM Symposium on Theory of Computing (2008), 255–264
- Raghavan, P., und Thompson, C.D. [1987]: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. Combinatorica 7 (1987), 365–374
- Robertson, N., und Seymour, P.D. [1986]: Graph minors VI; Disjoint paths across a disc. Journal of Combinatorial Theory B 41 (1986), 115–138
- Robertson, N., und Seymour, P.D. [1995]: Graph minors XIII; The disjoint paths problem. Journal of Combinatorial Theory B 63 (1995), 65–110
- Rothschild, B., und Whinston, A. [1966]: Feasibility of two-commodity network flows. Operations Research 14 (1966), 1121–1129
- Scheffler, P. [1994]: A practical linear time algorithm for disjoint paths in graphs with bounded tree-width. Technical Report No. 396/1994, FU Berlin, Fachbereich 3 Mathematik
- Schwärzler, W. [2009]: On the complexity of the planar edge-disjoint paths problem with terminals on the outer boundary. Combinatorica 29 (2009), 121–126

- Sebő, A. [1993]: Integer plane multiflows with a fixed number of demands. *Journal of Combinatorial Theory B* 59 (1993), 163–171
- Seymour, P.D. [1980]: Four-terminus flows. *Networks* 10 (1980), 79–86
- Seymour, P.D. [1981]: On odd cuts and multicommodity flows. *Proceedings of the London Mathematical Society* (3) 42 (1981), 178–192
- Shahrokhi, F., und Matula, D.W. [1990]: The maximum concurrent flow problem. *Journal of the ACM* 37 (1990), 318–334
- Sherman, J. [2009]: Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$ -approximations to sparsest cut. *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science* (2009), 363–372
- Vygen, J. [1995]: *NP*-completeness of some edge-disjoint paths problems. *Discrete Applied Mathematics* 61 (1995), 83–90
- Wagner, D., und Weihe, K. [1995]: A linear-time algorithm for edge-disjoint paths in planar graphs. *Combinatorica* 15 (1995), 135–150
- Young, N. [1995]: Randomized rounding without solving the linear program. *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms* (1995), 170–178



20 Netzwerk-Design-Probleme

Der Begriff des Zusammenhangs ist ein sehr wichtiger in der kombinatorischen Optimierung. In Kapitel 8 haben wir gezeigt, wie man den Zusammenhang zwischen jedem Knotenpaar eines ungerichteten Graphen berechnen kann. Hier interessieren wir uns für Teilgraphen, die gewisse Zusammenhangsbedingungen erfüllen. Das allgemeine Problem lautet:

SURVIVABLE-NETWORK-DESIGN-PROBLEM

- Instanz:* Ein ungerichteter Graph G mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ und eine Zusammenhangsbedingung $r_{xy} \in \mathbb{Z}_+$ für jedes (ungeordnete) Knotenpaar x, y .
- Aufgabe:* Bestimme einen aufspannenden Teilgraphen H von G mit minimalem Gewicht, so dass es für je zwei Knoten x, y mindestens r_{xy} paarweise kantendisjunkte Wege von x nach y in H gibt.

Praktische Anwendungen kommen z. B. beim Entwurf von Telekommunikationsnetzwerken vor, die Ausfälle einzelner Kanten „überleben“ sollen.

Bei einem verwandten Problem dürfen Kanten beliebig oft gewählt werden (siehe Goemans und Bertsimas [1993] und Bertsimas und Teo [1997]). Dies kann jedoch als ein Spezialfall des obigen Problems betrachtet werden, da G beliebig viele parallele Kanten haben kann.

Zunächst werden wir einen bekannten Spezialfall betrachten, nämlich das STEINERBAUM-PROBLEM. Dort haben wir eine Menge $T \subseteq V(G)$ sogenannter Terminale mit $r_{xy} = 1$ für $x, y \in T$ und $r_{xy} = 0$ sonst. Ziel ist es, ein kürzestes alle Terminale verbindendes Netzwerk zu finden; ein solches Netzwerk wird Konnektor genannt und ein kreisfreier Konnektor heißt Steinerbaum:

Definition 20.1. Sei G ein ungerichteter Graph und $T \subseteq V(G)$. Ein Konnektor für T ist ein zusammenhängender Graph Y mit $T \subseteq V(Y)$. Ein Steinerbaum für T in G ist ein Baum S mit $T \subseteq V(S) \subseteq V(G)$ und $E(S) \subseteq E(G)$. Die Elemente von T heißen **Terminale** und die von $V(S) \setminus T$ heißen **Steinerpunkte** von S .

Gelegentlich verlangt man auch, dass alle Blätter eines Steinerbaumes Terminalen sind; offensichtlich kann man dies immer durch das Entfernen von Kanten erreichen. In Abschnitt 20.1 besprechen wir einen exakten Algorithmus für das STEINERBAUM-PROBLEM, während die Abschnitte 20.2 und 20.3 den Approximationsalgorithmen gewidmet sind.

In Abschnitt 20.4 wenden wir uns dem allgemeinen SURVIVABLE-NETWORK-DESIGN-PROBLEM zu. In den Abschnitten 20.5 und 20.6 betrachten wir zwei Approximationsalgorithmen. Der erste ist zwar schneller, aber der zweite läuft stets mit Approximationsgüte 2 in polynomieller Zeit.

Schließlich werden wir in Abschnitt 20.7 ein Netzwerk-Design-Problem kennlernen, das in polynomieller Zeit exakt gelöst werden kann, nämlich das sogenannte VPN-Problem.

20.1 Steinerbäume

Wir betrachten hier das folgende Problem:

STEINERBAUM-PROBLEM

Instanz: Ein ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$ und eine Menge $T \subseteq V(G)$.

Aufgabe: Bestimme einen Steinerbaum S für T in G mit minimalem Gewicht $c(E(S))$.

Die Spezialfälle $T = V(G)$ (aufspannender Baum) und $|T| = 2$ (kürzester Weg) haben wir bereits in den Kapiteln 6 und 7 behandelt. Für diese gab es polynomielle Algorithmen, das allgemeine Problem ist jedoch *NP*-schwer:

Satz 20.2. (Karp [1972]) *Das STEINERBAUM-PROBLEM ist sogar dann NP-schwer, wenn alle Kantengewichte gleich eins sind.*

Beweis: Wir werden VERTEX-COVER in unser Problem transformieren; ersteres ist nach Korollar 15.24 *NP*-vollständig. Gegeben sei ein Graph G als Teil einer VERTEX-COVER-Instanz. Wir können annehmen, dass er mehr als eine Kante enthält. Sei nun H der Graph mit den Knoten $V(H) := V(G) \cup E(G)$ und den Kanten $\{v, e\}$ für $v \in e \in E(G)$ und $\{v, w\}$ für $v, w \in V(G), v \neq w$. Ein Beispiel ist in Abb. 20.1 gegeben. Wir setzen $c(e) := 1$ für alle $e \in E(H)$ und $T := E(G)$.

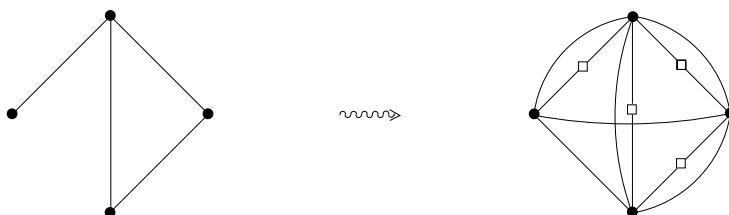


Abbildung 20.1.

Ist $X \subseteq V(G)$ eine Knotenüberdeckung von G , so können wir in H die Knoten von X durch einen Baum mit $|X| - 1$ Kanten verbinden und auch jeden in

T liegenden H -Knoten durch eine H -Kante mit dem Baum verbinden. Dadurch erhalten wir einen Steinerbaum für T mit $|X| - 1 + |E(G)|$ Kanten. Sei andererseits $(T \cup X, F)$ ein Steinerbaum für T in H . Dann ist X eine Knotenüberdeckung von G und $|F| = |T \cup X| - 1 = |X| + |E(G)| - 1$.

Somit hat G genau dann eine Knotenüberdeckung der Kardinalität k , wenn H einen Steinerbaum für T mit $k + |E(G)| - 1$ Kanten hat. \square

Diese Transformation ermöglicht auch das folgende stärkere Resultat:

Satz 20.3. (Bern und Plassmann [1989]) *Das STEINERBAUM-PROBLEM ist sogar dann MAXSNP-schwer, wenn alle Kantengewichte gleich eins sind.*

Beweis: Die im obigen Beweis konstruierte Transformation ist i. A. keine L-Reduktion; wir behaupten aber, dass sie eine ist, falls der Grad von G beschränkt ist. Nach Satz 16.46 ist das MINIMUM-VERTEX-COVER-PROBLEM für Graphen mit Grad höchstens gleich 4 MAXSNP-schwer.

Für jeden Steinerbaum $(T \cup X, F)$ in H und die entsprechende Knotenüberdeckung X von G haben wir

$$\begin{aligned} |X| - \text{OPT}(G) &= (|F| - |E(G)| + 1) - (\text{OPT}(H, T) - |E(G)| + 1) \\ &= |F| - \text{OPT}(H, T). \end{aligned}$$

Ferner gilt $\text{OPT}(H, T) \leq 2|T| - 1 = 2|E(G)| - 1$ und $\text{OPT}(G) \geq \frac{|E(G)|}{4}$, falls der Grad von G höchstens gleich 4 ist. Damit folgt $\text{OPT}(H, T) < 8\text{OPT}(G)$, und demnach ist die Transformation in der Tat eine L-Reduktion. \square

Zwei weitere Varianten des STEINERBAUM-PROBLEMS sind *NP*-schwer: das EUKLIDISCHE STEINERBAUM-PROBLEM (Garey, Graham und Johnson [1977]) und das MANHATTAN-STEINERBAUM-PROBLEM (Garey und Johnson [1977]). In beiden sucht man ein Netzwerk (eine Menge von Abschnitten gerader Linien) mit minimaler Gesamtlänge, welches eine gegebene Punktmenge in der Ebene verbindet. Der einzige Unterschied zwischen den beiden Problemen ist, dass die geradlinigen Verbindungen im MANHATTAN-STEINERBAUM-PROBLEM nur senkrecht und waagerecht verlaufen dürfen. Im Gegensatz zum MAXSNP-schweren STEINERBAUM-PROBLEM in Graphen haben diese beiden geometrischen Versionen ein Approximationsschema. Eine von Arora [1998] stammende Variante dieses Algorithmus löst auch das EUKLIDISCHE TSP und einige weitere geometrische Probleme; wir werden diesen in Abschnitt 21.2 besprechen (siehe Aufgabe 10, Kapitel 21). Das STEINERBAUM-PROBLEM in planaren Graphen hat auch ein Approximationsschema; dieses wurde von Borradaile, Klein und Mathieu [2009] gefunden.

Hanan [1966] hat gezeigt, dass das MANHATTAN-STEINERBAUM-PROBLEM auf das STEINERBAUM-PROBLEM in endlichen Gittergraphen reduziert werden kann: Es gibt immer eine optimale Lösung, bei der alle geradlinigen Verbindungen auf dem von den Terminalen induzierten Koordinatengitter liegen. Das MANHATTAN-STEINERBAUM-PROBLEM spielt eine wichtige Rolle im VLSI-Design (VLSI bedeutet „very large scale integration“), wo elektronische Komponenten durch senkrechte und waagerechte Drähte verbunden werden müssen; siehe

Korte, Prömel und Steger [1990], Martin [1992] und Hetzel [1995]. Hier sucht man viele paarweise disjunkte Steinerbäume; dies ist eine Verallgemeinerung des in Kapitel 19 besprochenen DISJUNKTE-WEGE-PROBLEMS.

Wir werden nun einen von Dreyfus und Wagner [1972] stammenden auf dynamischer Optimierung basierenden Algorithmus beschreiben. Dieser löst das STEINERBAUM-PROBLEM exakt, hat aber i. A. exponentielle Laufzeit.

Der DREYFUS-WAGNER-ALGORITHMUS bestimmt einen optimalen Steinerbaum für jede Teilmenge von T , beginnend mit den zweielementigen Teilmengen. Er benutzt die folgenden Rekursionsformeln:

Lemma 20.4. *Sei (G, c, T) eine Instanz des STEINERBAUM-PROBLEMS. Für jedes $U \subseteq T$ und $x \in V(G) \setminus U$ definieren wir*

$$\begin{aligned} p(U) &:= \min\{c(E(S)) : S \text{ ist ein Steinerbaum für } U \text{ in } G\}; \\ q(U \cup \{x\}, x) &:= \min\{c(E(S')) + c(E(S'')) : \emptyset \neq U' \subset U, \\ &\quad S' \text{ ist ein Steinerbaum für } U' \cup \{x\} \text{ in } G, \\ &\quad S'' \text{ ist ein Steinerbaum für } (U \setminus U') \cup \{x\} \text{ in } G\}. \end{aligned}$$

Dann gilt für alle $U \subseteq V(G)$ mit $|U| \geq 2$ und alle $x \in V(G) \setminus U$:

$$\begin{aligned} (a) \quad q(U \cup \{x\}, x) &= \min_{\emptyset \neq U' \subset U} \left(p(U' \cup \{x\}) + p((U \setminus U') \cup \{x\}) \right), \\ (b) \quad p(U \cup \{x\}) &= \min \left\{ \min_{y \in U} \left(p(U) + \text{dist}_{(G,c)}(x, y) \right), \right. \\ &\quad \left. \min_{y \in V(G) \setminus U} \left(q(U \cup \{y\}, y) + \text{dist}_{(G,c)}(x, y) \right) \right\}. \end{aligned}$$

Beweis: Die Gleichung (a) ist trivial, sowie die Ungleichung “ \leq ” von (b).

Um die Ungleichung “ \geq ” von (b) zu beweisen, betrachte man einen optimalen Steinerbaum S für $U \cup \{x\}$; wir können annehmen, dass sämtliche Blätter terminale sind. Ist $|\delta_S(x)| \geq 2$, so folgt

$$p(U \cup \{x\}) = c(E(S)) \geq q(U \cup \{x\}, x) = q(U \cup \{x\}, x) + \text{dist}_{(G,c)}(x, x),$$

wobei die Ungleichung folgt, indem man als U' die Terminalmenge in einer Zusammenhangskomponente von $S - x$ wählt.

Ist $|\delta_S(x)| = 1$, so sei y derjenige Knoten in S , der x am nächsten ist und für den $y \in U$ oder $|\delta_S(y)| \geq 3$ gilt. Wir unterscheiden zwei Fälle: Ist $y \in U$, so haben wir

$$p(U \cup \{x\}) = c(E(S)) \geq p(U) + \text{dist}_{(G,c)}(x, y),$$

anderenfalls haben wir

$$p(U \cup \{x\}) = c(E(S)) \geq \min_{y \in V(G) \setminus U} (q(U \cup \{y\}, y) + \text{dist}_{(G,c)}(x, y)).$$

Für (b) berechnen wir das Minimum über diese drei Formeln. \square

Diese Rekursionsformeln legen sofort den folgenden auf der dynamischen Optimierung basierenden Algorithmus nahe:

DREYFUS-WAGNER-ALGORITHMUS

Input: Ein ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$ und eine Menge $T \subseteq V(G)$.
Output: Die Länge $p(T)$ eines optimalen Steinerbaumes für T in G .

- ① **If** $|T| \leq 1$ **then** setze $p(T) := 0$ und **stop**.
 Berechne $\text{dist}_{(G,c)}(x, y)$ für alle $x, y \in V(G)$.
 Setze $p(\{x, y\}) := \text{dist}_{(G,c)}(x, y)$ für alle $x, y \in V(G)$.
- ② **For** $k := 2$ **to** $|T| - 1$ **do**:
 - For** alle $U \subseteq T$ mit $|U| = k$ und alle $x \in V(G) \setminus U$ **do**:

$$\text{Setze } q(U \cup \{x\}, x) := \min_{\emptyset \neq U' \subset U} (p(U' \cup \{x\}) + p((U \setminus U') \cup \{x\})).$$
 - For** alle $U \subseteq T$ mit $|U| = k$ und alle $x \in V(G) \setminus U$ **do**:

$$\text{Setze } p(U \cup \{x\}) := \min \left\{ \min_{y \in U} (p(U) + \text{dist}_{(G,c)}(x, y)), \right.$$

$$\left. \min_{y \in V(G) \setminus U} (q(U \cup \{y\}, y) + \text{dist}_{(G,c)}(x, y)) \right\}.$$

Satz 20.5. (Dreyfus und Wagner [1972]) Der DREYFUS-WAGNER-ALGORITHMUS bestimmt die Länge eines optimalen Steinerbaumes korrekt in $O(3^t n + 2^t n^2 + mn + n^2 \log n)$ -Zeit, wobei $n = |V(G)|$, $m = |E(G)|$ und $t = |T|$.

Beweis: Die Korrektheit folgt mit Lemma 20.4. In ① wird ein KÜRZESTEWEGE-PROBLEM FÜR ALLE PAARE gelöst; dies kann nach Satz 7.8 in $O(mn + n^2 \log n)$ -Zeit bewerkstelligt werden.

Die erste Rekursion in ② benötigt $O(3^t n)$ -Zeit, da es 3^t Möglichkeiten gibt, T in die drei Teilmengen U' , $U \setminus U'$ und $T \setminus U$ zu partitionieren. Die zweite Rekursion in ② benötigt offensichtlich $O(2^t n^2)$ -Zeit. \square

In obiger Form berechnet der DREYFUS-WAGNER-ALGORITHMUS die Länge eines optimalen Steinerbaumes, nicht aber einen optimalen Steinerbaum selbst. Dies kann jedoch leicht durch das Speichern weiterer Information und Backtracking erreicht werden. Dies haben wir bereits im Zusammenhang mit DIJKSTRAS ALGORITHMUS (Abschnitt 7.1) ausführlich besprochen. Hougardy, Silvanus und Vygen [2017] haben eine Variante vorgeschlagen, welche dieselbe Worst-Case-Laufzeit hat, aber in der Praxis viel schneller läuft.

Beachte, dass der Algorithmus i. A. exponentielle Zeit und exponentiellen Speicherplatz benötigt. Handelt es sich um eine beschränkte Anzahl von Terminalen, so läuft der Algorithmus in $O(n^3)$ -Zeit. Es gibt einen weiteren interessanten Spezialfall, wo der Algorithmus in polynomieller Zeit (und mit polynomiellem Speicherplatz) läuft: Ist G ein planarer Graph und liegen alle Terminalen am äußeren Gebiet, so kann der DREYFUS-WAGNER-ALGORITHMUS so modifiziert werden, dass er in

$O(n^3t^2)$ -Zeit läuft (Aufgabe 4). Für den allgemeinen Fall (mit vielen Terminalen) haben Fuchs et al. [2007] und Vygen [2011] die Laufzeit verbessert. Falls alle Kantengewichte kleine ganze Zahlen sind, ist der Algorithmus von Björklund et al. [2007] besser.

Da für das allgemeine STEINERBAUM-PROBLEM keine Hoffnung besteht, einen exakten polynomiellen Algorithmus zu finden, sind Approximationsalgorithmen durchaus wertvoll. Einige basieren auf der Idee, den optimalen Steinerbaum für T in G durch einen aufspannenden Baum minimalen Gewichtes in dem durch T induzierten Teilgraphen des metrischen Abschlusses von G zu approximieren.

Satz 20.6. *Sei G ein zusammenhängender Graph mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ und (\bar{G}, \bar{c}) der metrische Abschluss von G . Sei ferner $T \subseteq V(G)$. Ist S ein optimaler Steinerbaum für T in G und M ein aufspannender Baum minimalen Gewichtes in $\bar{G}[T]$ (bezüglich \bar{c}), so ist $\bar{c}(E(M)) \leq 2c(E(S))$.*

Beweis: Betrachte den zwei Kopien von jeder S -Kante enthaltenden Graphen H . Es ist H eulersch, also gibt es nach Satz 2.24 einen eulerschen Spaziergang W in H . Das erstmalige Auftreten der Elemente von T in W bestimmt eine Reihenfolge der Knoten von T und damit eine Tour W' in $\bar{G}[T]$. Da \bar{c} die Dreiecksungleichung $[\bar{c}(\{x, z\}) \leq \bar{c}(\{x, y\}) + \bar{c}(\{y, z\}) \leq c(\{x, y\}) + c(\{y, z\})$ für alle x, y, z] erfüllt, folgt

$$\bar{c}(W') \leq c(W) = c(E(H)) = 2c(E(S)).$$

Da W' einen aufspannenden Baum von $\bar{G}[T]$ enthält (man entferne einfach eine Kante), ist der Beweis abgeschlossen. \square

Dieser Satz ist von Gilbert und Pollak [1968] (mit Verweis auf E.F. Moore), Choukhmane [1978], Kou, Markowsky und Berman [1981], und Takahashi und Matsuyama [1980] veröffentlicht worden. Er legt sofort den folgenden 2-Approximationsalgorithmus nahe:

KOU-MARKOWSKY-BERMAN-ALGORITHMUS

Input: Ein zusammenhängender ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$ und eine Menge $T \subseteq V(G)$.

Output: Ein Steinerbaum für T in G .

- ① Berechne den metrischen Abschluss (\bar{G}, \bar{c}) und einen kürzesten Weg P_{st} für alle $s, t \in T$.
- ② Bestimme einen aufspannenden Baum M minimalen Gewichtes in $\bar{G}[T]$ (bezüglich \bar{c}).

Setze $E(S) := \bigcup_{\{x, y\} \in E(M)} E(P_{xy})$ und $V(S) := \bigcup_{\{x, y\} \in E(M)} V(P_{xy})$.

- ③ Liefere einen minimalen zusammenhängenden aufspannenden Teilgraphen von S als Output.

Satz 20.7. (Kou, Markowsky und Berman [1981]) *Der KOU-MARKOWSKY-BERMAN-ALGORITHMUS ist ein 2-Approximationsalgorithmus für das STEINERBAUM-PROBLEM und läuft in $O(k(m + n \log n))$ -Zeit, wobei $k = |T|$, $n = |V(G)|$ und $m = |E(G)|$.*

Beweis: Die Korrektheit und die Approximationsgüte folgen direkt mittels Satz 20.6. ① kann folgendermaßen implementiert werden: Für jedes $t \in T$ lässt man DIJKSTRAS ALGORITHMUS mit t als Quelle laufen. ② kann man in Laufzeit $O(k^2)$ mit PRIMS ALGORITHMUS (Satz 6.6) bewerkstelligen. ③ kann mittels BFS mit $O(kn)$ -Laufzeit implementiert werden. \square

Mehlhorn [1988] und Kou [1990] haben eine $O(m + n \log n)$ -Implementierung für diesen Algorithmus vorgeschlagen. Sie basiert auf der Idee, nicht $\tilde{G}[T]$, sondern einen ähnlichen Graphen zu berechnen, dessen aufspannende Bäume minimalen Gewichtes auch aufspannende Bäume minimalen Gewichtes in $\tilde{G}[T]$ sind.

Der aufspannende Baum minimalen Gewichtes selbst liefert eine 2-Approximation für jede metrische Instanz des STEINERBAUM-PROBLEMS. Ein Algorithmus mit einer besseren Approximationsgüte war unbekannt, bis Zelikovsky [1993] seinen $\frac{11}{6}$ -Approximationsalgorithmus für das STEINERBAUM-PROBLEM fand. Seitdem ist die Approximationsgüte weiter auf 1,75 (Berman und Ramaiyer [1994]), auf 1,65 (Karpinski und Zelikovsky [1997]), auf 1,60 (Hougardy und Prömel [1999]), auf 1,55 (Robins und Zelikovsky [2005]) (siehe Abschnitt 20.2), und schließlich 1,39 Byrka et al. [2013] (siehe Abschnitt 20.3), verbessert worden.

Andererseits folgt mit Satz 20.3 und Korollar 16.40, dass es kein Approximationsschema geben kann, sofern $P \neq NP$. In der Tat haben Chlebík und Chlebíková [2008] gezeigt, dass es keinen 1,01-Approximationsalgorithmus für das STEINERBAUM-PROBLEM gibt, sofern $P \neq NP$.

Ein Algorithmus, der recht effizient optimale Steinerbäume berechnet, besonders für das MANHATTAN-STEINERBAUM-PROBLEM, wurde von Warne, Winter und Zachariasen [2000] entwickelt.

20.2 Der Robins-Zelikovsky-Algorithmus

Definition 20.8. Ein voller Steinerbaum für eine Menge T von Terminalen in einem Graphen G ist ein Baum Y in G , dessen Blättermenge gerade die Menge T ist. Jeder minimale Steinerbaum für T kann in volle Steinerbäume für Teilmengen von T zerlegt werden; diese sind seine **vollen Komponenten**. Eine Vereinigung voller Komponenten, von denen jede höchstens k Terminalen hat, heißt **k -beschränkt** (bezüglich der gegebenen Menge der Terminalen). Genauer: Ein Graph Y heißt k -beschränkt (in G bezüglich T), falls es volle Steinerbäume Y_i für $T \cap V(Y_i)$ in G mit $|T \cap V(Y_i)| \leq k$ ($i = 1, \dots, t$) gibt, so dass erstens der aus $\{(i, v) : i = 1, \dots, t, v \in V(Y_i)\}, \{(i, v), (i, w) : i = 1, \dots, t, \{v, w\} \in E(Y_i)\}$ mittels Kontraktion der Menge $\{i : v \in V(Y_i)\} \times \{v\}$ für jedes $v \in T$ hervorgehende Graph zusammenhängend ist, zweitens $V(Y) = \bigcup_{i=1}^t V(Y_i)$ gilt, und drittens

$E(Y)$ die disjunkte Vereinigung der Mengen $E(Y_i)$ ist. Beachte, dass parallele Kanten auftreten können.

Wir definieren den **k -Steiner-Quotienten** ρ_k folgendermaßen:

$$\rho_k := \sup_{(G,c,T)} \left\{ \frac{\min\{c(E(Y)) : Y \text{ ist ein } k\text{-beschränkter Konnektor für } T\}}{\min\{c(E(Y)) : Y \text{ ist ein Steinerbaum für } T\}} \right\},$$

wobei das Supremum über alle Instanzen des STEINERBAUM-PROBLEMS genommen wird.

Es bestehen z. B. 2-beschränkte Konnektoren aus Wegen zwischen Terminalen. Somit entsprechen optimale 2-beschränkte Konnektoren für T in (G, c) den aufspannenden Bäumen minimalen Gewichtes in $(\bar{G}[T], \bar{c})$; demnach gilt $\rho_2 \leq 2$ nach Satz 20.6. Die Sterne mit Einheitsgewichten beweisen in der Tat, dass $\rho_2 = 2$ (und dass allgemein $\rho_k \geq \frac{k}{k-1}$). Beschränken wir das Supremum auf Instanzen des MANHATTAN-STEINERBAUM-PROBLEMS, so ist der Steiner-Quotient besser, z. B. $\frac{3}{2}$ für $k = 2$ (Hwang [1976]).

Satz 20.9. (Du, Zhang und Feng [1991]) Es gilt $\rho_{2^s} \leq \frac{s+1}{s}$.

Beweis: Sei (G, c, T) eine Instanz und Y ein optimaler Steinerbaum im metrischen Abschluss. O. B. d. A. können wir annehmen, dass Y ein voller Steinerbaum ist (sonst betrachten wir volle Komponenten separat). Indem wir gegebenenfalls Knoten verdoppeln und Kanten der Länge Null hinzufügen, können wir ferner annehmen, dass Y ein voller binärer Baum ist, dessen Blätter genau die Terminale sind. Ein Knoten, nämlich die Wurzel, hat Grad 2 und alle anderen Steinerpunkte Grad 3. Wir sagen, dass ein Knoten $v \in V(Y)$ auf der Höhe i ist, falls seine Distanz von der Wurzel gleich i ist. Alle Terminale sind auf der gleichen Höhe h (der Höhe des binären Baumes).

Wir definieren nun s 2 ^{s} -beschränkte Konnektoren für T , dessen Gesamtlänge höchstens gleich $(s+1)c(E(Y))$ ist. Für jedes $v \in V(Y)$ sei $P(v)$ ein Weg in Y von v aus zu irgendeinem Blatt, so dass all diese Wege paarweise kantendisjunkt sind (z. B. einmal links abwärts und dann immer rechts).

Sei Y_i für $i = 1, \dots, s$ die Vereinigung der folgenden vollen Komponenten:

- der von den Knoten bis zur Höhe i induzierte Teilbaum von Y plus $P(v)$ für jeden Knoten v auf der Höhe i ;
- für jeden Knoten u auf der Höhe $ks + i$: Der von den Nachfolgern von u bis zur Höhe $(k+1)s + i$ induzierte Teilbaum plus $P(v)$ für jeden Knoten v in dem Teilbaum auf der Höhe $(k+1)s + i$ ($k = 0, \dots, \lfloor \frac{h-1-i}{s} \rfloor - 1$);
- für jeden Knoten u auf der Höhe $\lfloor \frac{h-1-i}{s} \rfloor s + i$: Der von allen Nachfolgern von u induzierte Teilbaum.

Offensichtlich ist jeder dieser Bäume 2 ^{s} -beschränkt und die Vereinigung der Bäume in Y_i ist Y , d. h. ein Konnektor für T . Auch enthält jede der Mengen Y_i jede Kante von Y genau einmal, deren Auftreten in einem Weg $P(v)$ nicht mitgezählt. Ferner

wird jeder Weg $P(v)$ in nur einem Y_i benutzt. Somit kommt jede Kante höchstens $s + 1$ mal vor. \square

Insbesondere geht $\rho_k \rightarrow 1$ mit $k \rightarrow \infty$. Somit können wir nicht erwarten, dass wir für ein festes k den optimalen k -beschränkten Konnektor in polynomieller Zeit finden können. Tatsächlich ist dieses Problem für jedes feste $k \geq 4$ NP-schwer (Garey und Johnson [1977]). Die Schranke in Satz 20.9 ist bestmöglich: Borchers und Du [1997] haben bewiesen, dass $\rho_k = \frac{(s+1)2^s+t}{s2^s+t}$ für alle $k \geq 2$, wobei $k = 2^s+t$ und $0 \leq t < 2^s$.

Wir werden einen Algorithmus besprechen, der mit einem aufspannenden Baum minimalen Gewichtes in dem von T induzierten Teilgraphen des metrischen Abschlusses beginnt und dann versucht, ihn mittels k -beschränkter voller Steinerbäume zu verbessern. Dieser Algorithmus beschließt jedoch nur die Hinzunahme von höchstens der Hälfte eines solchen Steinerbaumes, des sogenannten Verlustes des Steinerbaumes. Für einen gegebenen Steinerbaum Y ist ein **Verlust** von Y eine Kantenmenge F mit minimalen Gesamtkosten, deren Kanten jeden Steinerpunkt von Grad mindestens gleich 3 mit einem Terminal verbinden. Abbildung 20.2 zeigt ein Beispiel eines vollen Steinerbaumes mit einem Verlust (die fetten Kanten), unter der Annahme, dass die Kantenkosten proportional zu den Kantenlängen sind.

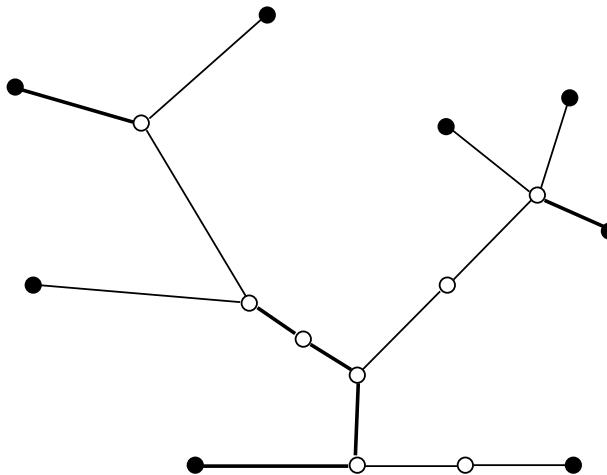


Abbildung 20.2.

Proposition 20.10. Sei Y ein voller Steinerbaum für T , $c : E(Y) \rightarrow \mathbb{R}_+$ und F ein Verlust von Y . Dann ist $c(F) \leq \frac{1}{2}c(E(Y))$.

Beweis: Sei r ein beliebiger Knoten von Y . Ferner sei U die Menge der Steinerpunkte $v \in V(Y) \setminus T$ mit Grad mindestens gleich 3. Für jedes $v \in U$ sei $P(v)$ ein Weg mit minimalen Kosten aus der Menge (der Kardinalität mindestens gleich

zwei) derjenigen Wege von v nach einem Knoten $w \in U \cup T$ mit der Eigenschaft, dass die Distanz zwischen w und r größer ist als die zwischen v und r . Die Vereinigung der Kantenmengen dieser Wege verbindet jedes Element von U mit einem Terminal und hat höchstens die Hälfte der Gesamtkosten der Kantenmenge von Y . \square

Anstatt dass wir die Verluste von k -beschränkten vollen Steinerbäumen explizit kontrahieren, modifizieren wir die Kosten wie folgt:

Proposition 20.11. *Sei G ein vollständiger Graph, $T \subseteq V(G)$, $c : E(G) \rightarrow \mathbb{R}_+$ und $k \geq 2$. Sei ferner $S \subseteq T$ mit $|S| \leq k$, Y ein Steinerbaum für S in G und L ein Verlust von Y . Setze $c'(e) := 0$ für $e \in L$ und $c'(e) := c(e)$ sonst. Wir definieren $c/(Y, L) : E(G) \rightarrow \mathbb{R}_+$ durch $c/(Y, L)(\{v, w\}) := \min\{c(\{v, w\}), \text{dist}_{(Y, c')}(v, w)\}$ für $v, w \in S$ mit $v \neq w$ und $c/(Y, L)(e) := c(e)$ für alle weiteren Kanten. Dann gilt:*

- (a) *Es gibt einen aufspannenden Baum M in $G[S]$ mit $c/(Y, L)(E(M)) + c(L) \leq c(E(Y))$.*
- (b) *Für jeden k -beschränkten Konnektor H' für T in G gibt es einen k -beschränkten Konnektor H für T in G mit $c(E(H)) \leq c/(Y, L)(E(H')) + c(L)$.*

Beweis: (a): Wir können annehmen, dass L minimal ist, folglich enthält jede Zusammenhangskomponente von $(V(Y), L)$ höchstens ein Element aus S . Man partitioniere nun $E(Y) \setminus L$ in Kantenmengen maximaler Wege in Y , bei denen alle internen Knoten den Grad 2 in Y haben. Für jeden solchen Weg P gibt es zwei terminale $s, t \in S$ mit der Eigenschaft, dass der s - t -Weg Q in Y nur Kanten von P und L enthält. Die disjunkte Vereinigung dieser Wege (je einer für jedes solche P) enthält jede Kante von $E(Y) \setminus L$ genau einmal und jede Kante von L genau einmal. Somit ist sie zusammenhängend und entspricht einem aufspannenden Baum M in $G[S]$ mit $\sum_{\{s, t\} \in E(M)} \text{dist}_{(Y, c')}(s, t) \leq c(E(Y) \setminus L)$, womit (a) folgt.

(b): Zum Beweis der zweiten Aussage sei H' ein k -beschränkter Konnektor für T . Ersetze jede Kante $e = \{v, w\} \in E(H')$ mit $c/(Y, L)(e) < c(e)$ durch einen kürzesten v - w -Weg in (Y, c') und eliminiere parallele Kanten. Dann ist der resultierende Graph H ein k -beschränkter Konnektor für T , für den $c(E(H)) = c'(E(H)) + c(E(H) \cap L) \leq c/(Y, L)(E(H')) + c(L)$ gilt. \square

Wir werden die Kostenfunktion wiederholt durch Hinzufügen von Kanten modifizieren, denen volle Komponenten entsprechen. Das folgende Resultat besagt, dass die Reduktion der Gesamtkosten eines aufspannenden Baumes minimalen Gewichtes nicht wächst, wenn vorher noch weitere Kanten hinzugefügt werden:

Lemma 20.12. (Zelikovsky [1993], Berman und Ramaiyer [1994]) *Sei G ein Graph, $c : E(G) \rightarrow \mathbb{R}_+$, $T \subseteq V(G)$, (T, U) ein weiterer Graph und $c' : U \rightarrow \mathbb{R}_+$. Sei $m : 2^U \rightarrow \mathbb{R}_+$, wobei $m(X)$ die Gesamtkosten eines aufspannenden Baumes minimalen Gewichtes in $(T, E(G[T]) \cup X)$ sind. Dann ist m supermodular.*

Beweis: Sei $A \subseteq U$ und $f \in U$. Wir wenden KRUSKALS ALGORITHMUS parallel auf $G[T]$ und $G[T] + f$ an und prüfen die Kanten von $G[T]$ in derselben Reihenfolge (mit steigenden Kosten). Beide Abläufe stimmen überein, außer dass der erste nicht f wählt, während der zweite nicht die erste Kante wählt, die einen f enthaltenden Kreis in $G + f$ schließt. Somit unterscheiden sich die minimalen Kosten der aufspannenden Bäume in den zwei Graphen um:

$\min\{\gamma : G[T] + f \text{ enthält einen } f \text{ enthaltenden Kreis, dessen Kanten Gesamtkosten höchstens gleich } \gamma \text{ haben}\} - c'(f)$.

Offensichtlich wird diese Differenz nur dann abnehmen, wenn wir $G[T] + A$ und $(G[T] + A) + f$ betrachten, statt $G[T]$ und $G[T] + f$. Somit gilt

$$m(A) - m(A \cup \{f\}) \leq m(\emptyset) - m(\{f\}).$$

Seien nun $X, Y \subseteq U$, $Y \setminus X = \{y_1, \dots, y_k\}$ und $m_i(Z) := m((X \cap Y) \cup \{y_1, \dots, y_{i-1}\} \cup Z)$ für $i = 1, \dots, k$. Wenden wir Obiges auf m_i an, so erhalten wir Supermodularität:

$$\begin{aligned} m(X) - m(X \cup Y) &= \sum_{i=1}^k (m_i(X \setminus Y) - m_i((X \setminus Y) \cup \{y_i\})) \\ &\leq \sum_{i=1}^k (m_i(\emptyset) - m_i(\{y_i\})) \\ &= m(X \cap Y) - m(Y). \end{aligned}$$

□

Nun beschreiben wir den Algorithmus. Mit $mst(c)$ bezeichnen wir die minimalen Kosten eines aufspannenden Baumes in dem von T induzierten Teilgraphen von (\bar{G}, c) .

ROBINS-ZELIKOVSKY-ALGORITHMUS

Input: Ein ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$ und eine Menge $T \subseteq V(G)$ von Terminalen. Eine Zahl $k \geq 2$.

Output: Ein Steinerbaum für T in G .

- ① Berechne den metrischen Abschluss (\bar{G}, \bar{c}) von (G, c) .
- ② Wähle eine Teilmenge S von höchstens k Terminalen und ein Paar $K = (Y, L)$, wobei Y ein optimaler Steinerbaum für S ist und L ein Verlust von Y , so dass $\frac{mst(\bar{c}) - mst(\bar{c}/K)}{\bar{c}(L)}$ maximiert wird und mindestens gleich 1 ist.
- If es gibt keine solche Wahl then go to ④.
- ③ Setze $\bar{c} := \bar{c}/K$.
- Go to ②.

- ④ Berechne einen aufspannenden Baum minimalen Gewichtes in dem von T induzierten Teilgraphen von (\bar{G}, \bar{c}) .
 Ersetze alle Kanten durch kürzeste Wege in (G, c') , wobei $c'(e) := 0$ falls $e \in L$ für eine im Algorithmus gewählte Menge L und $c'(e) = c(e)$ sonst.
 Berechne abschließend einen minimalen zusammenhängenden Teilgraphen, der T aufspannt.
-

Angenommen, der Algorithmus terminiert bei Iteration $t + 1$. Es sei $K_i := (Y_i, L_i)$ der Steinerbaum und sein in der i -ten Iteration gewählter Verlust ($i = 1, \dots, t$). Sei ferner c_0 die Kostenfunktion \bar{c} nach ①, und sei $c_i := c_{i-1}/K_i$ die Kostenfunktion \bar{c} nach i Iterationen ($i = 1, \dots, t$).

Lemma 20.13. *Der Algorithmus berechnet einen Steinerbaum für T mit Gewicht höchstens $mst(c_t) + \sum_{i=1}^t c(L_i)$.*

Beweis: Dies folgt aus Proposition 20.11(b). □

Sei Y^* ein optimaler k -beschränkter Konnektor für T , seien $Y_1^*, \dots, Y_{t^*}^*$ k -beschränkte volle Steinerbäume mit $Y^* := Y_1^* \cup \dots \cup Y_{t^*}^*$, sei L_j^* ein Verlust von Y_j^* und $K_j^* := (Y_j^*, L_j^*)$ ($j = 1, \dots, t^*$), und sei $L^* := L_1^* \cup \dots \cup L_{t^*}^*$. Wir schreiben c/K^* statt $c/K_1^*/\dots/K_{t^*}^*$.

Lemma 20.14. $mst(c/K^*) + c(L^*) = c(E(Y^*))$. □

Beweis: Die Ungleichung “ \leq ” folgt mit Proposition 20.11(a); die andere mit Proposition 20.11(b). □

Lemma 20.15. *Es gilt $mst(c_t) \leq c(E(Y^*)) \leq mst(c_0)$.*

Beweis: Es gilt $c(E(Y^*)) \leq mst(c_0)$ trivialerweise. Bei Terminierung des Algorithmus gilt $mst(c_t) - mst(c_t/K_j^*) \leq c(L_j^*)$ für $j = 1, \dots, t^*$. Mit Lemma 20.12 folgt sodann

$$\begin{aligned} mst(c_t) - mst(c/K^*) &\leq mst(c_t) - mst(c_t/K^*) \\ &= \sum_{j=1}^{t^*} (mst(c_t/K_1^*/\dots/K_{j-1}^*) - mst(c_t/K_1^*/\dots/K_j^*)) \\ &\leq \sum_{j=1}^{t^*} (mst(c_t) - mst(c_t/K_j^*)) \\ &\leq \sum_{j=1}^{t^*} c(L_j^*), \end{aligned}$$

und damit haben wir $mst(c_t) \leq mst(c/K^*) + c(L^*)$. Nun benutze man Lemma 20.14. □

Lemma 20.16. Es gilt $mst(c_t) + \sum_{i=1}^t c(L_i) \leq c(E(Y^*))\left(1 + \frac{\ln 3}{2}\right)$.

Beweis: Sei $i \in \{1, \dots, t\}$. Mit der Wahl von L_i in der i -ten Iteration des Algorithmus folgt dann (mittels Lemma 20.12 für die dritte Ungleichung und Monotonie für die letzte Ungleichung):

$$\begin{aligned} \frac{mst(c_{i-1}) - mst(c_i)}{c(L_i)} &\geq \max_{j=1, \dots, t^*} \frac{mst(c_{i-1}) - mst(c_{i-1}/K_j^*)}{c(L_j^*)} \\ &\geq \frac{\sum_{j=1}^{t^*} (mst(c_{i-1}) - mst(c_{i-1}/K_j^*))}{\sum_{j=1}^{t^*} c(L_j^*)} \\ &\geq \frac{\sum_{j=1}^{t^*} (mst(c_{i-1}/K_1^*/\dots/K_{j-1}^*) - mst(c_{i-1}/K_1^*/\dots/K_j^*))}{c(L^*)} \\ &= \frac{mst(c_{i-1}) - mst(c_{i-1}/K^*)}{c(L^*)} \\ &\geq \frac{mst(c_{i-1}) - mst(c/K^*)}{c(L^*)}. \end{aligned}$$

Die linke Seite ist aber mindestens gleich 1. Somit folgt

$$\begin{aligned} \sum_{i=1}^t c(L_i) &\leq \sum_{i=1}^t (mst(c_{i-1}) - mst(c_i)) \frac{c(L^*)}{\max\{c(L^*), mst(c_{i-1}) - mst(c/K^*)\}} \\ &\leq \int_{mst(c_t)}^{mst(c_0)} \frac{c(L^*)}{\max\{c(L^*), x - mst(c/K^*)\}} dx. \end{aligned}$$

Nach Lemma 20.14 gilt $c(E(Y^*)) = mst(c/K^*) + c(L^*)$ und nach Lemma 20.15 gilt $mst(c_t) \leq c(E(Y^*)) \leq mst(c_0)$. Damit folgt

$$\begin{aligned} \sum_{i=1}^t c(L_i) &\leq \int_{mst(c_t)}^{mst(c/K^*)+c(L^*)} 1 dx + \int_{c(L^*)}^{mst(c_0)-mst(c/K^*)} \frac{c(L^*)}{x} dx \\ &= c(E(Y^*)) - mst(c_t) + c(L^*) \ln \frac{mst(c_0) - mst(c/K^*)}{c(L^*)}. \end{aligned}$$

Da $mst(c_0) \leq 2 \text{OPT}(G, c, T) \leq c(E(Y^*)) + mst(c/K^*) + c(L^*)$ (nach Lemma 20.14), folgt

$$mst(c_t) + \sum_{i=1}^t c(L_i) \leq c(E(Y^*)) \left(1 + \frac{c(L^*)}{c(E(Y^*))} \ln \left(1 + \frac{c(E(Y^*))}{c(L^*)}\right)\right).$$

Nach Proposition 20.10 gilt $0 \leq c(L^*) \leq \frac{1}{2}c(E(Y^*))$, und $\max\{x \ln(1 + \frac{1}{x}) : 0 < x \leq \frac{1}{2}\}$ wird in $x = \frac{1}{2}$ angenommen (da die Ableitung $\ln(1 + \frac{1}{x}) - \frac{1}{x+1}$ stets positiv ist). Somit folgt $mst(c_t) + \sum_{i=1}^t c(L_i) \leq c(E(Y^*))\left(1 + \frac{\ln 3}{2}\right)$. \square

Dieser Beweis stammt im Wesentlichen von Gröpl et al. [2001]. Als Folgerung haben wir:

Satz 20.17. (Robins und Zelikovsky [2005]) *Der Robins-Zelikovsky-Algorithmus hat die Approximationsgüte $\rho_k(1 + \frac{\ln 3}{2})$ und läuft für festes k in polynomieller Zeit. Für genügend großes k ist die Approximationsgüte kleiner als 1,55.*

Beweis: Nach Lemma 20.13 ist der Output des Algorithmus ein Steinerbaum mit Kosten höchstens gleich $mst(c_t) + \sum_{i=1}^t c(L_i)$. Nach Lemma 20.16 ist dies höchstens gleich $\rho_k(1 + \frac{\ln 3}{2})$. Wählen wir $k = \min\{|V(G)|, 2^{2233}\}$ und wenden Satz 20.9 an, so erhalten wir eine Approximationsgüte gleich $\rho_k(1 + \frac{\ln 3}{2}) \leq \frac{2234}{2233}(1 + \frac{\ln 3}{2}) < 1,55$.

Es gibt höchstens n^k mögliche Teilmengen S und für jedes S gibt es höchstens n^{k-2} Wahlen für die (höchstens $k-2$) Steinerpunkte mit Grad mindestens gleich 3 in einem optimalen Steinerbaum Y für S . Demnach gibt es für ein gegebenes Y höchstens $(2k-3)^{k-2}$ Wahlen eines Verlustes (bis auf Kanten mit verschwindenden Kosten). Somit benötigt jede Iteration (für festes k) $O(n^{2k})$ -Zeit, und es gibt höchstens n^{2k-2} Iterationen. \square

20.3 Rundung des Gerichtete-Komponenten-LP

In diesem Abschnitt präsentieren wir den momentan bekanntesten Approximationsalgorithmus für das STEINERBAUM-PROBLEM. Er stammt von Byrka et al. [2013], und seine Approximationszahl liegt beliebig nahe bei $\ln 4$.

Sei (G, c, T) eine Instanz des STEINERBAUM-PROBLEMS. Wiederum beschränken wir uns auf k -beschränkte volle Komponenten für ein genügend großes k . Hier werden wir sie aber orientieren: Sei \mathcal{C}^k die Menge der Paare (t, R) für alle $t \in R \subseteq T$ mit $|R| \leq k$; diese nennen wir die (gerichteten) Komponenten. Für jedes $C = (t, R) \in \mathcal{C}^k$ geben wir mit $c(C)$ die minimalen Kosten eines Steinerbaumes für R in (G, c) an. Man beachte, dass man $c(C)$ mit $C \in \mathcal{C}^k$ mittels Satz 20.5 für jedes feste k in polynomieller Zeit berechnen kann.

Man wähle ein festes Terminal $r \in T$ als Wurzel. Für $r \in U \subset T$ sei $\delta_{\mathcal{C}^k}^+(U)$ die Menge aller $(t, R) \in \mathcal{C}^k$ mit $t \in U$ und $R \setminus U \neq \emptyset$. Nun können wir das sogenannte “Gerichtete-Komponenten-LP” angeben:

$$\begin{aligned} \min \quad & \sum_{C \in \mathcal{C}^k} c(C)x_C \\ \text{bzgl.} \quad & \sum_{C \in \delta_{\mathcal{C}^k}^+(U)} x_C \geq 1 \quad (r \in U \subset T) \\ & x_C \geq 0 \quad (C \in \mathcal{C}^k) \end{aligned} \tag{20.1}$$

Dieses LP ist eine Relaxierung des Problems: "Man finde einen k -beschränkten Steinerbaum mit minimalen Kosten", denn es kann jeder k -beschränkte Steinerbaum als Arboreszenz mit Wurzel in r orientiert werden, und man kann $x_{(t,R)} := 1$ für jedes R setzen, das eine volle Komponente mit Wurzel t ist.

Das LP (20.1) kann für jedes feste k in polynomieller Zeit gelöst werden, da die Anzahl der Variablen $O(k|T|^k)$ ist und die Schnittnebenbedingungen folgendermaßen ersetzt werden können: Man fordert einen r - s -Fluss mit Wert 1 für alle $s \in T \setminus \{r\}$ in einem Hilfsdigraph mit Knotenmenge $T \cup \{v_C : C \in \mathcal{C}^k\}$ und Kanten (t, v_C) und (v_C, s) für $s \in R \setminus \{t\}$ mit Kapazität x_C für jedes $C = (t, R) \in \mathcal{C}^k$ (Aufgabe 9).

Der Algorithmus von Byrka et al. [2013] löst das LP (20.1), wählt eine Komponente $C = (t, R)$, wobei $C \in \mathcal{C}^k$ mit der Wahrscheinlichkeit $\frac{x_C}{\sum_{C' \in \mathcal{C}^k} x_{C'}}$ gewählt wird, kontrahiert R und iteriert weiter, bis nur ein Terminal übrig ist. Für $\mathcal{C}' \subseteq \mathcal{C}^k$ sei $(G, c, T)/\mathcal{C}'$ diejenige Instanz, die durch Kontraktion von R für jedes $(t, R) \in \mathcal{C}'$ entsteht (und die durch Kontraktion entstehenden Knoten sind Terminalen).

ITERATIVER RANDOMISIERTER RUNDUNGSALGORITHMUS

Input: Ein ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$ und eine Menge $T \subseteq V(G)$ von Terminalen. Eine Zahl $k \geq 2$.

Output: Ein Steinerbaum für T in G .

- ① $\mathcal{C}' := \emptyset$.
- ② Berechne eine optimale Lösung x für das LP (20.1) für $(G, c, T)/\mathcal{C}'$.
- ③ Wähle eine Komponente $C \in \mathcal{C}^k$ zufällig, wobei jedes C mit der Wahrscheinlichkeit $\frac{x_C}{\sum_{C' \in \mathcal{C}^k} x_{C'}}$ gewählt wird.
- ④ Setze $\mathcal{C}' := \mathcal{C}' \cup \{C\}$.
- ⑤ If es ist mehr als ein Terminal übrig then go to ②.
- ⑥ Seien $V(Y) = \bigcup_{C \in \mathcal{C}'} V(Y_C)$ und $E(Y) = \bigcup_{C \in \mathcal{C}'} E(Y_C)$, wobei Y_C für $C = (t, R) \in \mathcal{C}'$ ein optimaler Steinerbaum für R in (G, c) ist.
Gib einen minimalen zusammenhängenden alle Terminalen enthaltenden Teilgraph von Y zurück.

Der Algorithmus kann derandomisiert werden, die obige randomisierte Version lässt sich aber leichter analysieren. Sei \mathcal{C}'_t die Menge \mathcal{C}' nach t Iterationen.

Wir werden den Fortschritt mittels eines Baumes (T, W) messen; dieser wird wie folgt zufällig gewählt.

Sei Y_k^* ein optimaler k -beschränkter Steinerbaum für T . Man transformiere Y_k^* in einen vollen Steinerbaum Y^* mit denselben Kosten, in dem alle Steinerpunkte Grad 3 haben außer einem Steinerpunkt r^* mit Grad 2 (wie im Beweis von Satz 20.9; siehe Abb. 20.3, wo die Kontraktion der punktierten Kanten in Y^* (Abbildung rechts) Y_k^* (Abbildung links) ergibt; Terminalen sind als Quadrate angezeigt). Wir können Y^* als einen Baum mit Wurzel in r^* betrachten, dessen Blätter die Terminalen sind. Unabhängig für jeden Knoten $v \in V(Y^*) \setminus T$ wählen wir nun eine der beiden zu den Kindern von v führenden Kanten als “Verliererkante”, jede mit der Wahrscheinlichkeit $\frac{1}{2}$. Sei L die Menge der Verliererkanten. Wir können (Y^*, L) als ein Ausscheidungsturnier betrachten, in dem die Terminalen die Spieler sind

und die Steinerpunkte die Spiele, und jeder Spieler entlang seinem Weg bis r^* spielt, aber ausscheidet wenn er eine Verliererkante passiert. Sei nun W die Menge derjenigen Terminalpaare, die sich in diesem Turnier einander gegenüberstehen; genauer

$$W := \left\{ \{u, v\} \in \binom{T}{2} : \text{der } u-v\text{-Weg in } Y^* \text{ enthält nur eine Verliererkante} \right\}.$$

Es ist (T, W) ein Baum; er heißt *Zeugenbaum* (die unteren gebogenen Kanten des rechten Bildes in Abb. 20.3). Für jedes $e \in E(Y^*)$ sei $W(e)$ die Menge der Kanten $\{u, v\} \in W$ mit der Eigenschaft: Der $u-v$ -Weg in Y^* enthält e . Die Menge $W(e)$ entspricht denjenigen Spielen, die der Gewinner von Spiel v nach v spielt, wobei v derjenige Endknoten von e ist, der weiter entfernt von r^* ist. Somit folgt: Für eine Verliererkante e gilt $|W(e)| = 1$ und die Wahrscheinlichkeit, dass $|W(e)|$ mehr als q Elemente enthält, ist höchstens 2^{-q} (für alle $q \in \mathbb{N}$).

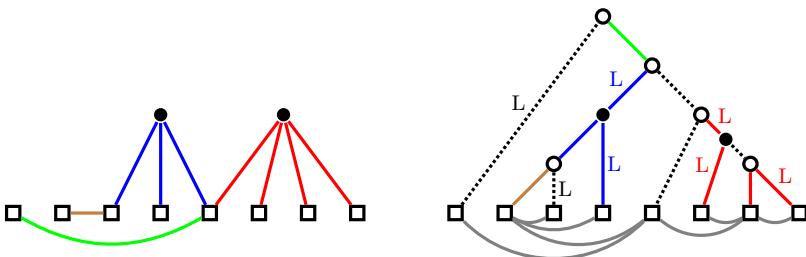


Abbildung 20.3.

Wir registrieren den Fortschritt, indem wir Kanten des Zeugenbaumes so markieren, dass die nicht markierten Kanten von W die Terminalen verbinden (nach Kontraktion der während früherer Iterationen gewählten Komponenten). Es sei $\mathcal{B}_W(R)$ die Menge der verschiedenen Markierungsmöglichkeiten der Kanten des Zeugenbaumes, falls die Komponente $C = (t, R)$ gewählt wird. Sie heißt die Familie der *Briickenmengen* von R , genauer:

$$\mathcal{B}_W(R) := \{B \subseteq W : (W \setminus B)/R \text{ ist ein Baum}\}.$$

Wir möchten zeigen, dass man für die Verbindung der Terminalen (nach Kontraktion der während früherer Iterationen gewählten Komponenten) keine Kanten $e \in E(Y^*)$ benötigt, für die sämtliche Kanten von $W(e)$ markiert sind:

Lemma 20.18. *Seien (T, W) ein Baum, $t \in \mathbb{N}$, und $C_1, \dots, C_t \in \mathcal{C}^k$. Seien $W_0 := W$, $B_i \in \mathcal{B}_{(T, W_{i-1})/C_1/\dots/C_{i-1}}(R)$ (wobei $C_i = (t, R)$), und $W_i := W_{i-1} \setminus B_i$ für $i = 1, \dots, t$. Dann sind alle Terminalen in derselben Zusammenhangskomponente von $(V(Y^*), \{e \in E(Y^*) : W(e) \cap W_t \neq \emptyset\})/C_1/\dots/C_t$.*

Beweis: Zunächst folgt mittels Induktion über t , dass $(T, W_t)/C_1/\dots/C_t$ ein Baum ist, unter Bezugnahme auf die Definition von Brückenmengen.

Wir nehmen nun an, dass $(V(Y^*), \{e \in E(Y^*) : W(e) \cap W_t \neq \emptyset\})/C_1/\dots/C_t$ Terminale in verschiedenen Zusammenhangskomponenten hat. Dann gibt es eine Teilmenge $U \subset V(Y^*)$ mit $U \cap T \neq \emptyset$ und $T \setminus U \neq \emptyset$, so dass $R \subseteq U$ oder $R \cap U = \emptyset$ für alle $(t, R) \in \{C_1, \dots, C_t\}$ und $W(e) \cap W_t = \emptyset$ für alle $e \in \delta_{Y^*}(U)$. Da $(T, W_t)/C_1/\dots/C_t$ zusammenhängend ist, gibt es eine Kante $\{u, v\} \in W_t$ mit $u \in U$ und $v \notin U$. Damit enthält der u - v -Weg in Y^* eine Kante $e \in \delta_{Y^*}(U)$. Dann gilt jedoch $\{u, v\} \in W(e)$, ein Widerspruch. \square

Um zu zeigen, dass wir Kanten schnell markieren können, benötigen wir das folgende Brückenlemma:

Lemma 20.19. Seien (T, W) ein Baum, $d : W \rightarrow \mathbb{R}_+$ und x eine zulässige Lösung des Gerichtete-Komponenten-LP (20.1). Dann gilt

$$\sum_{e \in W} d(e) \leq \sum_{C=(t,R) \in \mathcal{C}^k} x_C \max \left\{ \sum_{e \in B} d(e) : B \in \mathcal{B}_W(R) \right\}.$$

Beweis: Sei $e \in \binom{T}{2}$ und setze $\bar{d}(e) := \max\{d(f) : f \in E(P_e)\}$, wobei P_e der Weg in (T, W) zwischen den Endknoten von e ist. Da der Baum (T, W) die Bedingung (b) des Satzes 6.3 bzgl. des vollständigen Graphen auf T mit Gewichten \bar{d} erfüllt, ist er ein aufspannender Baum mit minimalem Gewicht bzgl. \bar{d} .

Es seien $C = (t, R) \in \mathcal{C}^k$ und $B_C \in \mathcal{B}_W(R)$, so dass $\sum_{e \in B_C} d(e)$ maximiert wird. Sei $r(u)$ mit $u \in T$ dasjenige Element von R , welches in derselben Zusammenhangskomponente von $(T, W \setminus B_C)$ liegt wie u . Nun konstruieren wir einen aufspannenden Baum (R, S_C) , indem wir $S_C := \{\{r(u), r(v)\} : \{u, v\} \in B_C\}$ setzen.

Man beachte, dass $\bar{d}(\{r(u), r(v)\}) = d(\{u, v\})$ für alle $\{u, v\} \in B_C$, da B_C die Summe $\sum_{e \in B_C} d(e)$ maximiert. Also folgt $\sum_{e \in S_C} \bar{d}(e) = \sum_{e \in B_C} d(e)$. Wir können (R, S_C) als Arboreszenz mit Wurzel in t orientieren und bezeichnen die Menge der orientierten Kanten mit \vec{S}_C .

Für $e = (v, w) \in T \times T$ mit $v \neq w$ setzen wir nun $y_e := \sum_{C \in \mathcal{C}^k : e \in \vec{S}_C} x_C$. Dann gilt $\sum_{e \in U \times (T \setminus U)} y_e \geq \sum_{C \in \delta_{\mathcal{C}^k}^+} x_C \geq 1$ für alle $r \in U \subset T$. Somit ist y eine zulässige Lösung des LP in Korollar 6.15 angewendet auf den vollständigen Digraphen auf der Knotenmenge T . Da (T, W) ein aufspannender Baum mit minimalem Gewicht bzgl. \bar{d} ist (und als Arboreszenz mit Wurzel in r orientiert werden kann), gilt

$$\begin{aligned} \sum_{e \in W} \bar{d}(e) &\leq \sum_{e=(v,w) \in T \times T : v \neq w} \bar{d}(\{v, w\}) y_e = \sum_{e \in \binom{T}{2}} \bar{d}(e) \sum_{C \in \mathcal{C}^k : e \in \vec{S}_C} x_C \\ &= \sum_{C \in \mathcal{C}^k} x_C \sum_{e \in \vec{S}_C} \bar{d}(e) = \sum_{C \in \mathcal{C}^k} x_C \sum_{e \in B_C} d(e). \end{aligned}$$

Der Beweis folgt nun, da $d(e) = \bar{d}(e)$ für $e \in W$. \square

Wir werden nun zeigen, dass es eine Taktik für die Kantenmarkierung (Wahl der Brückenmenge) gibt, bei der jede (noch nicht markierte) Kante von W in jeder Iteration mit Wahrscheinlichkeit mindestens $\frac{1}{M}$ markiert wird:

Lemma 20.20. *Seien (T, W) ein Baum und x eine zulässige Lösung des LP (20.1). Sei ferner $\mathcal{P} := \{(C, B) : C = (t, R) \in \mathcal{C}^k, B \in \mathcal{B}_W(R)\}$. Dann gibt es eine Zahl $M > 0$ und eine Wahrscheinlichkeitsverteilung $p : \mathcal{P} \rightarrow [0, 1]$, so dass $\sum_{B \in \mathcal{B}_W(R)} p(C, B) = \frac{x_C}{M}$ für $C = (t, R) \in \mathcal{C}^k$ und $\sum_{(C, B) \in \mathcal{P}: e \in B} p(C, B) \geq \frac{1}{M}$ für $e \in W$.*

Beweis: Setzen wir $M := \sum_{C \in \mathcal{C}^k} x_C$ und substituieren $p(C, B) = \frac{x_C}{M} q(C, B)$ für $(C, B) \in \mathcal{P}$, so ist die Behauptung äquivalent zu der Aussage, dass das folgende lineare Ungleichungssystem eine Lösung hat:

$$\begin{aligned} \sum_{B \in \mathcal{B}_W(R)} q(C, B) &\leq 1 \quad (C = (t, R) \in \mathcal{C}^k) \\ \sum_{(C, B) \in \mathcal{P}: e \in B} x_C q(C, B) &\geq 1 \quad (e \in W) \\ q(C, B) &\geq 0 \quad ((C, B) \in \mathcal{P}) \end{aligned}$$

Angenommen, das Gegenteil sei der Fall. Dann folgt nach Korollar 3.25: Es gibt ein $z \in \mathbb{R}_+^{\mathcal{C}^k}$ und ein $d \in \mathbb{R}_+^W$ mit

$$\begin{aligned} \sum_{C \in \mathcal{C}^k} z_C &< \sum_{e \in W} d_e \\ z_C &\geq x_C \sum_{e \in B} d_e \quad ((C, B) \in \mathcal{P}). \end{aligned}$$

Daraus folgt aber

$$\sum_{C \in \mathcal{C}^k} x_C \max \left\{ \sum_{e \in B} d_e : B \in \mathcal{B}_W(R) \right\} \leq \sum_{C \in \mathcal{C}^k} z_C < \sum_{e \in W} d_e,$$

im Widerspruch zum Brückenlemma 20.19. \square

Man beachte, dass wir dieses Lemma auf den Baum $(T, W_{i-1})/C_1/\dots/C_{i-1}$ in Iteration i anwenden (siehe Lemma 20.18).

Die Zahl M kann von Iteration zu Iteration variieren. Sei M zum Zwecke der hiesigen Analyse eine globale obere Schranke. Auch modifizierte man die Lösungen des LP, indem man den Wert der Variable $x_{(r, \{r\})}$ (diese Dummy-Komponente hat verschwindendes Gewicht) so weit vergrößert, dass $\sum_{C \in \mathcal{C}^k} x_C = M$ in jeder Iteration ist. Nun wird in vielen Iterationen die Dummy-Komponente gewählt (ohne etwas Weiteres zu unternehmen), was nichts am späteren Verlauf ändert. Dieser kleine Kunstgriff erlaubt es uns, mit einem einzigen Wert von M zu arbeiten.

Lemma 20.21. Sei $X \subseteq W$ und man nehme an, dass jedes Element in jeder Iteration mit Wahrscheinlichkeit mindestens $\frac{1}{M}$ markiert wird. Dann ist die erwartete Anzahl von Iterationen, bis alle Elemente von X markiert sind, höchstens gleich $MH(|X|)$, wobei $H(i) := 1 + \frac{1}{2} + \dots + \frac{1}{i}$ die i -te harmonische Zahl ist.

Beweis: Sei $\tau(q)$ (eine obere Schranke für) die erwartete Anzahl der Iterationen, bis alle Elemente einer q -elementigen Menge X markiert sind. Mit Lemma 20.20 folgt $\tau(1) \leq 1 + (1 - \frac{1}{M}) + (1 - \frac{1}{M})^2 + \dots = M$. Nun benutzen wir Induktion über q . Sei $|X| = q > 1$. Für $i = 0, \dots, q+1$ sei λ_i die Wahrscheinlichkeit, dass mindestens i Elemente von X in der ersten Iteration markiert werden. Es sind $\lambda_0 = 1$, $\lambda_{q+1} = 0$, und nach Lemma 20.20 gilt $\sum_{i=1}^q \lambda_i \geq \frac{q}{M}$. Dann haben wir

$$\begin{aligned}\tau(q) &\leq 1 + \sum_{i=0}^q (\lambda_i - \lambda_{i+1})\tau(q-i) \\ &\leq 1 + (1 - \lambda_1)\tau(q) + \sum_{i=1}^q (\lambda_i - \lambda_{i+1})MH(q-i) \\ &= 1 + (1 - \lambda_1)\tau(q) - \sum_{i=1}^q \lambda_i M \frac{1}{q-i+1} + \lambda_1 MH(q) \\ &\leq 1 + (1 - \lambda_1)\tau(q) - \sum_{i=1}^q \lambda_i M \frac{1}{q} + \lambda_1 MH(q) \\ &\leq (1 - \lambda_1)\tau(q) + \lambda_1 MH(q),\end{aligned}$$

und da $\lambda_1 > 0$, folgt $\tau(q) \leq MH(q)$. □

Der Spezialfall, bei dem genau ein Element in jeder Iteration markiert wird, ist bekannt als das Bildchensammler-Problem. Nun haben wir alles beisammen, um den folgenden Satz zu formulieren:

Satz 20.22. (Byrka et al. [2013]) Der ITERATIVE RANDOMISIERTE RUNDUNGSALGORITHMUS berechnet einen Steinerbaum Y für T mit erwarteten Kosten von höchstens ($\ln 4$) mal den Kosten eines optimalen k -beschränkten Steinerbaums. Die Wahl eines genügend großen k ergibt eine Approximationszahl von 1,39.

Beweis: Für eine Kante $e \in E(Y^*)$ des optimalen Steinerbaumes Y^* sei $\tau(e)$ die erwartete Anzahl von Iterationen, bis alle Elemente von $W(e)$ markiert sind. Da die Wahrscheinlichkeit, dass $|W(e)|$ mehr als q Elemente hat, höchstens 2^{-q} ist (für alle $q \in \mathbb{N}$), folgt mit Lemma 20.21:

$$\tau(e) \leq \sum_{q \geq 1} \text{Prob}(|W(e)| = q)MH(q) \leq \sum_{q \geq 1} 2^{-q} MH(q) = M \sum_{q \geq 1} \frac{2^{1-q}}{q} = M \ln 4.$$

Sei f_t der LP-Wert nach t Iterationen und Y_t^* der aus Y^* resultierende Wald, wenn man diejenigen Kanten e entfernt, für die alle Elemente von $W(e)$ nach t

Iterationen markiert sind. Mit Lemma 20.18 folgt $f_t \leq \rho_k \sum_{e \in E(Y_t^*)} c(e)$, wobei ρ_k wiederum der k -Steiner-Quotient ist (siehe Definition 20.8). Somit können die erwarteten Kosten des Outputs des Algorithmus folgendermaßen beschränkt werden:

$$\begin{aligned} \sum_{t \geq 1} \frac{1}{M} \text{Exp}(f_{t-1}) &\leq \sum_{t \geq 1} \frac{\rho_k}{M} \text{Exp} \left(\sum_{e \in E(Y_{t-1}^*)} c(e) \right) \\ &= \frac{\rho_k}{M} \sum_{e \in E(Y^*)} \tau(e) c(e) \\ &\leq \rho_k (\ln 4) \sum_{e \in E(Y^*)} c(e). \end{aligned}$$

Der Satz folgt nun durch die Wahl eines genügend großen k , so dass $\rho_k \leq \frac{1,39}{\ln 4}$ (nach Satz 20.9 genügt $k = 2^{375}$). \square

Goemans et al. [2012] hat bewiesen, dass man die Lösung des LP in jeder Iteration vermeiden kann, und damit gezeigt, dass das LP (20.1) eine Ganzzahligkeitslücke von höchstens $\ln 4$ hat.

20.4 Survivable-Network-Design

Bevor wir uns dem allgemeinen SURVIVABLE-NETWORK-DESIGN-PROBLEM zuwenden, gehen wir auf zwei Spezialfälle ein. Sind alle Zusammenhangsbedingungen r_{xy} gleich 0 oder 1, so heißt das Problem das STEINERWALD-PROBLEM (offensichtlich ist das STEINERBAUM-PROBLEM ein Spezialfall hiervon). Der erste Approximationsalgorithmus für das STEINERWALD-PROBLEM wurde von Agrawal, Klein und Ravi [1995] beschrieben.

Ein weiterer interessanter Spezialfall ist das Problem der Bestimmung eines k -fach kantenzusammenhängenden Teilgraphen mit minimalem Gewicht (hier gilt $r_{xy} = k$ für alle x, y). In Aufgabe 11 wird ein kombinatorischer 2-Approximationsalgorithmus für diesen Spezialfall diskutiert; siehe ferner die dort zitierten Arbeiten zu diesem Problem.

Bei der Betrachtung des allgemeinen SURVIVABLE-NETWORK-DESIGN-PROBLEMS mit gegebenen Zusammenhangsbedingungen r_{xy} für alle $x, y \in V(G)$ ist es von Vorteil, eine Funktion $f : 2^{V(G)} \rightarrow \mathbb{Z}_+$ folgendermaßen zu definieren: $f(\emptyset) := f(V(G)) := 0$ und $f(S) := \max_{x \in S, y \in V(G) \setminus S} r_{xy}$ für $\emptyset \neq S \subset V(G)$. Damit kann das vorliegende Problem als ganzzahliges LP formuliert werden:

$$\begin{aligned} \min \quad & \sum_{e \in E(G)} c(e)x_e \\ \text{bzgl.} \quad & \sum_{e \in \delta(S)} x_e \geq f(S) \quad (S \subseteq V(G)) \\ & x_e \in \{0, 1\} \quad (e \in E(G)). \end{aligned} \tag{20.2}$$

Wir werden dieses ganzzahlige LP nicht in seiner allgemeinen Form betrachten, sondern werden uns eine besondere Eigenschaft von f zunutze machen:

Definition 20.23. Eine Funktion $f : 2^U \rightarrow \mathbb{Z}_+$ heißt **brauchbar**, falls die folgenden drei Bedingungen erfüllt sind:

- für alle $S \subseteq U$ gilt $f(S) = f(U \setminus S)$;
- für alle $A, B \subseteq U$ mit $A \cap B = \emptyset$ gilt $f(A \cup B) \leq \max\{f(A), f(B)\}$;
- es gilt $f(\emptyset) = 0$.

Offensichtlich ist die weiter oben definierte Funktion f brauchbar. Brauchbare Funktionen wurden von Goemans und Williamson [1995] eingeführt; sie haben ferner einen 2-Approximationsalgorithmus für brauchbare Funktionen f mit $f(S) \in \{0, 1\}$ für alle S beschrieben. Klein und Ravi [1993] haben einen 3-Approximationsalgorithmus für brauchbare Funktionen f mit $f(S) \in \{0, 2\}$ für alle S beschrieben.

Die folgende Eigenschaft von brauchbaren Funktionen spielt eine wesentliche Rolle:

Proposition 20.24. Eine brauchbare Funktion $f : 2^U \rightarrow \mathbb{Z}_+$ ist **schwach supermodular**, d. h. es gilt mindestens eine der beiden folgenden Bedingungen für alle $A, B \subseteq U$:

- $f(A) + f(B) \leq f(A \cup B) + f(A \cap B)$.
- $f(A) + f(B) \leq f(A \setminus B) + f(B \setminus A)$.

Beweis: Definitionsgemäß haben wir

$$f(A) \leq \max\{f(A \setminus B), f(A \cap B)\}; \quad (20.3)$$

$$f(B) \leq \max\{f(B \setminus A), f(A \cap B)\}; \quad (20.4)$$

$$\begin{aligned} f(A) &= f(U \setminus A) \leq \max\{f(B \setminus A), f(U \setminus (A \cup B))\} \\ &= \max\{f(B \setminus A), f(A \cup B)\}; \end{aligned} \quad (20.5)$$

$$\begin{aligned} f(B) &= f(U \setminus B) \leq \max\{f(A \setminus B), f(U \setminus (A \cup B))\} \\ &= \max\{f(A \setminus B), f(A \cup B)\}. \end{aligned} \quad (20.6)$$

Wir unterscheiden vier Fälle, je nachdem welche der vier Zahlen $f(A \setminus B)$, $f(B \setminus A)$, $f(A \cap B)$ und $f(A \cup B)$ die kleinste ist. Ist $f(A \setminus B)$ die kleinste, so addieren wir (20.3) und (20.6). Ist $f(B \setminus A)$ die kleinste, so addieren wir (20.4) und (20.5). Ist $f(A \cap B)$ die kleinste, so addieren wir (20.3) und (20.4). Ist $f(A \cup B)$ die kleinste, so addieren wir (20.5) und (20.6). \square

Im Rest dieses Abschnitts zeigen wir, wie man die folgende LP-Relaxierung von (20.2) löst:

$$\begin{aligned} \min \quad & \sum_{e \in E(G)} c(e)x_e \\ \text{bzgl.} \quad & \sum_{e \in \delta(S)} x_e \geq f(S) \quad (S \subseteq V(G)) \\ & x_e \geq 0 \quad (e \in E(G)) \\ & x_e \leq 1 \quad (e \in E(G)). \end{aligned} \quad (20.7)$$

Wir wissen nicht, wie man dieses LP in polynomieller Zeit für beliebige Funktionen f lösen kann, nicht einmal für beliebige schwach supermodulare f . Deswegen beschränken wir uns auf brauchbare Funktionen f . Nach Satz 4.21 genügt es, das SEPARATIONS-PROBLEM zu lösen. Dazu verwenden wir einen Gomory-Hu-Baum:

Lemma 20.25. *Sei G ein ungerichteter Graph mit Kapazitäten $u \in \mathbb{R}_+^{E(G)}$ und $f : 2^{V(G)} \rightarrow \mathbb{Z}_+$ eine brauchbare Funktion. Sei H ein Gomory-Hu-Baum für (G, u) . Dann gilt für jedes $\emptyset \neq S \subset V(G)$:*

- (a) $\sum_{e' \in \delta_G(S)} u(e') \geq \max_{e \in \delta_H(S)} \sum_{e' \in \delta_G(C_e)} u(e')$,
- (b) $f(S) \leq \max_{e \in \delta_H(S)} f(C_e)$,

wobei C_e und $V(H) \setminus C_e$ zwei Zusammenhangskomponenten von $H - e$ sind.

Beweis: (a): Nach Definition des Gomory-Hu-Baumes ist $\delta_G(C_e)$ für jedes $e = \{x, y\} \in E(H)$ ein x - y -Schnitt mit minimaler Kapazität, und für $\{x, y\} \in \delta_H(S)$ ist die linke Seite von (a) die Kapazität eines gewissen x - y -Schnittes.

Zum Beweis von (b) seien X_1, \dots, X_l die Zusammenhangskomponenten von $H - S$. Da $H[X_i]$ zusammenhängend und H ein Baum ist, gilt für jedes $i \in \{1, \dots, l\}$:

$$V(H) \setminus X_i = \bigcup_{e \in \delta_H(X_i)} C_e$$

(eventuell muss C_e durch $V(H) \setminus C_e$ ersetzt werden). Da f brauchbar ist, folgt

$$f(X_i) = f(V(G) \setminus X_i) = f(V(H) \setminus X_i) = f\left(\bigcup_{e \in \delta_H(X_i)} C_e\right) \leq \max_{e \in \delta_H(X_i)} f(C_e).$$

Mit $\delta_H(X_i) \subseteq \delta_H(S)$ gilt demnach

$$f(S) = f(V(G) \setminus S) = f\left(\bigcup_{i=1}^l X_i\right) \leq \max_{i \in \{1, \dots, l\}} f(X_i) \leq \max_{e \in \delta_H(S)} f(C_e). \quad \square$$

Nun können wir zeigen, wie man das SEPARATIONS-PROBLEM für (20.7) mittels der Fundamentalschnitte eines Gomory-Hu-Baumes löst. Beachte, dass die explizite Speicherung der brauchbaren Funktion f exponentiellen Speicherplatz benötigen würde; somit nehmen wir an, dass f durch ein Orakel gegeben wird.

Satz 20.26. *Sei G ein ungerichteter Graph, $x \in \mathbb{R}_+^{E(G)}$ und $f : 2^{V(G)} \rightarrow \mathbb{Z}_+$ eine brauchbare Funktion (gegeben durch ein Orakel). Man kann in $O(n^4 + n\theta)$ -Zeit entweder eine Menge $S \subseteq V(G)$ mit $\sum_{e \in \delta_G(S)} x_e < f(S)$ finden, oder entscheiden, dass es keine solche gibt. Hier ist $n = |V(G)|$ und θ die von dem Orakel für f benötigte Zeit.*

Beweis: Zunächst berechnen wir einen Gomory-Hu-Baum H für G , mit den durch x gegebenen Kapazitäten. Nach Satz 8.38 kann H in $O(n^4)$ -Zeit berechnet werden.

Mit Lemma 20.25(b) folgt, dass es für jedes $\emptyset \neq S \subset V(G)$ ein $e \in \delta_H(S)$ mit $f(S) \leq f(C_e)$ gibt. Nach Lemma 20.25(a) gilt $f(S) - \sum_{e' \in \delta_G(S)} x_{e'} \leq f(C_e) - \sum_{e' \in \delta_G(C_e)} x_{e'}$. Damit folgt

$$\max_{\emptyset \neq S \subset V(G)} \left(f(S) - \sum_{e' \in \delta_G(S)} x_{e'} \right) = \max_{e \in E(H)} \left(f(C_e) - \sum_{e' \in \delta_G(C_e)} x_{e'} \right). \quad (20.8)$$

Demnach kann das SEPARATIONS-PROBLEM für (20.7) durch das Prüfen von nur $n - 1$ Schnitten gelöst werden. \square

Es ist lohnend, (20.8) mit Satz 12.19 zu vergleichen.

Im Gegensatz zur LP-Relaxierung (20.7) besteht keine Hoffnung, eine optimale ganzzahlige Lösung in polynomieller Zeit zu finden: Nach Satz 20.2 würde damit $P = NP$ folgen. Also werden wir Approximationsalgorithmen für (20.2) betrachten.

Im nächsten Abschnitt beschreiben wir einen primal-dualen Approximationsalgorithmus, welcher schrittweise Kanten den am meisten verletzten Schnitten hinzufügt. Dieser kombinatorische Algorithmus funktioniert gut, wenn die maximale Zusammenhangsbedingung $k := \max_{S \subseteq V(G)} f(S)$ nicht allzu groß ist. Insbesondere ist er ein 2-Approximationsalgorithmus für den Fall $k = 1$, der das STEINERWALD-PROBLEM enthält. In Abschnitt 20.6 beschreiben wir einen 2-Approximationsalgorithmus für den allgemeinen Fall. Dieser hat jedoch den Nachteil, dass er die obige Lösung der LP-Relaxierung (20.7) verwendet, die zwar polynomielle Laufzeit hat, aber in der Praxis zu ineffizient ist.

20.5 Ein primal-dualer Approximationsalgorithmus

Der in diesem Abschnitt präsentierte Algorithmus wurde schrittweise in den folgenden Arbeiten entwickelt: Williamson et al. [1995], Gabow, Goemans und Williamson [1998] und Goemans et al. [1994] in dieser Reihenfolge.

Gegeben sei ein ungerichteter Graph G mit Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ und eine brauchbare Funktion f . Ziel ist es, eine Kantenmenge F zu bestimmen, deren Inzidenzvektor (20.2) erfüllt.

Der Algorithmus läuft in $k := \max_{S \subseteq V(G)} f(S)$ Phasen ab. Da f brauchbar ist, gilt $k = \max_{v \in V(G)} f(\{v\})$; folglich kann k leicht berechnet werden. In der Phase p ($1 \leq p \leq k$) wird die brauchbare Funktion f_p mit $f_p(S) := \max\{f(S) + p - k, 0\}$ betrachtet. Nach Phase p wird der Inzidenzvektor der aktuellen Kantenmenge F das ganzzahlige LP (20.2) bezüglich f_p mit Sicherheit erfüllen. Wir beginnen mit einigen Definitionen.

Definition 20.27. Gegeben sei eine brauchbare Funktion g , ein $F \subseteq E(G)$ und $X \subseteq V(G)$. Wir sagen, dass X bezüglich (g, F) verletzt wird, falls $|\delta_F(X)| <$

$g(X)$. Die inklusionsminimalen bezüglich (g, F) verletzten Knotenmengen heißen die **aktiven Knotenmengen** bezüglich (g, F) . Die Kantenmenge $F \subseteq E(G)$ erfüllt g , falls es keine bezüglich (g, F) verletzten Knotenmengen gibt. Wir sagen: F erfüllt g **fast**, falls $|\delta_F(X)| \geq g(X) - 1$ für alle $X \subseteq V(G)$.

Während des gesamten Algorithmus wird die aktuelle Kantenmenge F die aktuelle Funktion f_p fast erfüllen. Die aktiven Knotenmengen werden eine zentrale Rolle spielen. Das folgende Resultat ist entscheidend:

Lemma 20.28. Gegeben sei eine brauchbare Funktion g , eine g fast erfüllende Kantenmenge $F \subseteq E(G)$ und zwei verletzte Knotenmengen A und B . Dann gilt: Entweder sind die zwei Mengen $A \setminus B$ und $B \setminus A$ beide verletzt, oder die zwei Mengen $A \cap B$ und $A \cup B$ sind beide verletzt. Insbesondere sind die aktiven Knotenmengen bezüglich (g, F) paarweise disjunkt.

Beweis: Der Beweis folgt direkt mittels Proposition 20.24 und Lemma 2.1(c) und (d). \square

Dieses Lemma zeigt insbesondere, dass es höchstens $n = |V(G)|$ aktive Knotenmengen gibt. Wir werden nun zeigen, wie man die aktiven Knotenmengen berechnet; wie im Beweis von Satz 20.26 verwenden wir dazu einen Gomory-Hu-Baum.

Satz 20.29. (Gabow, Goemans und Williamson [1998]) Gegeben sei eine brauchbare Funktion g (durch ein Orakel) und eine g fast erfüllende Kantenmenge $F \subseteq E(G)$. Dann können die aktiven Knotenmengen bezüglich (g, F) in $O(n^4 + n^2\theta)$ -Zeit berechnet werden. Hier ist $n = |V(G)|$ und θ die von dem Orakel für g benötigte Zeit.

Beweis: Zunächst berechnen wir einen Gomory-Hu-Baum H für $(V(G), F)$ (mit Einheitskapazitäten). Nach Satz 8.38 kann H in $O(n^4)$ -Zeit berechnet werden. Mit Lemma 20.25 folgt für jedes $\emptyset \neq S \subset V(G)$:

$$|\delta_F(S)| \geq \max_{e \in \delta_H(S)} |\delta_F(C_e)| \quad (20.9)$$

und

$$g(S) \leq \max_{e \in \delta_H(S)} g(C_e), \quad (20.10)$$

wobei C_e und $V(H) \setminus C_e$ zwei Zusammenhangskomponenten von $H - e$ sind.

Sei A eine aktive Knotenmenge. Nach (20.10) gibt es eine Kante $e = \{s, t\} \in \delta_H(A)$ mit $g(A) \leq g(C_e)$, und nach (20.9) gilt $|\delta_F(A)| \geq |\delta_F(C_e)|$. Da g von F fast erfüllt wird, folgt somit

$$1 = g(A) - |\delta_F(A)| \leq g(C_e) - |\delta_F(C_e)| \leq 1.$$

Also gilt durchweg Gleichheit, insbesondere gilt $|\delta_F(A)| = |\delta_F(C_e)|$. Damit ist $\delta_F(A)$ ein kapazitätsminimaler s - t -Schnitt in $(V(G), F)$. O. B. d. A. können wir annehmen: A enthält t , aber nicht s .

Sei G' der Digraph $(V(G), \{(v, w), (w, v) : \{v, w\} \in F\})$. Betrachte einen s - t -Fluss f mit maximalem Wert in G' und den Residualgraphen G'_f . Es gehe der azyklische Digraph G'' aus G'_f hervor durch Kontraktion der Menge S der von s aus erreichbaren Knoten zu einem einzigen Knoten v_S , Kontraktion der Menge T derjenigen Knoten, von denen aus t erreichbar ist, zu einem einzigen Knoten v_T , und Kontraktion einer jeden starken Zusammenhangskomponente X von $G'_f - (S \cup T)$ zu einem Knoten v_X . Es gibt eine Bijektion zwischen den kapazitätsminimalen s - t -Schnitten in G' und den gerichteten v_T - v_S -Schnitten in G'' (siehe Aufgabe 5, Kapitel 8; dies folgt leicht aus dem Max-Flow-Min-Cut-Theorem (Satz 8.6) und Lemma 8.3). Insbesondere ist A die Vereinigung derjenigen Mengen X , für die $v_X \in V(G'')$ gilt. Da $g(A) > |\delta_F(A)| = |\delta_{G'}^-(A)| = \text{value}(f)$ und g brauchbar ist, gibt es einen Knoten $v_X \in V(G'')$ mit $X \subseteq A$ und $g(X) > \text{value}(f)$.

Wir zeigen nun, wie man A bestimmt. Ist $g(T) > \text{value}(f)$, so setze $Z := T$; andernfalls sei v_Z irgendein Knoten von G'' mit $g(Z) > \text{value}(f)$ und $g(Y) \leq \text{value}(f)$ für alle Knoten $v_Y \in V(G'') \setminus \{v_Z\}$, von denen aus v_Z erreichbar ist.

Setze ferner

$$B := T \cup \bigcup \{Y : v_Z \text{ ist von } v_Y \text{ aus in } G'' \text{ erreichbar}\}.$$

Da

$$\begin{aligned} \text{value}(f) < g(Z) &= g(V(G) \setminus Z) \leq \max\{g(V(G) \setminus B), g(B \setminus Z)\} \\ &= \max\{g(B), g(B \setminus Z)\} \end{aligned}$$

und

$$g(B \setminus Z) \leq \max\{g(Y) : v_Y \in V(G'') \setminus \{v_Z\}, Y \subseteq B\} \leq \text{value}(f),$$

folgt $g(B) > \text{value}(f) = |\delta_{G'}^-(B)| = |\delta_F(B)|$; somit wird B bezüglich (g, F) verletzt. Es ist B keine echte Teilmenge von A (da A aktiv ist), und sowohl A als auch B enthält T ; folglich gilt $A \subseteq B$ nach Lemma 20.28. Daraus folgt nun $Z = X$, weil v_Z der einzige Knoten mit $Z \subseteq B$ und $g(Z) > \text{value}(f)$ ist und A all diejenigen Mengen Y enthält, für die v_Z von v_Y aus erreichbar ist (da $\delta_{G'_f}^-(A) = \emptyset$).

Also ist $A = B$.

Für ein gegebenes Paar (s, t) kann eine Menge B wie oben (falls es eine gibt) in linearer Zeit gefunden werden, indem man G'' (mit dem ALGORITHMUS FÜR STARKE ZUSAMMENHANGSKOMPONENTEN) konstruiert und dann eine topologische Ordnung von G'' (siehe Satz 2.20) beginnend mit v_T findet. Dieses Verfahren wiederholen wir für jedes geordnete Paar (s, t) mit $\{s, t\} \in E(H)$.

Auf diese Weise erhalten wir eine Liste mit höchstens $2n - 2$ Kandidaten für aktive Knotenmengen. Die Laufzeit wird offensichtlich durch die $O(n)$ malige Bestimmung eines maximalen Flusses in G' und den $O(n^2)$ maligen Anruf des Orakels für g dominiert. Abschließend entfernen wir in $O(n^2)$ -Zeit diejenigen verletzten Knotenmengen unter den Kandidaten, die nicht inklusionsminimal sind. \square

Die Laufzeit kann verbessert werden, wenn $\max_{S \subseteq V(G)} g(S)$ klein ist (siehe Aufgabe 13). Wir wenden uns nun der Beschreibung des Algorithmus zu.

PRIMAL-DUALER ALGORITHMUS FÜR NETZWERK-DESIGN

Input: Ein ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$ und ein Orakel für eine brauchbare Funktion $f : 2^{V(G)} \rightarrow \mathbb{Z}_+$.
Output: Eine f erfüllende Menge $F \subseteq E(G)$.

- ① **If** $E(G)$ erfüllt f nicht **then stop** (das Problem ist unzulässig).
- ② Setze $F := \emptyset$, $k := \max_{v \in V(G)} f(\{v\})$ und $p := 1$.
- ③ Setze $i := 0$.
 Setze $\pi(v) := 0$ für alle $v \in V(G)$.
 Sei \mathcal{A} die Familie der aktiven Knotenmengen bezüglich (F, f_p) mit
 $f_p(S) := \max\{f(S) + p - k, 0\}$ für alle $S \subseteq V(G)$.
- ④ **While** $\mathcal{A} \neq \emptyset$ **do**:
 - Setze $i := i + 1$.
 - Setze $\epsilon := \min \left\{ \frac{c(e) - \pi(v) - \pi(w)}{|\{A \in \mathcal{A} : e \in \delta_G(A)\}|} : e = \{v, w\} \in \bigcup_{A \in \mathcal{A}} \delta_G(A) \setminus F \right\}$,
 - und sei e_i eine dieses Minimum annehmende Kante.
 - Erhöhe $\pi(v)$ um ϵ für alle $v \in \bigcup_{A \in \mathcal{A}} A$.
 - Setze $F := F \cup \{e_i\}$.
 - Aktualisiere \mathcal{A} .
- ⑤ **For** $j := i$ **down to** 1 **do**:
 - If** $F \setminus \{e_j\}$ erfüllt f_p **then** setze $F := F \setminus \{e_j\}$.
- ⑥ **If** $p = k$ **then stop**, **else** setze $p := p + 1$ und **go to** ③.

Die Zulässigkeitsprüfung in ① benötigt nach Satz 20.26 $O(n^4 + n\theta)$ -Zeit. Bevor wir die Implementierung von ③ und ④ erörtern, zeigen wir, dass der Output F in der Tat bezüglich f zulässig ist. Es bezeichne F_p die Menge F am Ende von Phase p (und $F_0 := \emptyset$).

Lemma 20.30. Zu jedem Zeitpunkt der Phase p wird f_p von F fast erfüllt und $F \setminus F_{p-1}$ ist ein Wald. Am Ende von Phase p wird f_p von F_p erfüllt.

Beweis: Es gilt $f_1(S) = \max\{0, f(S) + 1 - k\} \leq \max\{0, \max_{v \in S} f(\{v\}) + 1 - k\} \leq 1$ (da f brauchbar ist). Somit wird f_1 von der leeren Menge fast erfüllt.

Nach ④ gibt es keine aktiven Knotenmengen mehr, also wird f_p von F erfüllt. In ⑤ bleibt diese Eigenschaft explizit erhalten. Somit wird jedes f_p von F_p erfüllt, also werden die f_{p+1} mit $p = 0, \dots, k-1$ von jedem F_p fast erfüllt. Um zu sehen, dass $F \setminus F_{p-1}$ ein Wald ist, beachte man: Jede zu F hinzugefügte Kante liegt in $\delta(A)$ für irgendeine aktive Knotenmenge A und muss die erste zu F in dieser Phase hinzugefügte Kante von $\delta(A)$ sein (da $|\delta_{F_{p-1}}(A)| = f_{p-1}(A)$). Somit bildet keine Kante einen Kreis in $F \setminus F_{p-1}$. \square

Demnach kann Satz 20.29 zur Bestimmung von \mathcal{A} angewendet werden. Die Anzahl der in jeder Phase auftretenden Iterationen ist höchstens gleich $n - 1$. Die einzige noch zu besprechende Implementierungsfrage betrifft die Bestimmung von ϵ und e_i in ④.

Lemma 20.31. *Die Bestimmung von ϵ und e_i in ④ des Algorithmus kann in $O(mn)$ -Zeit pro Phase bewerkstelligt werden.*

Beweis: Bei jeder Iteration einer Phase führen wir Folgendes aus. Zunächst ordnen wir jedem Knoten eine Zahl zu, entsprechend der aktiven Knotenmenge, der er angehört (oder Null, falls er keiner angehört). Dies kann in $O(n)$ -Zeit erledigt werden (beachte, dass die aktiven Knotenmengen nach Lemma 20.28 paarweise disjunkt sind). Für jede Kante e kann jetzt die Anzahl derjenigen aktiven Knotenmengen, die genau einen Endknoten von e enthalten, in $O(1)$ -Zeit bestimmt werden. Somit können ϵ und e_i in $O(m)$ -Zeit bestimmt werden. Da es höchstens $n - 1$ Iterationen pro Phase gibt, ist die Zeitschranke bewiesen. \square

Wir weisen hier darauf hin, dass Gabow, Goemans und Williamson [1998] diese Schranke mit einer ausgeklügelten Implementierung auf $O\left(n^2 \sqrt{\log \log n}\right)$ verbessert haben.

Satz 20.32. (Goemans et al. [1994]) *Der PRIMAL-DUALE ALGORITHMUS FÜR NETZWERK-DESIGN liefert als Output eine f erfüllende Kantenmenge F in $O\left(kn^5 + kn^3\theta\right)$ -Zeit, wobei $k = \max_{S \subseteq V(G)} f(S)$, $n = |V(G)|$ und θ die von dem Orakel für f benötigte Zeit ist.*

Beweis: Die Zulässigkeit von F ist durch Lemma 20.30 gewährleistet, da $f_k = f$.

Ein Orakel für jedes f_p gebraucht natürlich das Orakel für f und benötigt somit $\theta + O(1)$ -Zeit. Die Berechnung der aktiven Knotenmengen benötigt $O\left(n^4 + n^2\theta\right)$ -Zeit (Satz 20.29) und erfolgt $O(nk)$ mal. Die Bestimmung von ϵ und e_i kann in $O(n^3)$ Zeit pro Phase erledigt werden (Lemma 20.31). Der Rest kann leicht in $O(kn^2)$ -Zeit bewerkstelligt werden. \square

Aufgabe 13 zeigt, wie die Laufzeit auf $O\left(k^3n^3 + kn^3\theta\right)$ verbessert werden kann. Sie kann auf $O\left(k^2n^3 + kn^2\theta\right)$ verbessert werden, indem man einen anderen Aufräumschritt (⑤ des Algorithmus) und eine raffiniertere Implementierung benutzt (Gabow, Goemans und Williamson [1998]). Für festes k und $\theta = O(n)$ bedeutet dies, dass wir einen $O\left(n^3\right)$ -Algorithmus haben. Für den Spezialfall des SURVIVABLE-NETWORK-DESIGN-PROBLEMS (es wird f durch Zusammenhangsbedingungen r_{xy} bestimmt) kann die Laufzeit auf $O\left(k^2n^2 \sqrt{\log \log n}\right)$ verbessert werden.

Nun analysieren wir die Approximationsgüte des Algorithmus und rechtfertigen die Tatsache, dass wir ihn einen primal-dualen Algorithmus genannt haben. Das duale LP von (20.7) ist

$$\begin{aligned}
 \max \quad & \sum_{S \subseteq V(G)} f(S) y_S - \sum_{e \in E(G)} z_e \\
 \text{bzgl.} \quad & \sum_{S: e \in \delta(S)} y_S \leq c(e) + z_e \quad (e \in E(G)) \quad (20.11) \\
 & y_S \geq 0 \quad (S \subseteq V(G)) \\
 & z_e \geq 0 \quad (e \in E(G)).
 \end{aligned}$$

Dieses duale LP spielt eine entscheidende Rolle in der Analyse des Algorithmus.

Wir zeigen nun, wie der Algorithmus in jeder Phase p implizit eine zulässige duale Lösung $y^{(p)}$ konstruiert. Beginnend mit $y^{(p)} = 0$, wird $y_A^{(p)}$ in jeder Iteration dieser Phase für jedes $A \in \mathcal{A}$ um ϵ erhöht. Ferner setzen wir

$$z_e^{(p)} := \begin{cases} \sum_{S: e \in \delta(S)} y_S^{(p)} & \text{if } e \in F_{p-1} \\ 0 & \text{sonst.} \end{cases}$$

Es ist nicht sinnvoll, diese duale Lösung explizit im Algorithmus zu konstruieren. Die Variablen $\pi(v) = \sum_{S: v \in S} y_S$ ($v \in V(G)$) enthalten alle erforderlichen Informationen.

Lemma 20.33. (Williamson et al. [1995]) *Es ist $(y^{(p)}, z^{(p)})$ (wie oben definiert) für jedes p eine zulässige Lösung von (20.11).*

Beweis: Die Nichtnegativitätsbedingungen gelten offensichtlich. Nach Definition von $z_e^{(p)}$ sind die Nebenbedingungen für $e \in F_{p-1}$ erfüllt.

Ferner haben wir nach ④ des Algorithmus:

$$\sum_{S: e \in \delta(S)} y_S^{(p)} \leq c(e) \quad \text{für jedes } e \in E(G) \setminus F_{p-1},$$

weil e , wenn Gleichheit erreicht worden ist, zu F hinzugefügt wird und danach die Mengen S mit $e \in \delta(S)$ nicht mehr bezüglich (F, f_p) verletzt werden (wir erinnern daran, dass f_p nach Lemma 20.30 von $F \setminus \{e\} \supseteq F_{p-1}$ fast erfüllt wird). \square

Es bezeichne $\text{OPT}(G, c, f)$ den optimalen Zielfunktionswert des ganzzahligen LP (20.2). Als Nächstes beweisen wir das folgende

Lemma 20.34. (Goemans et al. [1994]) *Für jedes $p \in \{1, \dots, k\}$ haben wir*

$$\sum_{S \subseteq V(G)} y_S^{(p)} \leq \frac{1}{k-p+1} \text{OPT}(G, c, f).$$

Beweis: Es ist $\text{OPT}(G, c, f)$ größer oder gleich dem optimalen Zielfunktionswert der LP-Relaxierung (20.7), und Letzterer ist von unten beschränkt durch den Zielfunktionswert einer jeden zulässigen dualen Lösung (nach dem Dualitätssatz (Satz 3.20)). Da $(y^{(p)}, z^{(p)})$ nach Lemma 20.33 für das duale LP (20.11) zulässig ist, haben wir

$$\text{OPT}(G, c, f) \geq \sum_{S \subseteq V(G)} f(S) y_S^{(p)} - \sum_{e \in E(G)} z_e^{(p)}.$$

Beachte nun: Für jedes $S \subseteq V(G)$ kann y_S nur dann positiv werden, wenn S bezüglich (f_p, F_{p-1}) verletzt wird. Daraus folgt, dass

$$y_S^{(p)} > 0 \Rightarrow |\delta_{F_{p-1}}(S)| \leq f(S) + p - k - 1.$$

Demnach erhalten wir

$$\begin{aligned} \text{OPT}(G, c, f) &\geq \sum_{S \subseteq V(G)} f(S) y_S^{(p)} - \sum_{e \in E(G)} z_e^{(p)} \\ &= \sum_{S \subseteq V(G)} f(S) y_S^{(p)} - \sum_{e \in F_{p-1}} \left(\sum_{S: e \in \delta(S)} y_S^{(p)} \right) \\ &= \sum_{S \subseteq V(G)} f(S) y_S^{(p)} - \sum_{S \subseteq V(G)} |\delta_{F_{p-1}}(S)| y_S^{(p)} \\ &= \sum_{S \subseteq V(G)} (f(S) - |\delta_{F_{p-1}}(S)|) y_S^{(p)} \\ &\geq \sum_{S \subseteq V(G)} (k - p + 1) y_S^{(p)}. \end{aligned}$$

□

Lemma 20.35. (Williamson et al. [1995]) Bei jeder Iteration einer beliebigen Phase p gilt

$$\sum_{A \in \mathcal{A}} |\delta_{F_p \setminus F_{p-1}}(A)| \leq 2|\mathcal{A}|.$$

Beweis: Wir betrachten eine bestimmte Iteration der Phase p , die wir die aktuelle Iteration nennen. Sei \mathcal{A} die Familie der aktiven Knotenmengen am Anfang dieser Iteration und setze

$$H := (F_p \setminus F_{p-1}) \cap \bigcup_{A \in \mathcal{A}} \delta(A).$$

Beachte, dass alle Kanten von H während oder nach der aktuellen Iteration hinzugefügt worden sein müssen.

Sei $e \in H$. Es wird f_p nicht von $F_p \setminus \{e\}$ erfüllt, da e sonst während des Aufräumschrittes ⑤ der Phase p entfernt worden wäre. Sei also X_e eine kardinalitätsminimale bezüglich $(f_p, F_p \setminus \{e\})$ verletzte Knotenmenge. Da f_p von $F_p \setminus \{e\} \supseteq F_{p-1}$ fast erfüllt wird, folgt $\delta_{F_p \setminus F_{p-1}}(X_e) = \{e\}$.

Wir behaupten nun, dass die Familie $\mathcal{X} := \{X_e : e \in H\}$ laminar ist. Angenommen, es gäbe zwei Kanten $e, e' \in H$ (und es wurde e vor e' hinzugefügt), für welche $X_e \setminus X_{e'}$, $X_{e'} \setminus X_e$ und $X_e \cap X_{e'}$ sämtlich nichtleere Knotenmengen sind. Da X_e und $X_{e'}$ am Anfang der aktuellen Iteration verletzt werden, gilt nach Lemma 20.28: Entweder sind $X_e \cup X_{e'}$ und $X_e \cap X_{e'}$ beide verletzt, oder $X_e \setminus X_{e'}$ und $X_{e'} \setminus X_e$ sind beide verletzt. Der erste Fall ergibt

$$\begin{aligned} 1 + 1 &\leq |\delta_{F_p \setminus F_{p-1}}(X_e \cup X_{e'})| + |\delta_{F_p \setminus F_{p-1}}(X_e \cap X_{e'})| \\ &\leq |\delta_{F_p \setminus F_{p-1}}(X_e)| + |\delta_{F_p \setminus F_{p-1}}(X_{e'})| = 1 + 1 \end{aligned}$$

mit der Submodularität von $|\delta_{F_p \setminus F_{p-1}}|$ (Lemma 2.1(c)). Daraus folgt $|\delta_{F_p \setminus F_{p-1}}(X_e \cup X_{e'})| = |\delta_{F_p \setminus F_{p-1}}(X_e \cap X_{e'})| = 1$, im Widerspruch zur minimalen Wahl von X_e oder von $X_{e'}$, falls stattdessen $X_e \cap X_{e'}$ gewählt wurde.

Liegt der zweite Fall vor, so folgt mit Lemma 2.1(d): $|\delta_{F_p \setminus F_{p-1}}(X_e \setminus X_{e'})| = |\delta_{F_p \setminus F_{p-1}}(X_{e'} \setminus X_e)| = 1$. Die kleinere der beiden Mengen $X_e \setminus X_{e'}$ und $X_{e'} \setminus X_e$ widerspricht nun der minimalen Wahl von X_e oder $X_{e'}$.

Jetzt betrachten wir eine Baumdarstellung (T, φ) von \mathcal{X} , wobei T eine Arboreszenz ist (siehe Proposition 2.14). Für jedes $e \in H$ wird X_e am Anfang der aktuellen Iteration verletzt, da e zu dem Zeitpunkt noch nicht hinzugefügt wurde. Damit folgt mit Lemma 20.28, dass $A \subseteq X_e$ oder $A \cap X_e = \emptyset$ für alle $A \in \mathcal{A}$. Demnach enthält $\{\varphi(a) : a \in A\}$ für jedes $A \in \mathcal{A}$ nur ein Element, welches wir mit $\varphi(A)$ bezeichnen. Wir nennen einen Knoten $v \in V(T)$ **besetzt**, falls $v = \varphi(A)$ für irgendein $A \in \mathcal{A}$.

Wir behaupten, dass alle Knoten von T mit Ausgangsgrad 0 besetzt sind. Für solch einen Knoten v ist nämlich $\varphi^{-1}(v)$ ein inklusionsminimales Element von \mathcal{X} . Ein inklusionsminimales Element von \mathcal{X} wird aber am Anfang der aktuellen Iteration verletzt, enthält somit eine aktive Knotenmenge und ist demnach besetzt. Also ist der durchschnittliche Ausgangsgrad der besetzten Knoten kleiner als 1.

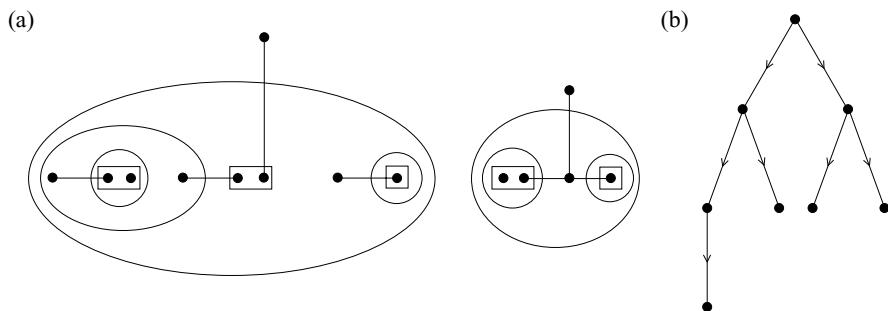


Abbildung 20.4.

Beachte, dass es Bijektionen zwischen H , \mathcal{X} und $E(T)$ gibt (siehe Abb. 20.4; (a) zeigt H , wobei die Elemente von \mathcal{A} als Rechtecke und die Elemente von \mathcal{X} als Ellipsen dargestellt sind; (b) zeigt T). Damit folgt für jedes $v \in V(T)$:

$$|\delta_T(v)| = |\delta_H(\{x \in V(G) : \varphi(x) = v\})| \geq \sum_{A \in \mathcal{A}: \varphi(A)=v} |\delta_{F_p \setminus F_{p-1}}(A)|.$$

Summieren wir über alle besetzten Knoten, so erhalten wir

$$\begin{aligned} \sum_{A \in \mathcal{A}} |\delta_{F_p \setminus F_{p-1}}(A)| &\leq \sum_{v \in V(T) \text{ besetzt}} |\delta_T(v)| \\ &< 2 |\{v \in V(T) : v \text{ besetzt}\}| \\ &\leq 2 |\mathcal{A}|. \end{aligned}$$

□

Der Beweis des nächsten Lemmas erläutert die Rolle der Bedingungen des komplementären Schlupfes:

Lemma 20.36. (Williamson et al. [1995]) *Für jedes $p \in \{1, \dots, k\}$ gilt*

$$\sum_{e \in F_p \setminus F_{p-1}} c(e) \leq 2 \sum_{S \subseteq V(G)} y_S^{(p)}.$$

Beweis: In jeder Phase p des Algorithmus bleiben die primalen Bedingungen des komplementären Schlupfes erhalten:

$$e \in F \setminus F_{p-1} \Rightarrow \sum_{S: e \in \delta(S)} y_S^{(p)} = c(e).$$

Somit haben wir

$$\sum_{e \in F_p \setminus F_{p-1}} c(e) = \sum_{e \in F_p \setminus F_{p-1}} \left(\sum_{S: e \in \delta(S)} y_S^{(p)} \right) = \sum_{S \subseteq V(G)} y_S^{(p)} |\delta_{F_p \setminus F_{p-1}}(S)|.$$

Es bleibt nur noch zu zeigen, dass

$$\sum_{S \subseteq V(G)} y_S^{(p)} |\delta_{F_p \setminus F_{p-1}}(S)| \leq 2 \sum_{S \subseteq V(G)} y_S^{(p)}. \quad (20.12)$$

Am Anfang der Phase p haben wir $y^{(p)} = 0$, somit gilt (20.12). Bei jeder Iteration wird die linke Seite um $\sum_{A \in \mathcal{A}} \epsilon |\delta_{F_p \setminus F_{p-1}}(A)|$ erhöht, während die rechte Seite um $2\epsilon |\mathcal{A}|$ erhöht wird. Mit Lemma 20.35 folgt dann, dass (20.12) nicht verletzt wird. □

In (20.12) kommen die dualen Bedingungen des komplementären Schlupfes

$$y_S^{(p)} > 0 \Rightarrow |\delta_{F_p}(S)| = f_p(S)$$

vor. Es gilt $|\delta_{F_p}(S)| \geq f_p(S)$ durchweg, während (20.12) grob gesagt bedeutet, dass $|\delta_{F_p}(S)| \leq 2f_p(S)$ im Durchschnitt gilt. Wie wir sehen werden, folgt hieraus die Approximationsgüte 2 für den Fall $k = 1$.

Satz 20.37. (Goemans et al. [1994]) Der PRIMAL-DUALE ALGORITHMUS FÜR NETZWERK-DESIGN liefert eine Kantenmenge F als Output, die f erfüllt und deren Gewicht höchstens gleich $2H(k) \text{OPT}(G, c, f)$ ist, und läuft in $O(kn^5 + kn^3\theta)$ -Zeit, wobei $n = |V(G)|$, $k = \max_{S \subseteq V(G)} f(S)$, $H(k) = 1 + \frac{1}{2} + \dots + \frac{1}{k}$ und θ die von dem Orakel für f benötigte Zeit ist.

Beweis: Die Korrektheit und die Laufzeit sind in Satz 20.32 bewiesen worden. Für das Gewicht von F haben wir wegen Lemma 20.36 und Lemma 20.34:

$$\begin{aligned}\sum_{e \in F} c(e) &= \sum_{p=1}^k \left(\sum_{e \in F_p \setminus F_{p-1}} c(e) \right) \\ &\leq \sum_{p=1}^k \left(2 \sum_{S \subseteq V(G)} y_S^{(p)} \right) \\ &\leq 2 \sum_{p=1}^k \frac{1}{k-p+1} \text{OPT}(G, c, f) \\ &= 2H(k) \text{OPT}(G, c, f).\end{aligned}$$

□

Der in diesem Abschnitt besprochene primal-duale Approximationsalgorithmus ist von Bertsimas und Teo [1995] in einem allgemeineren Rahmen präsentiert worden. Ein verwandtes, aber vermutlich schwierigeres Problem ergibt sich, wenn wir Knoten- statt Kantenzusammenhang betrachten (Ziel ist es hier, einen Teilgraphen zu finden, der mindestens eine angegebene Anzahl r_{ij} von intern disjunkten i - j -Wegen für jedes i und j enthält). Siehe auch die Bemerkungen am Ende des nächsten Abschnitts.

20.6 Jains Algorithmus

In diesem Abschnitt besprechen wir Jains [2001] 2-Approximationsalgorithmus für das SURVIVABLE-NETWORK-DESIGN-PROBLEM. Obwohl er eine viel bessere Approximationsgüte als der PRIMAL-DUALE ALGORITHMUS FÜR NETZWERK-DESIGN hat, ist er weniger relevant für die Praxis, da er auf der Äquivalenz von Optimierung und Separation basiert (siehe Abschnitt 4.6).

Der Algorithmus beginnt mit der Lösung der LP-Relaxierung (20.7). Tatsächlich bringen ganzzahlige Kantenkapazitäten $u : E(G) \rightarrow \mathbb{N}$ keine zusätzlichen Schwierigkeiten mit sich, d.h. wir können Kanten durchaus mehr als einmal wählen:

$$\begin{aligned}
 \min \quad & \sum_{e \in E(G)} c(e)x_e \\
 \text{bzgl.} \quad & \sum_{e \in \delta(S)} x_e \geq f(S) \quad (S \subseteq V(G)) \\
 & x_e \geq 0 \quad (e \in E(G)) \\
 & x_e \leq u(e) \quad (e \in E(G)).
 \end{aligned} \tag{20.13}$$

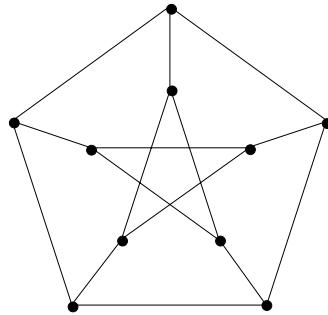


Abbildung 20.5.

Natürlich suchen wir im Grunde eine ganzzahlige Lösung. Lösen wir die LP-Relaxierung eines ganzzahligen LP und runden danach auf, so erhalten wir einen 2-Approximationsalgorithmus, falls die LP-Relaxierung immer eine halbganzzahlige optimale Lösung hat (Aufgabe 6, Kapitel 16, enthält ein Beispiel).

Die LP-Relaxierung (20.13) hat jedoch nicht diese Eigenschaft. Um dies zu sehen, betrachten wir den Petersen-Graph (Abb. 20.5) mit $u(e) = c(e) = 1$ für alle Kanten e und $f(S) = 1$ für alle $\emptyset \neq S \subset V(G)$. Hier ist der optimale Zielfunktionswert des LP (20.13) gleich 5 ($x_e = \frac{1}{3}$ für alle e ist ein optimaler Lösungsvektor), und für jeden zulässigen Vektor mit Zielfunktionswert 5 haben wir $\sum_{e \in \delta(v)} x_e = 1$ für alle $v \in V(G)$. Demnach hat ein optimaler halbganzzahliger Lösungsvektor $x_e = \frac{1}{2}$ für die Kanten e eines Hamilton-Kreises und $x_e = 0$ sonst. Der Petersen-Graph ist aber nicht hamiltonsch.

Trotzdem liefert die Lösung der LP-Relaxierung (20.13) einen 2-Approximationsalgorithmus. Entscheidend ist, dass es für jede optimale Basislösung x eine Kante e mit $x_e \geq \frac{1}{2}$ gibt (Satz 20.39). Der Algorithmus wird dann nur diese Komponenten aufrunden und fest setzen, um dann das übrig bleibende Problem zu betrachten, das mindestens eine Kante weniger hat.

Wir benötigen einige Vorbereitungen. Für eine Menge $S \subseteq V(G)$ sei χ^S der Inzidenzvektor von $\delta_G(S)$ bezüglich $E(G)$. Für einen zulässigen Vektor x von (20.13) heißt eine Menge $S \subseteq V(G)$ **straff**, falls $\chi^S x = f(S)$.

Lemma 20.38. (Jain [2001]) *Sei G ein Graph, $m := |E(G)|$ und $f : 2^{V(G)} \rightarrow \mathbb{Z}_+$ eine schwach supermodulare Funktion. Sei ferner x eine Basislösung des LP*

(20.13). Angenommen, es gelte $0 < x_e < 1$ für jedes $e \in E(G)$. Dann gibt es eine laminare Familie \mathcal{B} von m straffen Teilmengen von $V(G)$, so dass die Vektoren χ^B mit $B \in \mathcal{B}$ linear unabhängig in $\mathbb{R}^{E(G)}$ sind.

Beweis: Sei \mathcal{B} eine laminare Familie von straffen Teilmengen von $V(G)$, so dass die Vektoren χ^B mit $B \in \mathcal{B}$ linear unabhängig sind. Es sei $|\mathcal{B}| < m$; wir zeigen nun, wie man \mathcal{B} erweitert.

Da x eine Basislösung von (20.13) ist, d.h. eine Ecke des Polytops, gibt es m linear unabhängige Ungleichungsnebenbedingungen, die mit Gleichheit erfüllt sind (Proposition 3.9). Da $0 < x_e < 1$ für jedes $e \in E(G)$, entsprechen diese Nebenbedingungen einer Familie \mathcal{S} (die nicht notwendigerweise laminar ist) von m straffen Teilmengen von $V(G)$, so dass die Vektoren χ^S ($S \in \mathcal{S}$) linear unabhängig sind. Da $|\mathcal{B}| < m$, gibt es eine straffe Menge $S \subseteq V(G)$, so dass die Vektoren χ^B mit $B \in \mathcal{B} \cup \{S\}$ linear unabhängig sind. Wähle S , so dass

$$\gamma(S) := |\{B \in \mathcal{B} : B \text{ kreuzt } S\}|$$

minimal ist, wobei wir sagen: B kreuzt S falls $B \cap S \neq \emptyset$ und $B \setminus S \neq \emptyset$ und $S \setminus B \neq \emptyset$.

Ist $\gamma(S) = 0$, so können wir S der Familie \mathcal{B} hinzufügen und wir sind fertig. Angenommen, es sei $\gamma(S) > 0$ und es sei $B \in \mathcal{B}$ eine S kreuzende Menge. Da f schwach supermodular ist, haben wir

$$\begin{aligned} f(S \setminus B) + f(B \setminus S) &\geq f(S) + f(B) \\ &= \sum_{e \in \delta_G(S)} x_e + \sum_{e \in \delta_G(B)} x_e \\ &= \sum_{e \in \delta_G(S \setminus B)} x_e + \sum_{e \in \delta_G(B \setminus S)} x_e + 2 \sum_{e \in E_G(S \cap B, V(G) \setminus (S \cup B))} x_e, \end{aligned}$$

oder

$$\begin{aligned} f(S \cap B) + f(S \cup B) &\geq f(S) + f(B) \\ &= \sum_{e \in \delta_G(S)} x_e + \sum_{e \in \delta_G(B)} x_e \\ &= \sum_{e \in \delta_G(S \cap B)} x_e + \sum_{e \in \delta_G(S \cup B)} x_e + 2 \sum_{e \in E_G(S \setminus B, B \setminus S)} x_e. \end{aligned}$$

Im ersten Fall oben sind $S \setminus B$ und $B \setminus S$ beide straff und es ist $E_G(S \cap B, V(G) \setminus (S \cup B)) = \emptyset$. Damit folgt $\chi^{S \setminus B} + \chi^{B \setminus S} = \chi^S + \chi^B$. Im zweiten Fall sind $S \cap B$ und $S \cup B$ beide straff und es ist $E_G(S \setminus B, B \setminus S) = \emptyset$. Damit folgt $\chi^{S \cap B} + \chi^{S \cup B} = \chi^S + \chi^B$.

Demnach gibt es mindestens eine Menge T unter den vier Mengen $S \setminus B$, $B \setminus S$, $S \cap B$ und $S \cup B$, die straff ist und die Eigenschaft hat, dass die Vektoren χ^B mit $B \in \mathcal{B} \cup \{T\}$ linear unabhängig sind. Abschließend zeigen wir, dass $\gamma(T) < \gamma(S)$, im Widerspruch zur Wahl von S . Somit ist $\gamma(S) = 0$ und wir sind fertig.

Die Menge B kreuzt S , aber nicht T , also genügt es zu zeigen, dass es kein $C \in \mathcal{B}$ gibt, welches T aber nicht S kreuzt. Da T eine der vier Mengen $S \setminus B$,

$B \setminus S$, $S \cap B$ und $S \cup B$ ist, gilt für jede T kreuzende aber S nicht kreuzende Menge C , dass sie B kreuzt. Da \mathcal{B} laminar und $B \in \mathcal{B}$ ist, folgt $C \notin \mathcal{B}$. \square

Wir sind nun in der Lage, den Schlüsselsatz für JAINS ALGORITHMUS zu beweisen.

Satz 20.39. (Jain [2001]) *Sei G ein Graph und $f : 2^{V(G)} \rightarrow \mathbb{Z}_+$ eine schwach supermodulare Funktion, die nicht identisch Null ist. Sei x eine Basislösung des LP (20.13). Dann gibt es eine Kante $e \in E(G)$ mit $x_e \geq \frac{1}{2}$.*

Beweis: Wir können annehmen, dass $x_e > 0$ für jede Kante e , da wir e sonst entfernen können. Wir nehmen nun an, es sei $0 < x_e < \frac{1}{2}$ für alle $e \in E(G)$, und leiten hieraus einen Widerspruch ab.

Nach Lemma 20.38 gibt es eine laminare Familie \mathcal{B} von $m := |E(G)|$ straffen Teilmengen von $V(G)$, so dass die Vektoren χ^B mit $B \in \mathcal{B}$ linear unabhängig sind. Aus der linearen Unabhängigkeit folgt insbesondere, dass keine der χ^B Nullvektoren sind, also gilt $0 < \chi^B x = f(B)$ und somit ist $f(B) \geq 1$ für alle $B \in \mathcal{B}$. Ferner gilt $\bigcup_{B \in \mathcal{B}} \delta_G(B) = E(G)$. Mit der Annahme, dass $x_e < \frac{1}{2}$ für alle $e \in E(G)$, folgt $|\delta_G(B)| \geq 2f(B) + 1 \geq 3$ für alle $B \in \mathcal{B}$.

Sei (T, φ) eine Baumdarstellung von \mathcal{B} . Für jeden Knoten t der Arboreszenz T bezeichne T_t den maximalen Teilgraphen von T , der eine Arboreszenz mit Wurzel t ist (T_t enthält t und alle Nachfolger von t). Ferner sei $B_t := \{v \in V(G) : \varphi(v) \in V(T_t)\}$. Nach Definition der Baumdarstellung folgt $B_r = V(G)$ für die Wurzel r von T und $\mathcal{B} = \{B_t : t \in V(T) \setminus \{r\}\}$.

Behauptung: Für jedes $t \in V(T)$ gilt $\sum_{v \in B_t} |\delta_G(v)| \geq 2|V(T_t)| + 1$, mit Gleichheit nur dann, wenn $|\delta_G(B_t)| = 2f(B_t) + 1$.

Wir beweisen die Behauptung mittels Induktion über $|V(T_t)|$. Ist $\delta_T^+(t) = \emptyset$ (d.h. $V(T_t) = \{t\}$), so ist B_t ein inklusionsminimales Element von \mathcal{B} und somit gilt $\sum_{v \in B_t} |\delta_G(v)| = |\delta_G(B_t)| \geq 3 = 2|V(T_t)| + 1$, mit Gleichheit nur dann, wenn $|\delta_G(B_t)| = 3$ (woraus $f(B_t) = 1$ folgt).

Für den Induktionsschritt sei $t \in V(T)$ mit $\delta_T^+(t) \neq \emptyset$, etwa $\delta_T^+(t) = \{(t, s_1), \dots, (t, s_k)\}$, wobei k die Anzahl der Kinder von t ist. Setze $E_1 := \bigcup_{i=1}^k \delta_G(B_{s_i}) \setminus \delta_G(B_t)$ und $E_2 := \delta_G(B_t \setminus \bigcup_{i=1}^k B_{s_i})$ (siehe Beispiel in Abb. 20.6).

Beachte, dass $E_1 \cup E_2 \neq \emptyset$, da anderenfalls $\chi^{B_t} = \sum_{i=1}^k \chi^{B_{s_i}}$, im Widerspruch zur Annahme, dass die Vektoren χ^B mit $B \in \mathcal{B}$ linear unabhängig sind (entweder ist $B_t \in \mathcal{B}$ oder $t = r$ und somit $\chi^{B_t} = 0$). Ferner haben wir

$$|\delta_G(B_t)| + 2|E_1| = \sum_{i=1}^k |\delta_G(B_{s_i})| + |E_2| \quad (20.14)$$

und, da B_{s_1}, \dots, B_{s_k} und B_t straff sind,

$$f(B_t) + 2 \sum_{e \in E_1} x_e = \sum_{i=1}^k f(B_{s_i}) + \sum_{e \in E_2} x_e. \quad (20.15)$$

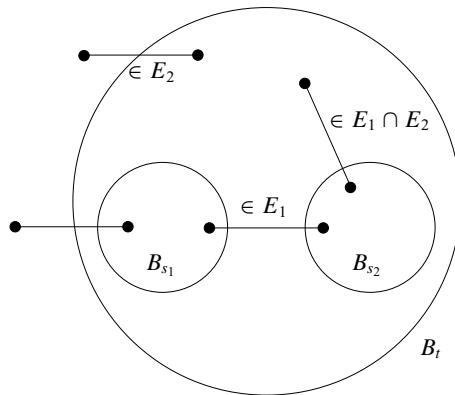


Abbildung 20.6.

Auch gilt nach der Induktionsvoraussetzung:

$$\begin{aligned}
 \sum_{v \in B_t} |\delta_G(v)| &\geq \sum_{i=1}^k \sum_{v \in B_{s_i}} |\delta_G(v)| + |E_2| \\
 &\geq \sum_{i=1}^k (2|V(T_{s_i})| + 1) + |E_2| \\
 &= 2|V(T_t)| - 2 + k + |E_2|.
 \end{aligned} \tag{20.16}$$

Nun unterscheiden wir drei Fälle.

Fall 1: Es gilt $k + |E_2| \geq 3$. Nach (20.16) folgt dann

$$\sum_{v \in B_t} |\delta_G(v)| \geq 2|V(T_t)| + 1,$$

mit Gleichheit nur dann, wenn $k + |E_2| = 3$ und $|\delta_G(B_{s_i})| = 2f(B_{s_i}) + 1$ für $i = 1, \dots, k$. Hier müssen wir zeigen, dass $|\delta_G(B_t)| = 2f(B_t) + 1$.

Mit (20.14) folgt

$$\begin{aligned}
 |\delta_G(B_t)| + 2|E_1| &= \sum_{i=1}^k |\delta_G(B_{s_i})| + |E_2| = 2 \sum_{i=1}^k f(B_{s_i}) + k + |E_2| \\
 &= 2 \sum_{i=1}^k f(B_{s_i}) + 3,
 \end{aligned}$$

also ist $|\delta_G(B_t)|$ ungerade. Mit (20.15) folgt nun

$$\begin{aligned}
 |\delta_G(B_t)| + 2|E_1| &= 2 \sum_{i=1}^k f(B_{s_i}) + 3 = 2f(B_t) + 4 \sum_{e \in E_1} x_e - 2 \sum_{e \in E_2} x_e + 3 \\
 &< 2f(B_t) + 2|E_1| + 3,
 \end{aligned}$$

da $E_1 \cup E_2 \neq \emptyset$. Somit haben wir $|\delta_G(B_t)| = 2f(B_t) + 1$, wie erwünscht.

Fall 2: Es gilt $k = 2$ und $E_2 = \emptyset$. Dann ist $E_1 \neq \emptyset$ und mit (20.15) ist $2 \sum_{e \in E_1} x_e$ eine ganze Zahl, somit gilt $2 \sum_{e \in E_1} x_e \leq |E_1| - 1$. Beachte, dass $E_1 \neq \delta_G(B_{s_1})$, da sonst $\chi^{B_{s_2}} = \chi^{B_{s_1}} + \chi^{B_t}$, im Widerspruch zur Annahme, dass die Vektoren χ^B mit $B \in \mathcal{B}$ linear unabhängig sind. Analog zeigen wir, dass $E_1 \neq \delta_G(B_{s_2})$. Für $i = 1, 2$ haben wir dann

$$2f(B_{s_i}) = 2 \sum_{e \in \delta(B_{s_i}) \setminus E_1} x_e + 2 \sum_{e \in E_1} x_e < |\delta_G(B_{s_i}) \setminus E_1| + |E_1| - 1 = |\delta_G(B_{s_i})| - 1.$$

Mit der Induktionsvoraussetzung folgt hieraus $\sum_{v \in B_{s_i}} |\delta_G(v)| > 2|V(T_{s_i})| + 1$, und wie in (20.16) erhalten wir

$$\begin{aligned} \sum_{v \in B_t} |\delta_G(v)| &\geq \sum_{i=1}^2 \sum_{v \in B_{s_i}} |\delta_G(v)| \geq \sum_{i=1}^2 (2|V(T_{s_i})| + 2) \\ &= 2|V(T_t)| + 2. \end{aligned}$$

Fall 3: Es gilt $k = 1$ und $|E_2| \leq 1$. Beachte: Aus $k = 1$ folgt $E_1 \subseteq E_2$ und somit gilt $|E_2| = 1$. Nach (20.15) haben wir

$$\sum_{e \in E_2 \setminus E_1} x_e - \sum_{e \in E_1} x_e = \sum_{e \in E_2} x_e - 2 \sum_{e \in E_1} x_e = f(B_t) - f(B_{s_1}).$$

Dies ist jedoch ein Widerspruch, da die rechte Seite ganzzahlig ist, die linke aber nicht; somit ist Fall 3 unmöglich.

Damit ist die Behauptung bewiesen. Für $t = r$ erhalten wir $\sum_{v \in V(G)} |\delta_G(v)| \geq 2|V(T)| + 1$, d. h. $2|E(G)| > 2|V(T)|$. Da aber andererseits $|V(T)| - 1 = |E(T)| = |\mathcal{B}| = |E(G)|$, haben wir einen Widerspruch, womit der Beweis abgeschlossen ist. \square

Dieses Resultat wurde von Nagarajan, Ravi und Singh [2010] verallgemeinert. Mit Kenntnis des obigen Satzes ist der folgende Algorithmus nahe liegend:

JAINS ALGORITHMUS

Input: Ein ungerichteter Graph G , Gewichte $c : E(G) \rightarrow \mathbb{R}_+$, Kapazitäten $u : E(G) \rightarrow \mathbb{N}$ und eine brauchbare Funktion $f : 2^{V(G)} \rightarrow \mathbb{Z}_+$ (gegeben durch ein Orakel).

Output: Ein Vektor $x \in \mathbb{Z}_+^{E(G)}$ mit $\sum_{e \in \delta_G(S)} x_e \geq f(S)$ für alle $S \subseteq V(G)$ und $x_e \leq u(e)$ für alle $e \in E(G)$.

- ① Setze $x_e := 0$, falls $c(e) > 0$, und $x_e := u(e)$, falls $c(e) = 0$, für jedes $e \in E(G)$.
- ② Bestimme eine optimale Basislösung y für das LP (20.13) bezüglich c , u' und f' , wobei $u'(e) := u(e) - x_e$ für alle $e \in E(G)$ und $f'(S) := f(S) - \sum_{e \in \delta_G(S)} x_e$ für alle $S \subseteq V(G)$.
If $y_e = 0$ für alle $e \in E(G)$ **then stop**.
- ③ Setze $x_e := x_e + \lceil y_e \rceil$ für alle $e \in E(G)$ mit $y_e \geq \frac{1}{2}$.
Go to ②.

Satz 20.40. (Jain [2001]) JAINS ALGORITHMUS findet eine ganzzahlige Lösung für das LP (20.13) mit Kosten höchstens gleich zweimal dem optimalen Zielfunktionswert des LP. Er kann so implementiert werden, dass er in polynomieller Zeit läuft. Somit ist er ein 2-Approximationsalgorithmus für das SURVIVABLE-NETWORK-DESIGN-PROBLEM.

Beweis:

Zunächst weisen wir darauf hin, dass f' zu jedem Zeitpunkt schwach supermodular ist: Es ist f brauchbar und somit schwach supermodular nach Proposition 20.24. Nun haben wir für $A, B \subseteq V(G)$ $\sum_{e \in \delta_G(A)} x_e + \sum_{e \in \delta_G(B)} x_e \geq \sum_{e \in \delta_G(A \cap B)} x_e + \sum_{e \in \delta_G(A \cup B)} x_e$ und $\sum_{e \in \delta_G(A)} x_e + \sum_{e \in \delta_G(B)} x_e \geq \sum_{e \in \delta_G(A \setminus B)} x_e + \sum_{e \in \delta_G(B \setminus A)} x_e$. Also ist f' schwach supermodular.

Nach der ersten Iteration haben wir $f'(S) \leq \sum_{e \in \delta_G(S)} \frac{1}{2} \leq \frac{|E(G)|}{2}$ für alle $S \subseteq V(G)$. Nach Satz 20.39 wird bei jeder weiteren Iteration mindestens eines der x_e um wenigstens 1 erhöht. Da jedes x_e nach der ersten Iteration um höchstens $\frac{|E(G)|}{2}$ erhöht wird, ist die Gesamtanzahl der Iterationen durch $\frac{|E(G)|^2}{2}$ beschränkt.

Das einzige Implementierungsproblem ist ②). Nach Satz 4.21 genügt es, das SEPARATIONS-PROBLEM zu lösen. Für einen gegebenen Vektor $y \in \mathbb{R}^{E(G)}$ müssen wir entscheiden, ob $\sum_{e \in \delta_G(S)} y_e \geq f'(S) = f(S) - \sum_{e \in \delta_G(S)} x_e$ für alle $S \subseteq V(G)$, und falls nicht, einen verletzten Schnitt finden. Da f brauchbar ist, kann dies nach Satz 20.26 in $O(n^4 + n\theta)$ Zeit bewerkstelligt werden, wobei $n = |V(G)|$ und θ die Zeitschranke des Orakels für f ist.

Abschließend beweisen wir, dass die Approximationsgüte gleich 2 ist. Der Beweis erfolgt mittels Induktion über die Anzahl der Iterationen. Terminierte der Algorithmus während der ersten Iteration, so hat die Lösung verschwindende Kosten und ist somit optimal.

Seien anderenfalls $x^{(1)}$ und $y^{(1)}$ die Vektoren x und y nach der ersten Iteration, und sei $x^{(t)}$ der Vektor x bei Terminierung. Setze $z_e := y_e^{(1)}$, falls $y_e^{(1)} < \frac{1}{2}$, und $z_e = 0$ sonst. Dann gilt $cx^{(1)} \leq 2c(y^{(1)} - z)$. Sei $f^{(1)}$ die Residualfunktion mit $f^{(1)}(S) := f(S) - \sum_{e \in \delta_G(S)} x_e^{(1)}$. Da z eine zulässige Lösung für $f^{(1)}$ ist, folgt mit der Induktionsvoraussetzung, dass $c(x^{(t)} - x^{(1)}) \leq 2cz$. Damit folgt

$$cx^{(t)} \leq cx^{(1)} + c(x^{(t)} - x^{(1)}) \leq 2c(y^{(1)} - z) + 2cz = 2cy^{(1)}.$$

Da $cy^{(1)}$ eine untere Schranke für die Kosten einer optimalen Lösung ist, sind wir fertig. \square

Es ist keine bessere Approximationsszahl bekannt, nicht einmal für das STEINER-WALD-PROBLEM. Melkonian und Tardos [2004] haben Jains Verfahren auf ein gerichtetes Netzwerk-Design-Problem erweitert. Fleischer, Jain und Williamson [2006], Cheriyan und Vetta [2007] und Chuzhoy und Khanna [2009] haben gezeigt, wie man einige Knotenzusammenhangsbedingungen mit berücksichtigen kann. Resultate von Kortsarz, Krauthgamer und Lee [2004] und Chakraborty, Chuzhoy und Khanna [2008] zeigen jedoch, dass sich die Knotenzusammenhangsversion des SURVIVABLE-NETWORK-DESIGN-PROBLEMS schwerlich approximieren lässt.

20.7 Das VPN-Problem

Bei vielen Anwendungen sind die Nachfragewerte nicht von Anfang an genau bekannt, man muss jedoch ein Netzwerk entwerfen, das für eine gewisse Nachfragemenge gilt. Es gibt viele verschiedene Versionen dieses Problems, das oft als robustes Network-Design umschrieben wird. Hier betrachten wir den Fall, dass jedes Paar von Terminalen durch einen festen Weg verbunden ist und dass die den Kanten zugeteilten Kapazitäten für ein beliebiges gebrochenes Matching in der Menge der Terminalen reichen. Genauer: Sei

$$D(W) := \left\{ (d_f)_{f \in \binom{W}{2}} : d_f \geq 0 \ (f \in \binom{W}{2}), \sum_{w \in W \setminus \{v\}} d_{\{v,w\}} \leq 1 \ (v \in W) \right\}$$

die Menge der gebrochenen Matchings in dem vollständigen Graphen mit Knotenmenge W . Dann definieren wir das

VPN-PROBLEM

- Instanz:* Ein ungerichteter Graph G mit den Gewichten $c : E(G) \rightarrow \mathbb{R}_+$ und eine Menge $W \subseteq V(G)$ von Terminalen.
- Aufgabe:* Bestimme einen $v-w$ -Weg P_f für jedes ungeordnete Paar $f = \{v, w\}$ von Terminalen mit $\sum_{e \in E(G)} c(e)u(e)$ minimal, wobei $u(e) = \max_{d \in D(W)} \sum_{f \in \binom{W}{2}: e \in E(P_f)} d_f$ die der Kante $e \in E(G)$ zuzuordnende Kapazität ist.

Man könnte sich ein *virtual private network* (VPN) vorstellen, das ein Unternehmen für die Kommunikation zwischen seiner Niederlassungen mietet. Aufgabe 23 schildert eine geringfügige Verallgemeinerung. Im Englischen spricht man vom *oblivious routing*, weil die Wege unabhängig von den konkreten Nachfragen $d \in D(W)$ gewählt werden. Die besondere Eigenschaft des VPN-PROBLEMS ist, dass es exakt in polynomieller Zeit gelöst werden kann. Dies folgt sofort aus dem folgenden Satz:

Satz 20.41. (Goyal, Olver und Shepherd [2013]) *Für jede Instanz des VPN-PROBLEMS existiert eine optimale Lösung mit der folgenden Eigenschaft: Für ein gewisses $s \in V(G)$ (das Drehkreuz) und $s-w$ -Wege P_w ($w \in W$) ordnen wir jeder Kante $e \in E(G)$ die Kapazität $u(e) := |\{w \in W : e \in E(P_w)\}|$ zu, und es gilt $E(P_f) \subseteq E(P_v) \cup E(P_w)$ für alle $f = \{v, w\} \in \binom{W}{2}$.*

Eine solche Lösung werden wir als Drehkreuzlösung bezeichnen. Wir leiten zunächst die folgende untere Schranke ab. Sei $\tau(T \triangle \{w\})$ das geringste Gewicht eines $(T \triangle \{w\})$ -Joins in (G, c) .

Lemma 20.42. *Für jede Instanz des VPN-PROBLEMS gibt es eine Menge $T \subseteq V(G)$ ungerader Kardinalität mit der Eigenschaft: Jede Lösung hat Kosten von mindestens $\sum_{w \in W} \tau(T \triangle \{w\})$.*

Beweis: Sei $(P_f)_{f \in \binom{W}{2}}$ eine Lösung der gegebenen Instanz und $u(e) := \max_{d \in D(W)} \sum_{f \in \binom{W}{2}: e \in E(P_f)} d_f$ die von dieser Lösung benötigte Kapazität für die Kante $e \in E(G)$. Dann gibt es, wie wir zeigen werden, eine Menge $T \subseteq V(G)$ ungerader Kardinalität, so dass $\sum_{e \in E(G)} c(e)u(e) \geq \sum_{w \in W} \tau(T \triangle \{w\})$.

Für jedes $t \in W$ sei $\mathcal{P}_t := \{E(P_{\{w,t\}}) : w \in W\}$ die Menge der Kantenmengen der Wege nach t , wobei $P_{\{t,t\}} := (\{t\}, \emptyset)$. Für jedes $e \in E(G)$ setzen wir $l(e, \mathcal{P}_t) := |\{P \in \mathcal{P}_t : e \in P\}|$ und $y(e, \mathcal{P}_t) := \min\{l(e, \mathcal{P}_t), |W| - l(e, \mathcal{P}_t)\} = |\{P \in \mathcal{P}_t : e \in E(P) \triangle H_t\}|$, wobei

$$H_t := \left\{ e \in E(G) : l(e, \mathcal{P}_t) \geq \frac{|W|}{2} \right\}$$

die Menge der (“schweren”) Kanten ist, die von mindestens der Hälfte der Wege nach t benutzt werden. Sei schließlich $T_t := \text{odd}(H_t) \triangle \{t\}$, wobei $\text{odd}(H_t)$ die Menge der Knoten ungeraden Grades in $(V(G), H_t)$ ist.

Sei $e \in E(G)$. Dann behaupten wir, dass

$$d_f^e := \begin{cases} \frac{1}{|W|} \left(\frac{y(e, \mathcal{P}_v)}{l(e, \mathcal{P}_v)} + \frac{y(e, \mathcal{P}_w)}{l(e, \mathcal{P}_w)} \right) & \text{if } e \in E(P_f) \\ 0 & \text{sonst} \end{cases}$$

mit $f = \{v, w\} \in \binom{W}{2}$ ein gebrochenes Matching definiert. In der Tat folgt für jedes $v \in W$,

$$\begin{aligned} \sum_{w \in W \setminus \{v\}} d_{\{v,w\}}^e &= \sum_{w \in W \setminus \{v\}: e \in E(P_{\{v,w\}})} \frac{1}{|W|} \left(\frac{y(e, \mathcal{P}_v)}{l(e, \mathcal{P}_v)} + \frac{y(e, \mathcal{P}_w)}{l(e, \mathcal{P}_w)} \right) \\ &\leq \sum_{w \in W \setminus \{v\}: e \in E(P_{\{v,w\}})} \frac{1}{|W|} \left(\frac{|W| - l(e, \mathcal{P}_v)}{l(e, \mathcal{P}_v)} + \frac{l(e, \mathcal{P}_w)}{l(e, \mathcal{P}_w)} \right) \\ &= \sum_{w \in W \setminus \{v\}: e \in E(P_{\{v,w\}})} \frac{1}{|W|} \cdot \frac{|W|}{l(e, \mathcal{P}_v)} \\ &= 1. \end{aligned}$$

Somit ist $d^e \in D(W)$, also gilt

$$\begin{aligned} u(e) &\geq \sum_{f \in \binom{W}{2}: e \in E(P_f)} d_f^e \\ &= \frac{1}{|W|} \sum_{f=\{v,w\} \in \binom{W}{2}: e \in E(P_f)} \left(\frac{y(e, \mathcal{P}_v)}{l(e, \mathcal{P}_v)} + \frac{y(e, \mathcal{P}_w)}{l(e, \mathcal{P}_w)} \right) \\ &= \frac{1}{|W|} \sum_{v \in W} \sum_{w \in W \setminus \{v\}: e \in E(P_{\{v,w\}})} \frac{y(e, \mathcal{P}_v)}{l(e, \mathcal{P}_v)} \\ &= \frac{1}{|W|} \sum_{v \in W} y(e, \mathcal{P}_v). \end{aligned}$$

Damit bekommen wir schließlich

$$\begin{aligned}
 \sum_{e \in E(G)} c(e)u(e) &\geq \frac{1}{|W|} \sum_{v \in W} \sum_{e \in E(G)} c(e)y(e, \mathcal{P}_v) \\
 &= \frac{1}{|W|} \sum_{v \in W} \sum_{w \in W} c(E(P_{\{v,w\}}) \Delta H_v) \\
 &\geq \frac{1}{|W|} \sum_{v \in W} \sum_{w \in W} \tau(T_v \Delta \{w\}) \\
 &\geq \min_{v \in W} \sum_{w \in W} \tau(T_v \Delta \{w\}).
 \end{aligned}$$

□

Dieser Beweis von Goyal, Olver und Shepherd [2013] basiert auf früheren Teilergebnissen von Gupta et al. [2001] und Grandoni et al. [2008]. Als Nächstes beweisen wir das folgende Lemma.

Lemma 20.43. *Sei H ein ungerichteter Graph und $T \subseteq V(H)$ mit $|T|$ gerade und der Eigenschaft: $E(H)$ enthält k kantendisjunkte T -Joins. Dann gibt es für jedes $t \in T$ ein $s \in V(H) \setminus \{t\}$ mit der Eigenschaft: H enthält k kantendisjunkte $s-t$ -Wege.*

Beweis: Betrachte einen Gomory-Hu-Baum G für H . Die Knotenmenge wenigstens einer der Zusammenhangskomponenten von $G - t$, nennen wir sie C , hat eine Schnittmenge ungerader Kardinalität mit T (da $|T|$ gerade und $t \in T$). Sei $f = \{s, t\} \in E(G)$ mit $s \in C$. Dann ist $\delta_H(C)$ ein $s-t$ -Schnitt minimaler Kardinalität in H (nach Definition des Gomory-Hu-Baumes). Da $|C \cap T|$ ungerade ist, hat jeder T -Join nichtleeren Schnitt mit $\delta(C)$, folglich gilt $|\delta_H(C)| \geq k$. Nach dem Satz von Menger 8.9 enthält H somit k kantendisjunkte $s-t$ -Wege. □

Dieser Beweis stammt von A. Sebő (siehe Goyal, Olver und Shepherd [2013]). Wir können nun den Hauptsatz dieses Abschnittes beweisen:

Beweis für Satz 20.41: Gegeben sei eine feste Instanz (G, c, W) des VPN-PROBLEMS. Sei $T \subseteq V(G)$ wie in Lemma 20.42, d.h. $|T|$ ist ungerade und jede Lösung der Instanz hat Kosten von mindestens $\sum_{w \in W} \tau(T \Delta \{w\})$. Sei J_w ein $(T \Delta \{w\})$ -Join mit minimalen Kosten und sei H der Graph mit $V(H) = V(G) \cup \{t\}$, dessen Kantenmenge die disjunkte Vereinigung von J_w ($w \in W$) plus einer Kante $\{t, w\}$ für jedes $w \in W$ ist.

Mit anderen Worten, $E(H)$ ist die disjunkte Vereinigung der $(T \Delta \{t\})$ -Joins $J_w \cup \{\{w, t\}\}$ ($w \in W$). Nach Lemma 20.43 gibt es nun einen Knoten $s \in V(G)$ mit der Eigenschaft: H enthält $|W|$ kantendisjunkte $s-t$ -Wege. Diese entsprechen kantendisjunkten Wegen P_w ($w \in W$) in G , wobei P_w den Knoten s mit w verbindet. Somit dient s als Drehkreuz. □

Korollar 20.44. *Das VPN-PROBLEM kann in $O(|W|(m + n \log n))$ Zeit gelöst werden, wobei $n = |V(G)|$ und $m = |E(G)|$.*

Beweis: Man berechne kürzeste Wege von jedem $w \in W$ aus nach allen $s \in V(G)$ mittels DIJKSTRAS ALGORITHMUS (Satz 7.4). Man wähle einen Knoten s , der $\sum_{w \in W} \text{dist}_{(G,c)}(s, w)$ minimiert. Dieser Knoten s dient nun als Drehkreuz in einer Drehkreuzlösung wie in Satz 20.41. \square

Aufgaben

- Sei (G, c, T) eine Instanz des STEINERBAUM-PROBLEMS, wobei G ein vollständiger Graph ist und $c : E(G) \rightarrow \mathbb{R}_+$ die Dreiecksungleichung erfüllt. Man beweise, dass es einen optimalen Steinerbaum für T mit höchstens $|T| - 2$ Steinerpunkten gibt.

- Im GERICHTETEN STEINERBAUM-PROBLEM liegt ein Digraph G vor mit den Gewichten $c : E(G) \rightarrow \mathbb{R}_+$, einer Wurzel $r \in V(G)$ und einer Menge $T \subseteq V(G)$ von Terminalen; die Aufgabe ist es, eine Arboreszenz in G mit minimalem Gewicht zu finden, die ihre Wurzel in r hat und alle Terminalen enthält. Man zeige, dass ein k -Approximationsalgorithmus für dieses Problem einen k -Approximationsalgorithmus für das MINIMUM-WEIGHT-SET-COVER-PROBLEM implizieren würde.

- Man beweise, dass das STEINERBAUM-PROBLEM sogar dann MAXSNP-schwer ist, wenn der Graph vollständig ist und alle Kantengewichte gleich 1 oder 2 sind.

Hinweis: Man modifiziere den Beweis von Satz 20.3. Was kann man für G unzusammenhängend sagen?

(Bern, Plassmann [1989])

- Man formuliere einen $O(n^3 t^2)$ Algorithmus für das STEINERBAUM-PROBLEM in einem planaren Graphen, für den alle Terminalen am äußeren Gebiet liegen, und beweise seine Korrektheit.

Hinweis: Man zeige, dass es genügt, im DREYFUS-WAGNER-ALGORITHMUS Mengen $U \subseteq T$ zu betrachten, die aufeinander folgen, d. h. es gibt einen Weg P , dessen Knoten alle am äußeren Gebiet liegen, so dass $V(P) \cap T = U$ gilt (hier können wir o. B. d. A. annehmen, dass G 2-fach zusammenhängend ist). (Erickson, Monma und Veinott [1987])

- Man beschreibe einen Algorithmus für das STEINERBAUM-PROBLEM, der für Instanzen (G, c, T) mit $|V(G) \setminus T| \leq k$, wobei k irgendeine Konstante ist, in $O(n^3)$ -Zeit läuft.

- Man beweise die folgende Verschärfung von Satz 20.6: Ist (G, c, T) eine Instanz des STEINERBAUM-PROBLEMS mit $|T| \geq 2$, (\bar{G}, \bar{c}) der metrische Abschluss, S ein optimaler Steinerbaum für T in G und M ein aufspannender Baum minimalen Gewichtes in $\bar{G}[T]$ bezüglich \bar{c} , so gilt

$$\bar{c}(M) \leq 2 \left(1 - \frac{1}{b}\right) c(S),$$

wobei b die Anzahl der Blätter (Knoten mit Grad 1) von S ist. Man zeige, dass dies die bestmögliche Schranke ist.

7. Man beweise, dass der 4-Steiner-Quotient ρ_4 gleich $\frac{3}{2}$ ist.
 8. Man verschärfe die Ungleichung in Proposition 20.10 für die Fälle $|T| = 3$ und $|T| = 4$.
 9. Man zeige, dass (20.1) für jedes feste k in polynomieller Zeit gelöst werden kann.
 - * 10. Man zeige, dass die Familie $\mathcal{B}_W(R)$ der Brückenn Mengen die Menge der Basen eines Matroids für jeden Baum (T, W) und jedes $R \subseteq T$ bildet.
(Goemans et al. [2012])
 11. Man finde einen kombinatorischen 2-Approximationsalgorithmus für das SURVIVABLE-NETWORK-DESIGN-PROBLEM mit $r_{ij} = k$ für alle i, j (d. h. für das MINIMUM-WEIGHT- k -EDGE-CONNECTED-SUBGRAPH-PROBLEM).
- Hinweis:* Man ersetze jede Kante durch ein Paar entgegengesetzt gerichteter Kanten (mit gleichen Gewichten) und wende entweder Aufgabe 26, Kapitel 13, oder Satz 6.18 an.
(Khuller und Vishkin [1994])
- Bemerkung:* Siehe Khuller und Raghavachari [1996], Gabow [2005], Jothi, Raghavachari und Varadarajan [2003] und Gabow et al. [2009] bezüglich weiterer Resultate für ähnliche Probleme.
12. Man zeige, dass die LP-Relaxierung (20.7) für den Spezialfall des SURVIVABLE-NETWORK-DESIGN-PROBLEMS als ein LP polynomieller Größe umformuliert werden kann.
 13. Man beweise die folgende Verschärfung von Satz 20.29. Gegeben sei eine brauchbare Funktion g (durch ein Orakel) und eine g fast erfüllende Menge $F \subseteq E(G)$. Dann können die aktiven Knotenmengen bezüglich (g, F) in $O(k^2n^2 + n^2\theta)$ -Zeit berechnet werden, wobei $n = |V(G)|$, $k = \max_{S \subseteq V(G)} g(S)$ und θ die von dem Orakel für g benötigte Zeit ist.

Hinweis: Die Idee ist, mit den Flussberechnungen aufzuhören, wenn der Wert des maximalen Flusses mindestens gleich k ist, weil Schnitte mit k oder mehr Kanten hier keine Rolle spielen.

Der GOMORY-HU-ALGORITHMUS (siehe Abschnitt 8.6) wird wie folgt modifiziert. Bei jedem Schritt ist jeder Knoten des Baumes T ein Wald (statt einer Kantenmenge). Die Kanten der Wälder entsprechen maximalen Fluss-Problemen, für welche der Wert eines maximalen Flusses mindestens gleich k ist. Bei jeder Iteration des modifizierten GOMORY-HU-ALGORITHMUS wähle man zwei Knoten s und t aus verschiedenen Zusammenhangskomponenten des einem Knoten von T entsprechenden Waldes. Ist der Wert des maximalen Flusses mindestens gleich k , so füge man dem Wald eine Kante $\{s, t\}$ hinzu. Andernfalls spalte man den Knoten wie im ursprünglichen Gomory-Hu-Verfahren. Man terminiert, wenn alle Knoten von T Bäume sind, und ersetzt abschließend jeden Knoten in T durch seinen entsprechenden Baum. Es ist klar, dass der modifizierte Gomory-Hu-Baum auch die Ungleichungen (20.9) und (20.10) erfüllt. Werden die Flussberechnungen mit dem FORD-FULKERSON-ALGORITHMUS bewerkstelligt und wird nach dem k -ten augmentierenden Weg terminiert, so kann die $O(k^2n^2)$ Schranke erreicht werden.

Bemerkung: Dies führt zu einer $O(k^3n^3 + kn^3\theta)$ Gesamlaufzeit des PRIMAL-DUALEN ALGORITHMUS FÜR NETZWERK-DESIGN.

(Gabow, Goemans und Williamson [1998])

- * 14. Man betrachte das SURVIVABLE-NETWORK-DESIGN-PROBLEM, welches, wie wir bereits sahen, ein Spezialfall von (20.2) ist.
 - (a) Es sei T ein aufspannender Baum maximalen Gewichtes in dem vollständigen Graphen mit Kosten r_{ij} auf der Kante $\{i, j\}$. Man zeige: Erfüllt eine Kantenmenge die Zusammenhangsbedingungen der Kanten von T , dann erfüllt sie alle Zusammenhangsbedingungen.
 - (b) Bei der Bestimmung der aktiven Knotenmengen am Anfang der Phase p brauchen wir nur einen augmentierenden i - j -Weg für jedes $\{i, j\} \in E(T)$ zu bestimmen (wir können den i - j -Fluss der Phase davor benutzen). Gibt es keinen augmentierenden i - j -Weg, so gibt es höchstens zwei mögliche aktive Knotenmengen. Unter diesen $O(n)$ Kandidaten können wir die aktiven Knotenmengen in $O(n^2)$ -Zeit finden.
 - (c) Man zeige, dass die Aktualisierung dieser Datenstrukturen in $O(kn^2)$ Gesamtzeit pro Phase bewerkstelligt werden kann.
 - (d) Man folgere hieraus, dass die aktiven Knotenmengen in $O(k^2n^2)$ Gesamlaufzeit berechnet werden können.

(Gabow, Goemans und Williamson [1998])

- 15. Man zeige, dass der Aufräumschritt ⑤ des PRIMAL-DUALEN ALGORITHMUS FÜR NETZWERK-DESIGN entscheidend ist: Ohne ihn würde der Algorithmus für $k = 1$ nicht einmal eine endliche Approximationsgüte erreichen.
- 16. Es ist kein Algorithmus für das MINIMUM-WEIGHT- T -JOIN-PROBLEM für dichte Graphen mit einer besseren Worst-Case-Komplexität als $O(n^3)$ (siehe Korollar 12.12) bekannt. Sei G ein ungerichteter Graph, $c : E(G) \rightarrow \mathbb{R}_+$ und $T \subseteq V(G)$ mit $|T|$ gerade. Man betrachte das ganzzahlige LP (20.2) mit $f(S) := 1$ für $|S \cap T|$ ungerade und $f(S) := 0$ sonst.
 - (a) Man beweise, dass unser primal-dualer Algorithmus, angewendet auf (20.2), einen Wald als Output liefert, mit der Eigenschaft, dass jede seiner Zusammenhangskomponenten eine gerade Anzahl von Elementen aus T enthält.
 - (b) Man beweise, dass jede optimale Lösung von (20.2) ein T -Join minimalen Gewichtes, eventuell plus einiger Kanten mit Gewicht 0, ist.
 - (c) Der primal-duale Algorithmus kann in $O(n^2 \log n)$ -Zeit implementiert werden, falls $f(S) \in \{0, 1\}$ für alle S . Man zeige, dass hieraus die Existenz eines 2-Approximationsalgorithmus mit derselben Laufzeit für das MINIMUM-WEIGHT- T -JOIN-PROBLEM mit nichtnegativen Gewichten folgt.

Hinweis: Nach (a) liefert der Algorithmus einen Wald F als Output. Für jede Zusammenhangskomponente C von F betrachte man $\tilde{G}[V(C) \cap T]$ und bestimme eine Tour mit Gewicht höchstens gleich zweimal dem Gewicht von C (siehe Beweis von Satz 20.6). Nun nehme man jede zweite Kante der Tour. (Eine ähnliche Idee dient als Basis von CHRISTOFIDES' ALGORITHMUS, siehe Abschnitt 21.1.)

(Goemans und Williamson [1995])

17. Man finde einen 2-Approximationsalgorithmus für das PUNKT-ZU-PUNKT-VERBINDUNGSPROBLEM: Gegeben seien ein ungerichteter Graph G mit den Kantengewichten $c : E(G) \rightarrow \mathbb{R}_+$ und Mengen $S, T \subseteq V(G)$ mit $S \cap T = \emptyset$ und $|S| = |T| \geq 1$; man bestimme eine Menge $F \subseteq E(G)$ mit minimalen Kosten, für die es eine Bijektion $\pi : S \rightarrow T$ und Wege von s nach $\pi(s)$ für alle $s \in S$ in $(V(G), F)$ gibt.

Hinweis: Man zeige, dass $f(X) = 1$, falls $|X \cap S| \neq |X \cap T|$, und $f(X) = 0$ sonst eine brauchbare Funktion definiert.

(Goemans und Williamson [1995])

18. Eine Instanz des Sammler-Steinerwald-Problems (im Englischen *prize-collecting Steiner forest problem*) besteht aus einer Instanz des STEINERWALD-PROBLEMS plus einer Strafe $\pi_{\{v,w\}} \in \mathbb{R}_+$ für jedes Terminalpaar. Ziel ist es, einen aufspannenden Wald H zu finden, der $c(E(H)) + \pi(H)$ minimiert, wobei $\pi(H)$ die Summe der Strafen auf denjenigen Terminalpaaren ist, die nicht in derselben Zusammenhangskomponente von H sind. Die natürliche LP-Relaxierung ist:

$$\min \left\{ c^\top x + \pi^\top z : \sum_{e \in \delta(U)} x_e + z_{\{v,w\}} \geq 1 \ (v \in U \subset V(G) \setminus \{w\}), \ x, z \geq 0 \right\}.$$

Man betrachte den folgenden Schwellenrundungsansatz: Für ein bestimmtes $0 \leq \alpha < 1$ setze man $x'_e := \frac{1}{1-\alpha}$ für alle e ; dann ist x' eine zulässige Lösung des Steinerwald-Problems mit denjenigen Terminalpaaren, deren Variable höchstens den Wert α hat. Man wende einen 2-Approximationsalgorithmus auf x' an.

- (a) Man zeige: Mit $\alpha = \frac{1}{3}$ ergibt dies einen 3-Approximationsalgorithmus.
 (b) Man Zeige: Wählt man das beste α , so kann man einen $(1/(1 - e^{-1/2}))$ -Approximationsalgorithmus bekommen.

(Goemans [unveröffentlicht]; siehe auch Könemann et al. [2017])

19. Man bestimme eine optimale Basislösung x für (20.13), wobei G der Petersen-Graph ist (Abb. 20.5) und $f(S) = 1$ für alle $\emptyset \neq S \subset V(G)$. Man finde eine maximale laminare Familie \mathcal{B} von straffen Kantenmengen bezüglich x , so dass die Vektoren χ^B mit $B \in \mathcal{B}$ linear unabhängig sind (siehe Lemma 20.38).

20. Man beweise, dass der optimale Zielfunktionswert von (20.13) beliebig nahe am halben Wert einer optimalen ganzzahligen Lösung liegen kann.

Bemerkung: Mit JAINS ALGORITHMUS (siehe Beweis von Satz 20.40) sieht man, dass er nicht weniger als die Hälfte sein kann.

21. Man zeige, dass man mit einer geringfügig veränderten Version von JAINS ALGORITHMUS die Anzahl der Iterationen in denen man ein LP lösen muss, durch

- (a) $2|V(G)|^2$,
 (b) $2|V(G)|$

beschränkt werden kann. Hier setze man $x_e := x_e + \lfloor y_e \rfloor$ für alle e falls es ein $y_e \geq 1$ gibt, anderenfalls führe man ein Update von x wie früher aus.

Hinweis: Man leite aus Lemma 20.38 ab, dass im zweiten Fall gilt: Alle außer $2|V(G)| - 2$ Kanten können entfernt werden. Für (b) lösche man eine weitere Kante in jeder Iteration.

22. Es sei $T(m, n)$ eine obere Schranke für die benötigte Laufzeit für die Lösung des LP in ② von JAINS ALGORITHMUS, ohne unbedingt eine optimale Basislösung zu liefern. Hier sind $m = |E(G)|$ und $n = |V(G)|$. Man beschreibe einen 2-Approximationsalgorithmus für das Survivable-Network-Design-Problem, der $O(m^2 T(m, n))$ -Laufzeit hat.
23. Gegeben seien eine endliche Menge W und ein $b \in \mathbb{R}_+^W$. Es sei

$$D(b) := \left\{ (d_f)_{f \in \binom{W}{2}} : d_f \geq 0 \ (f \in \binom{W}{2}), \sum_{w \in W \setminus \{v\}} d_{\{v, w\}} \leq b_v \ (v \in W) \right\}$$

die Menge der gebrochenen b -Matchings in dem vollständigen Graphen mit Knotenmenge W . Dann verallgemeinere man das VPN-PROBLEM dahingehend, dass man einen solchen Vektor b dem Input hinzufügt und $u(e) = \max_{d \in D(b)} \sum_{f \in \binom{W}{2}: e \in E(P_f)} d_f$ als die zu installierende Kapazität setzt. Man zeige, dass Satz 20.41 für diese Situation verallgemeinert werden kann und dass das Problem weiterhin in polynomieller Zeit gelöst werden kann.

Hinweis: Man beweise den Satz, indem man diese Verallgemeinerung auf den Fall mit $b_w = 1$ für alle $w \in W$ spezialisiert.

Literatur

Allgemeine Literatur:

- Cheng, X., und Du, D.-Z. [2001]: Steiner Trees in Industry. Kluwer, Dordrecht 2001
 Du, D.-Z., Smith, J.M., und Rubinstein, J.H. [2000]: Advances in Steiner Trees. Kluwer, Boston 2000
 Goemans, M.X., und Williamson, D.P. [1996]: The primal-dual method for approximation algorithms and its application to network design problems. In: Approximation Algorithms for NP-Hard Problems. (D.S. Hochbaum, Hrsg.), PWS, Boston, 1996
 Grötschel, M., Monma, C.L., und Stoer, M. [1995]: Design of survivable networks. In: Handbooks in Operations Research and Management Science; Volume 7; Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, Hrsg.), Elsevier, Amsterdam 1995
 Gupta, A., und Könemann, J. [2011]: Approximation algorithms for network design: a survey. Surveys in Operations Research and Management Science 16 (2011) 3–20
 Hwang, F.K., Richards, D.S., und Winter, P. [1992]: The Steiner Tree Problem; Annals of Discrete Mathematics 53. North-Holland, Amsterdam 1992
 Kerivin, H., und Mahjoub, A.R. [2005]: Design of survivable networks: a survey. Networks 46 (2005), 1–21
 Lau, L.C., Ravi, R., und Singh, M. [2011]: Iterative Methods in Combinatorial Optimization. Cambridge University Press 2011, Kapitel 10
 Prömel, H.J., und Steger, A. [2002]: The Steiner Tree Problem. Vieweg, Braunschweig 2002
 Stoer, M. [1992]: Design of Survivable Networks. Springer, Berlin 1992
 Vazirani, V.V. [2001]: Approximation Algorithms. Springer, Berlin 2001, Kapitel 22 und 23

Zitierte Literatur:

- Agrawal, A., Klein, P.N., und Ravi, R. [1995]: When trees collide: an approximation algorithm for the generalized Steiner tree problem in networks. *SIAM Journal on Computing* 24 (1995), 440–456
- Arora, S. [1998]: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM* 45 (1998), 753–782
- Berman, P., und Ramaiyer, V. [1994]: Improved approximations for the Steiner tree problem. *Journal of Algorithms* 17 (1994), 381–408
- Bern, M., und Plassmann, P. [1989]: The Steiner problem with edge lengths 1 and 2. *Information Processing Letters* 32 (1989), 171–176
- Bertsimas, D., und Teo, C. [1995]: From valid inequalities to heuristics: a unified view of primal-dual approximation algorithms in covering problems. *Operations Research* 46 (1998), 503–514
- Bertsimas, D., und Teo, C. [1997]: The parsimonious property of cut covering problems and its applications. *Operations Research Letters* 21 (1997), 123–132
- Björklund, A., Husfeldt, T., Kaski, P., und Koivisto, M. [2007]: Fourier meets Möbius: fast subset convolution. *Proceedings of the 39th Annual ACM Symposium on Theory of Computing* (2007), 67–74
- Borchers, A., und Du, D.-Z. [1997]: The k -Steiner ratio in graphs. *SIAM Journal on Computing* 26 (1997), 857–869
- Borradaile, G., Klein, P., und Mathieu, C. [2009]: An $O(n \log n)$ approximation scheme for Steiner tree in planar graphs. *ACM Transactions on Algorithms* 5 (2009), Article 31
- Byrka, J., Grandoni, F., Rothvoß, T., und Sanità, L. [2013]: Steiner tree approximation via iterative randomized rounding. *Journal of the ACM* 60 (2013), Artikel 6
- Chakrabarty, T., Chuzhoy, J., und Khanna, S. [2008]: Network design for vertex connectivity. *Proceedings of the 40th Annual ACM Symposium on Theory of Computing* (2008), 167–176
- Cheriyan, J., und Vetta, A. [2007]: Approximation algorithms for network design with metric costs. *SIAM Journal on Discrete Mathematics* 21 (2007), 612–636
- Chlebík, M., und Chlebíková, J. [2008]: The Steiner tree problem on graphs: Inapproximability results. *Theoretical Computer Science* 406 (2008), 207–214
- Choukhmane, E. [1978]: Une heuristique pour le problème de l’arbre de Steiner. *RAIRO Recherche Opérationnelle* 12 (1978), 207–212 [auf Französisch]
- Chuzhoy, J., und Khanna, S. [2009]: An $O(k^3 \log n)$ -approximation algorithm for vertex-connectivity survivable network design. *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science* (2009), 437–441
- Dreyfus, S.E., und Wagner, R.A. [1972]: The Steiner problem in graphs. *Networks* 1 (1972), 195–207
- Du, D.-Z., Zhang, Y., und Feng, Q. [1991]: On better heuristic for Euclidean Steiner minimum trees. *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science* (1991), 431–439 (siehe auch *Mathematical Programming* 57 (1992) 193–202)
- Erickson, R.E., Monma, C.L., und Veinott, A.F., Jr. [1987]: Send-and-split method for minimum concave-cost network flows. *Mathematics of Operations Research* 12 (1987), 634–664
- Fleischer, L., Jain, K., und Williamson, D.P. [2006]: Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *Journal of Computer and System Sciences* 72 (2006), 838–867

- Fuchs, B., Kern, W., Mölle, D., Richter, S., Rossmanith, P., und Wang, X. [2007]: Dynamic programming for minimum Steiner trees. *Theory of Computing Systems* 41 (2007), 493–500
- Gabow, H.N. [2005]: An improved analysis for approximating the smallest k -edge connected spanning subgraph of a multigraph. *SIAM Journal on Discrete Mathematics* 19 (2005), 1–18
- Gabow, H.N., Goemans, M.X., und Williamson, D.P. [1998]: An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming B* 82 (1998), 13–40
- Gabow, H.N., Goemans, M.X., Tardos, É., und Williamson, D.P. [2009]: Approximating the smallest k -edge connected spanning subgraph by LP-rounding. *Networks* 53 (2009), 345–357
- Garey, M.R., Graham, R.L., und Johnson, D.S. [1977]: The complexity of computing Steiner minimal trees. *SIAM Journal of Applied Mathematics* 32 (1977), 835–859
- Garey, M.R., und Johnson, D.S. [1977]: The rectilinear Steiner tree problem is NP -complete. *SIAM Journal on Applied Mathematics* 32 (1977), 826–834
- Gilbert, E.N., und Pollak, H.O. [1968]: Steiner minimal trees. *SIAM Journal on Applied Mathematics* 16 (1968), 1–29
- Goemans, M.X., und Bertsimas, D.J. [1993]: Survivable networks, linear programming and the parsimonious property. *Mathematical Programming* 60 (1993), 145–166
- Goemans, M.X., Goldberg, A.V., Plotkin, S., Shmoys, D.B., Tardos, É., und Williamson, D.P. [1994]: Improved approximation algorithms for network design problems. *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms* (1994), 223–232
- Goemans, M.X., Olver, N., Rothvoß, T., und Zenklusen, R. [2012]: Matroids and integrality gaps for hypergraphic Steiner tree relaxations. *Proceedings of the 44th Annual ACM Symposium on Theory of Computing* (2012), 1161–1176
- Goemans, M.X., und Williamson, D.P. [1995]: A general approximation technique for constrained forest problems. *SIAM Journal on Computing* 24 (1995), 296–317
- Goyal, N., Olver, N., und Shepherd, F.B. [2013]: The VPN conjecture is true. *Journal of the ACM* 60 (2013), Artikel 17
- Grandoni, F., Kaibel, V., Oriolo, G., und Skutella, M. [2008]: A short proof of the VPN tree routing conjecture on ring networks. *Operations Research Letters* 36 (2008), 361–365
- Gröpl, C., Hougardy, S., Nierhoff, T., und Prömel, H.J. [2001]: Approximation algorithms for the Steiner tree problem in graphs. In: Cheng und Du [2001], S. 235–279
- Gupta, A., Kleinberg, J., Kumar, A., Rastogi, R., und Yener, B. [2001]: Provisioning a virtual private network: a network design problem for multicommodity flow. *Proceedings of the 33th Annual ACM Symposium on Theory of Computing* (2001), 389–398
- Hanan, M. [1966]: On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics* 14 (1966), 255–265
- Hetzl, A. [1995]: Verdrahtung im VLSI-Design: Spezielle Teilprobleme und ein sequentielles Lösungsverfahren. Dissertation, Universität Bonn, 1995
- Hougardy, S., und Prömel, H.J. [1999]: A 1.598 approximation algorithm for the Steiner tree problem in graphs. *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms* (1999), 448–453
- Hougardy, S., Silvanus, J., und Vygen, J. [2017]: Dijkstra meets Steiner: a fast exact goal-oriented Steiner tree algorithm. *Mathematical Programming Computation* 9 (2017), 135–202
- Hwang, F.K. [1976]: On Steiner minimal trees with rectilinear distance. *SIAM Journal on Applied Mathematics* 30 (1976), 104–114

- Jain, K. [2001]: A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica* 21 (2001), 39–60
- Jothi, R., Raghavachari, B., und Varadarajan, S. [2003]: A 5/4-approximation algorithm for minimum 2-edge-connectivity. *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms* (2003), 725–734
- Karp, R.M. [1972]: Reducibility among combinatorial problems. In: *Complexity of Computer Computations* (R.E. Miller, J.W. Thatcher, Hrsg.), Plenum Press, New York 1972, S. 85–103
- Karpinski, M., und Zelikovsky, A. [1997]: New approximation algorithms for Steiner tree problems. *Journal of Combinatorial Optimization* 1 (1997), 47–65
- Khuller, S., und Raghavachari, B. [1996]: Improved approximation algorithms for uniform connectivity problems. *Journal of Algorithms* 21 (1996), 434–450
- Khuller, S., und Vishkin, U. [1994]: Biconnectivity augmentations and graph carvings. *Journal of the ACM* 41 (1994), 214–235
- Klein, P.N., und Ravi, R. [1993]: When cycles collapse: a general approximation technique for constrained two-connectivity problems. *Proceedings of the 3rd Integer Programming and Combinatorial Optimization Conference* (1993), 39–55
- Könemann, J., Oliver, N., Pashkovich, K., Ravi, R., Swamy, C., und Vygen, J. [2017]: On the integrality gap of the prize-collecting Steiner forest LP. *Proceedings of APPROX 2017*, Artikel 17
- Korte, B., Prömel, H.J., und Steger, A. [1990]: Steiner trees in VLSI-layout. In: *Paths, Flows, and VLSI-Layout* (B. Korte, L. Lovász, H.J. Prömel, A. Schrijver, Hrsg.), Springer, Berlin 1990, S. 185–214
- Kortsarz, G., Krauthgamer, R., und Lee, J.R. [2004]: Hardness of approximation for vertex-connectivity network design problems. *SIAM Journal on Computing* 33 (2004), 704–720
- Kou, L.T. [1990]: On efficient implementation of an approximation algorithm for the Steiner tree problem. *Acta Informatica* 27 (1990), 369–380
- Kou, L.T., Markowsky, G., und Berman, L. [1981]: A fast algorithm for Steiner trees. *Acta Informatica* 15 (1981), 141–145
- Martin, A. [1992]: Packen von Steinerbäumen: Polyedrische Studien und Anwendung. Ph.D. thesis, Technical University of Berlin 1992 [auf Deutsch]
- Mehlhorn, K. [1988]: A faster approximation algorithm for the Steiner problem in graphs. *Information Processing Letters* 27 (1988), 125–128
- Melkonian, V., und Tardos, É. [2004]: Algorithms for a network design problem with crossing supermodular demands. *Networks* 43 (2004), 256–265
- Nagarajan, V., Ravi, R., und Singh, M. [2010]: Simpler analysis of LP extreme points for traveling salesman and survivable network design problems. *Operations Research Letters* 38 (2010), 156–160
- Robins, G., und Zelikovsky, A. [2005]: Tighter bounds for graph Steiner tree approximation. *SIAM Journal on Discrete Mathematics* 19 (2005), 122–134
- Takahashi, M., und Matsuyama, A. [1980]: An approximate solution for the Steiner problem in graphs. *Mathematica Japonica* 24 (1980), 573–577
- Vygen, J. [2011]: Faster algorithm for optimum Steiner trees. *Information Processing Letters*, 111 (2011), 1075–1079
- Warne, D.M., Winter, P., und Zachariasen, M. [2000]: Exact algorithms for plane Steiner tree problems: a computational study. In: *Advances in Steiner trees* (D.-Z. Du, J.M. Smith, J.H. Rubinstein, Hrsg.), Kluwer Academic Publishers, Boston, 2000, S. 81–116
- Williamson, D.P., Goemans, M.X., Mihail, M., und Vazirani, V.V. [1995]: A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica* 15 (1995), 435–454

Zelikovsky, A.Z. [1993]: An $11/6$ -approximation algorithm for the network Steiner problem.
Algorithmica 9 (1993), 463–470



21 Das Traveling-Salesman-Problem

In Kapitel 15 haben wir das TRAVELING-SALESMAN-PROBLEM (TSP) definiert und bewiesen, dass es *NP*-schwer ist (Satz 15.43). Das TSP ist das wahrscheinlich am besten untersuchte *NP*-schwere kombinatorische Optimierungsproblem und es gibt viele dafür entwickelte und verwendete Verfahren. Als erstes werden wir in den Abschnitten 21.1 und 21.2 Approximationsalgorithmen betrachten. In der Praxis haben sich für große Instanzen sogenannte lokale Suchalgorithmen (siehe Abschnitt 21.3) bewährt, obwohl sie keine endliche Approximationsgüte haben.

In Abschnitt 21.4 besprechen wir das Traveling-Salesman-Polytop (die konvexe Hülle der Inzidenzvektoren aller Touren in K_n). Mit einem Schnittebenenansatz (siehe Abschnitt 5.5), verbunden mit einem Branch-and-Bound Schema, kann man TSP-Instanzen mit einigen Tausend Städten optimal lösen. Dies werden wir in Abschnitt 21.6 besprechen, nachdem wir in Abschnitt 21.5 gezeigt haben, wie man gute untere Schranken erhalten kann. Beachte, dass all diese Ideen und Methoden auch auf andere kombinatorische Optimierungsprobleme angewendet werden können. Wir präsentieren sie jedoch im TSP-Kontext, weil dies ein Problem ist, wo diese Verfahren sich als besonders effektiv herausgestellt haben.

Hier betrachten wir nur das symmetrische TSP, obwohl das ASYMMETRISCHE TSP (wo die Distanz von i nach j nicht gleich der von j nach i zu sein braucht) auch interessant ist (siehe Aufgabe 4).

21.1 Approximationsalgorithmen für das TSP

In den nächsten beiden Abschnitten werden wir die Approximierbarkeit des TSP untersuchen. Wir beginnen mit dem folgenden negativen Ergebnis.

Satz 21.1. (Sahni und Gonzalez [1976]) *Gilt $P \neq NP$, so gibt es für kein $k \geq 1$ einen k -Approximationsalgorithmus für das TSP.*

Beweis: Angenommen, es gäbe einen k -Approximationsalgorithmus A für das TSP. Wir werden beweisen, dass es dann einen polynomiellen Algorithmus für das HAMILTON-KREIS-Problem gibt. Da aber letzteres nach Satz 15.25 *NP*-vollständig ist, folgt hieraus $P = NP$.

Für einen gegebenen Graphen G konstruieren wir eine Instanz des TSP mit $n = |V(G)|$ Städten: Die Distanzen $c : E(K_n) \rightarrow \mathbb{Z}_+$ seien gegeben durch

$$c(\{i, j\}) := \begin{cases} 1 & \text{für } \{i, j\} \in E(G) \\ 2 + (k - 1)n & \text{für } \{i, j\} \notin E(G). \end{cases}$$

Nun wenden wir A auf diese Instanz an. Hat die als Output gelieferte Tour die Länge n , so ist sie ein Hamilton-Kreis in G . Andernfalls ist die Länge der gelieferten Tour mindestens gleich $n + 1 + (k - 1)n = kn + 1$. Hat eine optimale Tour die Länge $\text{OPT}(K_n, c)$, so gilt $\frac{kn+1}{\text{OPT}(K_n, c)} \leq k$, da A ein k -Approximationsalgorithmus ist. Somit ist $\text{OPT}(K_n, c) > n$, d. h. G hat keinen Hamilton-Kreis. \square

In den meisten praktischen Anwendungen erfüllen die Distanzen der TSP-Instanzen die Dreiecksungleichung:

METRISCHES TSP

- Instanz:* Ein vollständiger Graph K_n mit Gewichten $c : E(K_n) \rightarrow \mathbb{R}_+$, so dass $c(\{x, y\}) + c(\{y, z\}) \geq c(\{x, z\})$ für alle $x, y, z \in V(K_n)$.
- Aufgabe:* Bestimme einen Hamilton-Kreis in K_n mit minimalem Gewicht.

Mit anderen Worten, es ist (K_n, c) sein eigener metrischer Abschluss.

Satz 21.2. *Das METRISCHE TSP ist stark NP-schwer.*

Beweis: Der Beweis von Satz 15.43 zeigt: Das TSP ist auch dann NP-schwer, wenn alle Distanzen gleich 1 oder 2 sind. \square

Es fallen einem sofort einige nahe liegende Heuristiken ein, die zu recht guten Lösungen führen. Eine der einfachsten ist die sogenannte Nearest-Neighbour-Heuristik: Für eine gegebene Instanz (K_n, c) des TSP wähle man ein beliebiges $v_1 \in V(K_n)$; dann wähle man für jedes $i = 2, \dots, n$ ein v_i aus $V(K_n) \setminus \{v_1, \dots, v_{i-1}\}$ mit $c(\{v_{i-1}, v_i\})$ minimal. Anders ausgedrückt: Ein Schritt besteht aus der Wahl der am nächsten liegenden noch nicht besuchten Stadt.

Die Nearest-Neighbour-Heuristik ist für kein $k' \geq 1$ ein k' -Approximationsalgorithmus für das METRISCHE TSP. Es gibt Instanzen (K_n, c) für unendlich viele n , für die die Nearest-Neighbour-Heuristik eine Tour der Länge $\frac{1}{3} \text{OPT}(K_n, c) \log n$ liefert (Rosenkrantz, Stearns und Lewis [1977]). Siehe auch Hurkens und Woeginger [2004].

Den Rest dieses Abschnitts widmen wir den Approximationsalgorithmen für das METRISCHE TSP. In diesen Algorithmen wird zunächst ein alle Knoten enthaltender geschlossener Spaziergang konstruiert, wobei Knotenwiederholungen erlaubt sind. Das folgende Lemma beweist, dass dies genügt, falls die Dreiecksungleichung gilt.

Lemma 21.3. *Gegeben sei eine Instanz (K_n, c) des METRISCHEN TSP und ein zusammenhängender eulerscher Graph G mit $V(G) = V(K_n)$ und eventuell parallelen Kanten. Dann können wir eine Tour in linearer Zeit konstruieren, deren Gewicht höchstens $c(E(G))$ beträgt.*

Beweis: Nach Satz 2.25 können wir in linearer Zeit einen eulerschen Spaziergang in G finden. Die Reihenfolge, in der die Knoten in diesem Spaziergang auftreten (wobei wir nur das erstmalige Auftreten berücksichtigen), definiert eine Tour. Aus der Dreiecksungleichung folgt sofort, dass die Länge dieser Tour höchstens $c(E(G))$ beträgt. \square

Wir sind dieser Idee bereits früher begegnet, nämlich bei der Approximation des STEINERBAUM-PROBLEMS (Satz 20.6).

DOPPELBAUM-ALGORITHMUS

Input: Eine Instanz (K_n, c) des METRISCHEN TSP.

Output: Eine Tour.

- ① Finde einen aufspannenden Baum T in K_n mit minimalem Gewicht bezüglich c .
- ② Sei G der zwei Kopien einer jeden Kante von T enthaltende Graph. Es erfüllt G die Voraussetzungen von Lemma 21.3.
Konstruiere eine Tour wie im Beweis von Lemma 21.3.

Satz 21.4. *Der DOPPELBAUM-ALGORITHMUS ist ein 2-Approximationsalgorithmus für das METRISCHE TSP. Seine Laufzeit beträgt $O(n^2)$.*

Beweis: Die Laufzeit folgt mit Satz 6.6. Es gilt $c(E(T)) \leq \text{OPT}(K_n, c)$, da das Entfernen einer einzigen Kante aus einer beliebigen Tour einen aufspannenden Baum ergibt. Somit gilt $c(E(G)) = 2c(E(T)) \leq 2 \text{OPT}(K_n, c)$. Der Satz folgt nun mit Lemma 21.3. \square

Für euklidische Instanzen (siehe Abschnitt 21.2) kann man in $O(n^3)$ -Zeit eine optimale Tour in dem metrischen Abschluss von (T, c) in ② finden, anstatt Lemma 21.3 anzuwenden (Burkard, Deineko und Woeginger [1998]). Die Approximationsgüte des DOPPELBAUM-ALGORITHMUS ist bestmöglich (Aufgabe 6). Der bisher beste Approximationsalgorithmus für das METRISCHE TSP stammt von Christofides [1976]:

CHRISTOFIDES' ALGORITHMUS

Input: Eine Instanz (K_n, c) des METRISCHEN TSP.

Output: Eine Tour.

- ① Finde einen aufspannenden Baum T in K_n mit minimalem Gewicht bezüglich c .
- ② Sei W die Menge der Knoten mit ungeradem Grad in T . Finde einen W -Join J in K_n mit minimalem Gewicht bezüglich c .
- ③ Sei $G := (V(K_n), E(T) \dot{\cup} J)$. Es erfüllt G die Voraussetzungen von Lemma 21.3.
Konstruiere eine Tour wie im Beweis von Lemma 21.3.

Wegen der Dreiecksungleichung kann man als J in ② ein perfektes Matching minimalen Gewichtes in $K_n[W]$ nehmen.

Satz 21.5. (Christofides [1976]) *CHRISTOFIDES' ALGORITHMUS ist ein $\frac{3}{2}$ -Approximationsalgorithmus für das METRISCHE TSP. Seine Laufzeit beträgt $O(n^3)$.*

Beweis: Die Laufzeit folgt mit Satz 12.10. Wie im Beweis von Satz 21.4 folgt $c(E(T)) \leq \text{OPT}(K_n, c)$. Da jede Tour die Vereinigung zweier W -Joins ist, gilt ferner $c(J) \leq \frac{1}{2} \text{OPT}(K_n, c)$. Demnach haben wir $c(E(G)) = c(E(T)) + c(J) \leq \frac{3}{2} \text{OPT}(K_n, c)$ und der Satz folgt nun mit Lemma 21.3. \square

Es ist unbekannt, ob es einen Approximationsalgorithmus mit einer besseren Approximationsgüte gibt. Andererseits haben wir das folgende negative Resultat:

Satz 21.6. (Papadimitriou und Yannakakis [1993]) *Das METRISCHE TSP ist MAXSNP-schwer.*

Beweis: Wir werden eine L-Reduktion vom MINIMUM-VERTEX-COVER-PROBLEM für Graphen mit Grad höchstens gleich 4 (letzteres Problem ist MAXSNP-schwer nach Satz 16.46) auf das METRISCHE TSP beschreiben.

Für einen gegebenen ungerichteten Graphen G mit Grad höchstens gleich 4 konstruieren wir eine Instanz (H, c) des METRISCHEN TSP wie folgt:

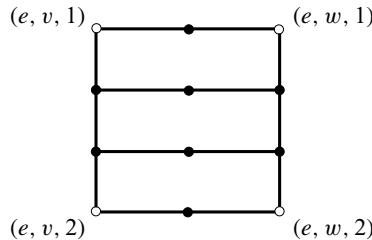


Abbildung 21.1.

Für jedes $e = \{v, w\} \in E(G)$ sei H_e der in Abb. 21.1 dargestellte Teilgraph mit zwölf Knoten und vierzehn Kanten. Die vier mit $(e, v, 1)$, $(e, v, 2)$, $(e, w, 1)$ und $(e, w, 2)$ bezeichneten Knoten von H_e haben eine besondere Bedeutung. Der Graph H_e hat die Eigenschaft, dass er einen hamiltonschen Weg von $(e, v, 1)$ nach $(e, v, 2)$ und einen weiteren von $(e, w, 1)$ nach $(e, w, 2)$ hat, aber keinen von (e, v, i) nach (e, w, j) für $i, j \in \{1, 2\}$.

Nun sei H der vollständige Graph mit $V(H) := \bigcup_{e \in E(G)} V(H_e)$. Für jede Kante $\{x, y\} \in E(H)$ sei

$$c(\{x, y\}) := \begin{cases} 1 & \text{für } \{x, y\} \in E(H_e) \text{ mit } e \in E(G); \\ \text{dist}_{H_e}(x, y) & \text{für } x, y \in V(H_e) \text{ mit } \{x, y\} \notin E(H_e) \\ & \text{und } e \in E(G); \\ 4 & \text{für } x = (e, v, i) \text{ und } y = (f, v, j) \text{ mit } e \neq f; \\ 5 & \text{sonst.} \end{cases}$$

In Abb. 21.2 wird diese Konstruktion dargestellt (nur Kanten der Länge 1 oder 4 sind abgebildet).

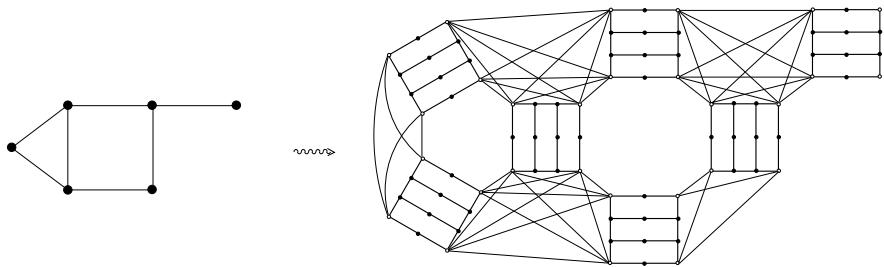


Abbildung 21.2.

Es ist (H, c) eine Instanz des METRISCHEN TSP. Wir behaupten nun, dass (H, c) die folgenden Eigenschaften hat:

- (a) Für jede Knotenüberdeckung X von G gibt es eine Tour der Länge $15|E(G)| + |X|$.
- (b) Für eine gegebene Tour T kann man in polynomieller Zeit eine weitere Tour T' konstruieren, die höchstens gleich lang ist und einen hamiltonschen Weg für jeden Teilgraphen H_e ($e \in E(G)$) enthält.
- (c) Für eine gegebene Tour der Länge $15|E(G)| + k$ kann man in polynomieller Zeit eine Knotenüberdeckung der Kardinalität k in G konstruieren.

Aus (a) und (c) folgt, dass wir eine L-Reduktion haben, weil die optimale Tourenlänge gleich $15|E(G)| + \tau(G) \leq 15(4\tau(G)) + \tau(G)$ ist, da G höchstens den Grad 4 hat.

Zum Beweis von (a) sei X eine Knotenüberdeckung von G und $(E_x)_{x \in X}$ eine Partition von $E(G)$ mit $E_x \subseteq \delta(x)$ ($x \in X$). Dann enthält der durch $\bigcup_{e \in E_x} V(H_e)$ induzierte Teilgraph für jedes $x \in X$ offensichtlich einen hamiltonschen Weg mit $11|E_x|$ Kanten der Länge 1 und $|E_x| - 1$ Kanten der Länge 4. Fügen wir $|X|$ Kanten zu der Vereinigung dieser hamiltonschen Wege hinzu, so erhalten wir eine Tour mit nur $|X|$ Kanten der Länge 5, $|E(G)| - |X|$ Kanten der Länge 4 und $11|E(G)|$ Kanten der Länge 1.

Zum Beweis von (b) sei T eine Tour und $e \in E(G)$ mit der Eigenschaft, dass T keinen hamiltonschen Weg in H_e enthält. Sei $\{x, y\} \in E(T)$ mit $x \notin V(H_e)$ und $y \in V(H_e)$, und sei z der erste Knoten nicht in $V(H_e)$ beim Durchlaufen von T von y aus, ohne x zu durchlaufen. Dann entfernen wir das Stück der Tour zwischen x und z und ersetzen es durch $\{x, (e, v, 1)\}$ und einem hamiltonschen Weg in H_e von $(e, v, 1)$ nach $(e, v, 2)$ und der Kante $\{(e, v, 2), z\}$ (mit $v \in e$ beliebig). An sämtlichen weiteren Stellen, wo T Knoten von H_e enthält, fügen wir eine Abkürzung in T ein. Wir behaupten, dass die resultierende Tour T' nicht länger als T ist.

Zunächst nehmen wir an, dass $|\delta_T(V(H_e))| \geq 4$. Setze $k := |\delta_T(V(H_e))|$. Dann ist das Gesamtgewicht der Kanten in T mit mindestens einem Endknoten in $V(H_e)$ mindestens gleich $4k + (12 - \frac{k}{2})$. In T' ist das entsprechende Gesamtgewicht höchstens gleich $5 + 5 + 11$, und wir haben noch weitere $\frac{k}{2} - 1$ Kanten wegen der Abkürzungen hinzugefügt. Da $5 + 5 + 11 + 5(\frac{k}{2} - 1) \leq 4k + (12 - \frac{k}{2})$, ist die Tour nicht länger geworden.

Als Nächstes nehmen wir an, dass $|\delta_T(V(H_e))| = 2$ und dass T aber eine Kante $\{x, y\}$ mit $x, y \in V(H_e)$ und $\{x, y\} \notin E(H_e)$ enthält. Somit ist das Gesamtgewicht der Kanten in T mit einem Endknoten in $V(H_e)$ mindestens gleich 21, wie man leicht prüfen kann. Da das entsprechende Gesamtgewicht in T' höchstens gleich $5 + 5 + 11 = 21$ ist, ist die Tour nicht länger geworden.

Zum Beweis von (c) sei T eine Tour der Länge $15|E(G)| + k$ für ein gewisses k . Nach (b) können wir annehmen, dass T einen hamiltonschen Weg für jedes H_e ($e \in E(G)$) enthält, etwa von $(e, v, 1)$ nach $(e, v, 2)$, wobei wir $v(e) := v$ gesetzt haben. Dann ist $X := \{v(e) : e \in E(G)\}$ eine Knotenüberdeckung von G . Es enthält T genau $11|E(G)|$ Kanten der Länge 1, $|E(G)|$ Kanten der Länge 4 oder 5 und mindestens $|X|$ Kanten der Länge 5. Demnach folgt, dass $|X| \leq k$. \square

Somit folgt nach Korollar 16.40, dass es kein Approximationsschema gibt, sofern $P \neq NP$. Karpinski, Lampis und Schmied [2013] haben gezeigt, dass sogar die Existenz eines $\frac{124}{123}$ -Approximationsalgorithmus bedeuten würde, dass $P = NP$.

Papadimitriou und Yannakakis [1993] haben bewiesen, dass das Problem sogar dann MAXSNP-schwer ist, wenn alle Gewichte gleich 1 oder 2 sind. Für diesen Spezialfall haben Berman und Karpinski [2006] einen $\frac{8}{7}$ -Approximationsalgorithmus gefunden. Siehe Vygen [2012] für einen Überblick über weitere Approximationsalgorithmen für Spezialfälle.

21.2 Das euklidische TSP

In diesem Abschnitt werden wir das TSP für euklidische Instanzen betrachten.

EUKLIDISCHES TSP

Instanz: Eine endliche Menge $V \subseteq \mathbb{R}^2$ mit $|V| \geq 3$.

Aufgabe: Bestimme einen Hamilton-Kreis T in dem vollständigen Graphen auf V , so dass die Gesamtlänge $\sum_{\{v,w\} \in E(T)} \|v - w\|_2$ minimiert wird.

Hier bezeichnet $\|v - w\|_2$ die euklidische Distanz zwischen v und w . Wir werden oft eine Kante mit der geradlinigen Verbindungsstrecke zwischen ihren Endknoten gleichsetzen. Somit kann jede optimale Tour als ein Polygon betrachtet werden (wobei Selbstkreuzungen nicht erlaubt sind).

Das EUKLIDISCHE TSP ist offensichtlich ein Spezialfall des METRISCHEN TSP und es ist auch stark NP -schwer (Garey, Graham und Johnson [1976], Papadimitriou [1977]). Man kann jedoch seine geometrische Struktur gewinnbringend einsetzen, wie wir jetzt zeigen werden.

Angenommen, wir haben n Punkte innerhalb eines Einheitsquadrats. Dann zerteilen wir dieses Quadrat mittels eines regulären Gitters in mehrere kleinere Rechtecke, von denen jedes einige der Punkte enthält. Nun bestimmen wir eine optimale Tour in jedem dieser Rechtecke und fügen diese Subtouren dann zu einer Gesamttour zusammen. Dieser Ansatz wurde von Karp [1977] vorgeschlagen;

er zeigte, dass man damit $(1 + \epsilon)$ -approximative Lösungen für fast alle zufällig erzeugten Instanzen im Einheitsquadrat erhalten kann. Dieses Verfahren wurde von Arora [1998] weiter entwickelt; er fand ein Approximationsschema für das EUKLIDISCHE TSP, welches wir nun besprechen werden. Ein ähnlicher Ansatz ist von Mitchell [1999] vorgeschlagen worden.

Sei ϵ mit $0 < \epsilon < 1$ fest im Rest dieses Abschnitts. Wir werden zeigen, wie man in polynomieller Zeit eine Tour finden kann, deren Gesamtlänge diejenige einer optimalen Tour um höchstens den Faktor $1 + \epsilon$ übersteigt. Wir beginnen mit der Rundung der Koordinaten:

Definition 21.7. Sei $V \subseteq \mathbb{R}^2$ eine Instanz des EUKLIDISCHEN TSP, $n := |V|$, $L := \max_{v, w \in V} \|v - w\|_2$, $v_x^{\min} := \min\{v_x : (v_x, v_y) \in V\}$ und $v_y^{\min} := \min\{v_y : (v_x, v_y) \in V\}$. Dann heißt

$$V' := \left\{ \left(1 + 2 \left\lfloor \frac{5n}{\epsilon L} (v_x - v_x^{\min}) \right\rfloor, 1 + 2 \left\lfloor \frac{5n}{\epsilon L} (v_y - v_y^{\min}) \right\rfloor \right) : (v_x, v_y) \in V \right\}$$

die zu V gehörige **wohlgerundete Instanz**.

Beachte, dass V' weniger Elemente als V enthalten kann. Ist $|V'| = 2$, so sagen wir: Der Kreis der Länge 2 der die Kante V' zweimal enthält, ist die einzige Tour für V' . Das folgende Resultat besagt, dass es genügt, wohlgerundete Instanzen zu betrachten:

Proposition 21.8. Sei $V \subseteq \mathbb{R}^2$ eine Instanz des EUKLIDISCHEN TSP, $n = |V|$ und V' die zugehörige **wohlgerundete Instanz**. Dann gilt:

- (a) für alle $(v_x, v_y) \in V'$ sind v_x und v_y ungerade ganze Zahlen im Bereich von 1 bis $1 + \frac{10n}{\epsilon}$;
- (b) aus jeder $(1 + \frac{\epsilon}{2})$ -optimalen Tour für V' können wir in $O(|V|)$ Laufzeit eine $(1 + \epsilon)$ -optimale Tour für V bilden.

Beweis: (a) folgt sofort. Gegeben sei eine Tour für V' mit Gesamtlänge l' höchstens gleich $(1 + \frac{\epsilon}{2}) \text{OPT}(V')$. Aus dieser konstruieren wir nun auf nahe liegende Weise eine Tour für die ursprüngliche Instanz V . Die Länge l dieser Tour ist höchstens gleich $\left(\frac{l'}{2} + 2n\right) \frac{\epsilon L}{5n}$. Ferner gilt

$$\text{OPT}(V') \leq 2 \left(\frac{5n}{\epsilon L} \text{OPT}(V) + 2n \right).$$

Zusammen folgt somit

$$l \leq \frac{\epsilon L}{5n} \left(2n + \left(1 + \frac{\epsilon}{2} \right) \left(\frac{5n}{\epsilon L} \text{OPT}(V) + 2n \right) \right) = \left(1 + \frac{\epsilon}{2} \right) \text{OPT}(V) + \frac{4\epsilon L}{5} + \frac{\epsilon^2 L}{5}.$$

Es ist aber $\text{OPT}(V) \geq 2L$ und $\epsilon \leq 1$, also haben wir $l \leq (1 + \epsilon) \text{OPT}(V)$. \square

Von nun an werden wir also nur noch wohlgerundete Instanzen betrachten. Die Koordinaten aller Punkte liegen innerhalb des Quadrats $[0, 2^N] \times [0, 2^N]$, wobei

$N := \lceil \log(2 + \frac{10n}{\epsilon}) \rceil$. Nun partitionieren wir dieses Quadrat schrittweise mittels eines regulären Gitters: Für $i = 1, \dots, N - 1$ sei $G_i := X_i \cup Y_i$ mit

$$\begin{aligned} X_i &:= \left\{ \left[\left(0, k2^{N-i} \right), \left(2^N, k2^{N-i} \right) \right] : k = 0, \dots, 2^i - 1 \right\}, \\ Y_i &:= \left\{ \left[\left(j2^{N-i}, 0 \right), \left(j2^{N-i}, 2^N \right) \right] : j = 0, \dots, 2^i - 1 \right\}. \end{aligned}$$

(Beachte: Es bezeichnet $[(x, y), (x', y')]$ die geradlinige Verbindungsstrecke zwischen (x, y) und (x', y') .)

Genauer: Wir betrachten **verschobene Gitter**: Seien $a, b \in \{0, 2, \dots, 2^N - 2\}$ gerade ganze Zahlen. Für $i = 1, \dots, N - 1$ sei $G_i^{(a,b)} := X_i^{(b)} \cup Y_i^{(a)}$ mit

$$\begin{aligned} X_i^{(b)} &:= \left\{ \left[\left(0, (b + k2^{N-i}) \bmod 2^N \right), \left(2^N, (b + k2^{N-i}) \bmod 2^N \right) \right] : \right. \\ &\quad \left. k = 0, \dots, 2^i - 1 \right\}, \end{aligned}$$

$$\begin{aligned} Y_i^{(a)} &:= \left\{ \left[\left((a + j2^{N-i}) \bmod 2^N, 0 \right), \left((a + j2^{N-i}) \bmod 2^N, 2^N \right) \right] : \right. \\ &\quad \left. j = 0, \dots, 2^i - 1 \right\}. \end{aligned}$$

(Beachte: $x \bmod y$ bezeichnet die eindeutig bestimmte Zahl z mit $0 \leq z < y$ und $\frac{x-z}{y} \in \mathbb{Z}$.) Wir weisen darauf hin, dass $G_{N-1} = G_{N-1}^{(a,b)}$ nicht von a oder b abhängt.

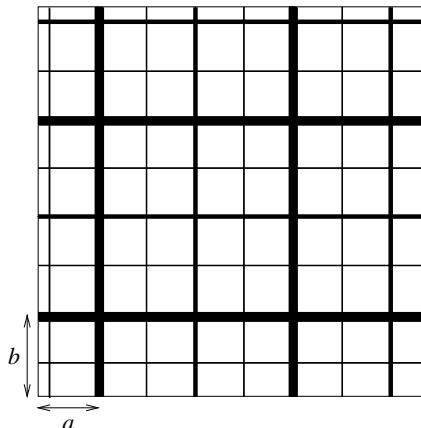


Abbildung 21.3.

Wir sagen: Eine Linie l ist auf **Stufe 1** bzw. auf Stufe i , falls $l \in G_1^{(a,b)}$ bzw. $l \in G_i^{(a,b)} \setminus G_{i-1}^{(a,b)}$ ($i = 2, \dots, N - 1$); siehe Abb. 21.3, wo die fetten Linien auf niedrigeren Stufen sind. Die **Regionen** des Gitters $G_i^{(a,b)}$ sind die Mengen

$$\left\{ (x, y) \in [0, 2^N] \times [0, 2^N] : \begin{array}{l} (x - a - j2^{N-i}) \bmod 2^N < 2^{N-i}, \\ (y - b - k2^{N-i}) \bmod 2^N < 2^{N-i} \end{array} \right\}$$

für $j, k \in \{0, \dots, 2^i - 1\}$. Für $i < N - 1$ können manche Regionen unzusammenhängend sein und aus zwei oder vier Rechtecken bestehen. Da alle Linien mittels gerader Koordinaten definiert sind, enthält keine Linie einen Punkt unserer wohlgerundeten Instanz des EUKLIDISCHEN TSP. Ferner enthält für alle a, b jede Region von G_{N-1} höchstens einen dieser Punkte.

Für ein Polygon T und eine Linie l von G_{N-1} bezeichnen wir mit $cr(T, l)$, wie oft l von T gekreuzt wird. Das folgende Resultat wird sich als nützlich herausstellen:

Proposition 21.9. *Für eine optimale Tour T einer wohlgerundeten Instanz V des EUKLIDISCHEN TSP gilt $\sum_{l \in G_{N-1}} cr(T, l) \leq \text{OPT}(V)$.*

Beweis: Betrachte eine Kante von T der Länge s . Sei x bzw. y die Länge der horizontalen bzw. vertikalen Projektion dieser Kante; es gilt also $s^2 = x^2 + y^2$. Ferner kreuzt sie Linien von G_{N-1} genau $\frac{x+y}{2} \leq s$ mal. Summieren wir über alle Kanten von T , so erhalten wir die gewünschte Ungleichung. \square

Setze $C := 7 + \left\lceil \frac{36}{\epsilon} \right\rceil$ und $P := N \left\lceil \frac{6}{\epsilon} \right\rceil$. Für jede Linie definieren wir nun **Portale**: Ist $l = [(0, (b + k2^{N-i}) \bmod 2^N), (2^N, (b + k2^{N-i}) \bmod 2^N)]$ eine horizontale Linie auf Stufe i , so wird die Menge der Portale von l gegeben durch

$$\left\{ \left(\left(a + \frac{h}{P} 2^{N-i} \right) \bmod 2^N, (b + k2^{N-i}) \bmod 2^N \right) : h = 0, \dots, P2^i \right\}.$$

Portale von vertikalen Linien werden analog definiert.

Definition 21.10. *Sei $V \subseteq [0, 2^N] \times [0, 2^N]$ eine wohlgerundete Instanz des EUKLIDISCHEN TSP. Seien $a, b \in \{0, 2, \dots, 2^N - 2\}$ gegeben und die verschobenen Gitter, die Größen C und P und die Portale wie oben. Eine Steinertour ist ein V enthaltender geschlossener Spaziergang bestehend aus Linienabschnitten, so dass die Kreuzungen mit jeder Linie der Gitter eine Teilmenge der Portale bilden. Eine Steinertour heißt leicht, falls sie für jedes i und jede Region von $G_i^{(a,b)}$ jede Kante der Region höchstens C mal kreuzt.*

Beachte, dass Steinertouren nicht notwendigerweise Polygone sind, da sie sich selbst kreuzen können. Um eine Steinertour leicht zu machen, werden wir oft das folgende Patching-Lemma benutzen:

Lemma 21.11. *Sei $V \subset \mathbb{R}^2$ eine Instanz des EUKLIDISCHEN TSP und T eine Tour für V . Sei l ein Abschnitt der Länge s auf einer Linie, die keinen Punkt von V enthält. Dann gibt es eine Tour für V , die um höchstens $6s$ länger ist als T und die l höchstens zweimal kreuzt.*

Beweis: Wir können o. B. d. A. annehmen, dass l ein vertikaler Linienabschnitt ist. Angenommen, T kreuzt l genau k mal, etwa mit den Kanten e_1, \dots, e_k . Sei $k \geq 3$;

sonst ist die Aussage trivial. Wir unterteilen nun jede der Kanten e_1, \dots, e_k mittels zweier neuer Knoten, ohne die Tourenlänge zu verlängern. Anders ausgedrückt: Wir ersetzen e_i durch einen Weg der Länge 3 mittels zweier neuer innerer Knoten $p_i, q_i \in \mathbb{R}^2$, die sehr nahe bei l liegen, wobei p_i links von l und q_i rechts von l liegt ($i = 1, \dots, k$). Es sei T' die resultierende Tour.

Sei $t := \lfloor \frac{k-1}{2} \rfloor$ (d.h. $k - 2 \leq 2t \leq k - 1$) und es gehe T'' aus T' durch Entfernen der Kanten $\{p_1, q_1\}, \dots, \{p_{2t}, q_{2t}\}$ hervor.

Es bestehe P aus einer kürzesten Tour durch p_1, \dots, p_k und einem perfekten Matching von p_1, \dots, p_{2t} mit minimalen Kosten. Analog bestehe Q aus einer kürzesten Tour durch q_1, \dots, q_k und einem perfekten Matching von q_1, \dots, q_{2t} mit minimalen Kosten. Die Gesamtlänge der Kanten in P ist höchstens gleich $3s$ und dasselbe gilt für die Kanten in Q .

Demnach kreuzt $T'' + P + Q$ die Linie l höchstens $k - 2t \leq 2$ mal und ist zusammenhängend und eulersch, und alle Knoten außer $p_1, \dots, p_k, q_1, \dots, q_k$ haben Grad 2. Nun verfahren wir wie in Lemma 21.3. Nach dem Satz von Euler (Satz 2.24) gibt es einen eulerschen Spaziergang in $T'' + P + Q$. Mittels Abkürzungswege kann dieser in eine Tour für V umgeändert werden, ohne die Länge oder die Anzahl der l -Kreuzungen zu erhöhen und ohne die Position irgendeiner Kreuzung zu ändern. \square

Der folgende Satz enthält die Schlüsselidee des Algorithmus:

Satz 21.12. (Arora [1998]) *Sei $V \subseteq [0, 2^N] \times [0, 2^N]$ eine wohlgerundete Instanz des EUKLIDISCHEN TSP. Sind a und b zwei zufällig gewählte Elemente aus $\{0, 2, \dots, 2^N - 2\}$, so gibt es mit Wahrscheinlichkeit mindestens gleich $\frac{1}{2}$ eine leichte Steinertour mit Länge höchstens gleich $(1 + \epsilon) \text{OPT}(V)$.*

Beweis: Sei T eine optimale Tour für V . Jedes Mal, wo die Tour eine Linie kreuzt, fügen wir einen Steinerpunkt hinzu.

Nun bewegen wir alle Steinerpunkte zu Portalen hin. Das einem Steinerpunkt nächste Portal auf einer Linie auf Stufe i kann bis zu einer Distanz $\frac{2^{N-i-1}}{P}$ entfernt liegen. Da eine Linie l mit Wahrscheinlichkeit $p(l, i) := \begin{cases} 2^{i-N} & \text{für } i > 1 \\ 2^{2-N} & \text{für } i = 1 \end{cases}$ auf Stufe i ist, beträgt der Erwartungswert der durch die Bewegung aller auf l liegenden Steinerpunkte zu Portalen hin bewirkten Verlängerung der Tour höchstens

$$\sum_{i=1}^{N-1} p(l, i) \cdot cr(T, l) \cdot 2 \cdot \frac{2^{N-i-1}}{P} = N \cdot \frac{cr(T, l)}{P}.$$

Als Nächstes modifizieren wir die Steinertour so, dass sie leicht wird. Sprechen wir von einem Linienabschnitt einer horizontalen bzw. vertikalen Linie von $G_i^{(a,b)}$, so meinen wir einen Linienabschnitt zwischen dem Punkt $((a+j2^{N-i}), (b+k2^{N-i}))$ und dem Punkt $((a+(j+1)2^{N-i}), (b+k2^{N-i}))$ bzw. dem Punkt $((a+j2^{N-i}), (b+(k+1)2^{N-i}))$ (alle Koordinaten sind mod 2^N zu nehmen), wobei $j, k \in \{0, \dots, 2^i - 1\}$.

Beachte, dass ein solcher Linienabschnitt aus zwei separaten Teilen bestehen kann. Betrachte das folgende Verfahren:

For $i := N - 1$ **down to 1 do:**

- Wende das Patching-Lemma 21.11 auf jeden mehr als $C - 4$ mal gekreuzten Linienabschnitt einer horizontalen Linie von $G_i^{(a,b)}$ an.
- Wende das Patching-Lemma 21.11 auf jeden mehr als $C - 4$ mal gekreuzten Linienabschnitt einer vertikalen Linie von $G_i^{(a,b)}$ an.

Zwei Bemerkungen sind hier fällig. Für aus zwei separaten Teilen bestehende horizontale oder vertikale Linienabschnitte wende man das Patching-Lemma auf jeden Teil separat an; hier kann also die Gesamtanzahl der Kreuzungen am Ende gleich 4 sein.

Beachte ferner, dass die Anwendung des Patching-Lemmas auf eine vertikale Linie l in Iteration i neue Kreuzungen (Steinerpunkte) auf einer horizontalen Linie mit einem Endknoten auf l erzeugen kann. Diese neuen Kreuzungen liegen auf Portalen und werden in den folgenden Iterationen des obigen Verfahrens nicht mehr betrachtet, weil sie auf höherstufigen Linien liegen.

Für jede Linie l beträgt die Anzahl der Anwendungen des Patching-Lemmas auf l höchstens $\frac{cr(T,l)}{C-7}$, da die Anzahl der Kreuzungen bei jeder Anwendung um mindestens $C - 7$ abnimmt (mindestens $C - 3$ Kreuzungen werden durch höchstens 4 ersetzt). Für eine gegebene Linie l bezeichne $c(l, i, a, b)$ die Anzahl der Anwendungen des Patching-Lemmas auf l bei Iteration i des obigen Verfahrens. Beachte, dass $c(l, i, a, b)$ unabhängig von der Stufe von l ist, solange diese höchstens i beträgt.

Demnach beträgt die gesamte durch die Anwendungen des Patching-Lemmas auf die Linie l bewirkte Verlängerung der Tour $\sum_{i \geq level(l)} c(l, i, a, b) \cdot 6 \cdot 2^{N-i}$. Ferner gilt

$$\sum_{i \geq level(l)} c(l, i, a, b) \leq \frac{cr(T, l)}{C - 7}.$$

Da l mit Wahrscheinlichkeit $p(l, j)$ auf Stufe j ist, beträgt der Erwartungswert der durch das obige Verfahren bewirkten Verlängerung der Tour höchstens

$$\begin{aligned} \sum_{j=1}^{N-1} p(l, j) \sum_{i \geq j} c(l, i, a, b) \cdot 6 \cdot 2^{N-i} &= 6 \sum_{i=1}^{N-1} c(l, i, a, b) \cdot 2^{N-i} \sum_{j=1}^i p(l, j) \\ &\leq \frac{12 cr(T, l)}{C - 7}. \end{aligned}$$

Nach Terminierung des obigen Verfahrens wird jede Linie (und damit auch jede Kante einer Region) höchstens $C - 4$ mal von der Tour gekreuzt, die durch das Verfahren neu hinzugefügten neuen Kreuzungen (siehe obige Bemerkung) nicht mitgezählt. Diese weiteren Kreuzungen liegen alle an jeweils einem der Endknoten der Linien. Benutzt die Tour aber ein und denselben Kreuzungspunkt drei oder mehr Male, so kann man zwei Kreuzungen entfernen, ohne die Tour zu verlängern oder weitere Kreuzungen hinzuzufügen. (Entfernt man zwei von drei parallelen Kanten aus einem zusammenhängenden eulerschen Graphen, so ist der resultierende

Graph weiterhin zusammenhängend und eulersch.) Somit erhalten wir höchstens vier zusätzliche Kreuzungen für jede Kante jeder Region (höchstens zwei pro Endknoten), woraus folgt, dass die Tour in der Tat leicht ist.

Demnach ist mit Proposition 21.9 der Erwartungswert der gesamten Verlängerung der Tour höchstens gleich

$$\sum_{l \in G_{N-1}} N \frac{cr(T, l)}{P} + \sum_{l \in G_{N-1}} \frac{12 cr(T, l)}{C - 7} \leq \text{OPT}(V) \left(\frac{N}{P} + \frac{12}{C - 7} \right) \leq \text{OPT}(V) \frac{\epsilon}{2}.$$

Damit folgt, dass die Wahrscheinlichkeit, dass die Verlängerung der Tour höchstens $\text{OPT}(V)\epsilon$ beträgt, mindestens gleich $\frac{1}{2}$ ist. \square

Mit diesem Satz können wir nun endlich ARORAS ALGORITHMUS beschreiben. Die Idee ist, alle leichten Steinertouren mittels dynamischer Optimierung aufzuzählen. Ein **Subproblem** besteht aus einer Region r eines Gitters $G_i^{(a,b)}$ mit $1 \leq i \leq N - 1$, einer Menge A gerader Kardinalität mit der Eigenschaft, dass jedes ihrer Elemente einem Portal auf einer der Kanten von r zugeteilt ist (wobei jeder Kante nicht mehr als C Elemente zugeteilt sind), und einem perfekten Matching M des vollständigen Graphen auf A . Somit haben wir für jede Region weniger als $(P+2)^{4C}(4C)!$ Subprobleme (bis auf die Umbenennung der Elemente von A). Eine Lösung eines solchen Subproblems ist eine Menge von $|M|$ Wegen $\{P_e : e \in M\}$, die den Durchschnitt einer leichten Steinertour für V mit r bilden, so dass $P_{\{v,w\}}$ die Endknoten v und w hat und jeder Punkt von $V \cap r$ genau einem der Wege angehört. Eine Lösung ist optimal, falls die Gesamtlänge der Wege minimal ist.

ARORAS ALGORITHMUS

Input: Eine wohlgerundete Instanz $V \subseteq [0, 2^N] \times [0, 2^N]$ des EUKLIDISCHEN TSP. Eine Zahl ϵ mit $0 < \epsilon < 1$.

Output: Eine bis auf den Faktor $(1 + \epsilon)$ optimale Tour.

- ① Wähle a und b zufällig aus $\{0, 2, \dots, 2^N - 2\}$.
Setze $R_0 := \{([0, 2^N] \times [0, 2^N], V)\}$.
- ② **For** $i := 1$ **to** $N - 1$ **do**:
 - Konstruiere $G_i^{(a,b)}$ und setze $R_i := \emptyset$.
 - For** jedes $(r, V_r) \in R_{i-1}$ mit $|V_r| \geq 2$ **do**:
 - Konstruiere die vier Regionen r_1, r_2, r_3, r_4 von $G_i^{(a,b)}$ mit $r_1 \cup r_2 \cup r_3 \cup r_4 = r$ und füge $(r_j, V_r \cap r_j)$ zu R_i hinzu ($j = 1, 2, 3, 4$).
- ③ **For** $i := N - 1$ **down to** 1 **do**:
 - For** jede Region $r \in R_i$ **do**: Löse alle zugehörigen Subprobleme optimal.
 - If** $|V_r| \leq 1$ **then** erledige dies direkt,
 - else** benutze die bereits berechneten optimalen Lösungen der Subprobleme für die vier Subregionen.
- ④ Berechne eine optimale leichte Steinertour für V mittels der optimalen Lösungen der Subprobleme für die vier Subregionen.
Entferne die Steinerpunkte um eine Tour zu erhalten, die nicht länger ist.

Satz 21.13. ARORAS ALGORITHMUS findet mit Wahrscheinlichkeit mindestens gleich $\frac{1}{2}$ eine Tour, die nicht länger als $(1 + \epsilon)\text{OPT}(V)$ ist. Seine Laufzeit ist $O(n(\log n)^c)$ für eine Konstante c (die linear von $\frac{1}{\epsilon}$ abhängt).

Beweis: Der Algorithmus wählt a und b zufällig und berechnet dann eine optimale leichte Steinertour. Nach Satz 21.12 hat diese mit Wahrscheinlichkeit mindestens gleich $\frac{1}{2}$ höchstens die Länge $(1 + \epsilon)\text{OPT}(V)$. Das Entfernen der Steinerpunkte am Ende kann die Tour nur kürzer machen.

Zur Abschätzung der Laufzeit betrachten wir die folgende Arboreszenz A : Die Region in R_0 bildet die Wurzel und jede Region $r \in R_i$ hat 0 oder 4 Kinder (ihre Subregionen in R_{i+1}). Sei S die Menge derjenigen Knoten von A , die 4 Kinder haben, die alle Blätter sind. Da diese Regionen ohne ihre Ränder paarweise disjunkt sind und jede von ihnen mindestens zwei Punkte von V enthält, gilt $|S| \leq \frac{n}{2}$. Da jeder Knoten von A entweder ein Blatt oder ein Vorfahre von mindestens einem Knoten in S ist, gibt es höchstens $\frac{1}{2}Nn$ Knoten, die nicht Blätter sind. Somit gibt es insgesamt höchstens $\frac{5}{2}Nn$ Knoten.

Für jede Region kommen höchstens $(P + 2)^{4C}(4C)!$ Subprobleme zustande. Diejenigen Subprobleme, die Regionen mit höchstens einem Punkt entsprechen, können direkt in $O(C)$ -Zeit gelöst werden. Für andere Subprobleme betrachten wir alle möglichen Multimengen von Portalen auf den vier Kanten zwischen den Subregionen und alle möglichen Durchlaufreihenfolgen dieser Portale. All diese höchstens $(P + 2)^{4C}(8C)!$ Möglichkeiten können dann in konstanter Zeit mittels der gespeicherten Lösungen der Subprobleme ausgewertet werden.

Somit ist die Laufzeit für alle Subprobleme einer Region $O((P + 2)^{8C}(4C)!(8C)!)$. Beachte, dass dies auch für unzusammenhängende Regionen gilt: Dadurch, dass die Tour nicht notwendigerweise von einer Zusammenhangskomponente einer Region zu einer anderen verläuft, wird das Problem nur leichter.

Da höchstens $\frac{5}{2}Nn$ Regionen betrachtet werden, da ferner $N = O(\log \frac{n}{\epsilon})$ (die Instanz ist wohlgerundet), $C = O(\frac{1}{\epsilon})$ und $P = O(\frac{N}{\epsilon})$, erhalten wir eine Gesamtaufzeit von

$$O\left(n \log \frac{n}{\epsilon} (P + 2)^{8C}(8C)^{12C}\right) = O\left(n(\log n)^{O\left(\frac{1}{\epsilon}\right)} O\left(\frac{1}{\epsilon}\right)^{O\left(\frac{1}{\epsilon}\right)}\right). \quad \square$$

Natürlich kann ARORAS ALGORITHMUS leicht durch Ausprobieren aller möglichen Werte von a und b derandomisiert werden. Dadurch wird der Laufzeit der Faktor $O\left(\frac{n^2}{\epsilon^2}\right)$ hinzugefügt. Folglich gilt:

Korollar 21.14. Es gibt ein Approximationsschema für das EUKLIDISCHE TSP. Für jedes feste $\epsilon > 0$ kann eine $(1 + \epsilon)$ -approximative Lösung in $O(n^3(\log n)^c)$ -Zeit für eine gewisse Konstante c bestimmt werden. \square

Rao und Smith [1998] haben die Laufzeit für jedes feste $\epsilon > 0$ auf $O(n \log n)$ verbessert. Die auftretenden Konstanten sind für vernünftige Werte von ϵ jedoch

immer noch recht groß. Somit scheint der praktische Nutzen recht beschränkt zu sein. Klein [2008] hat für Instanzen, die der metrische Abschluss eines planaren Graphen mit nichtnegativen Kantengewichten sind, ein Approximationsschema mit linearer Laufzeit für jedes feste $\epsilon > 0$ gefunden. Die in diesem Abschnitt besprochenen Verfahren lassen sich, leicht modifiziert, auch auf weitere geometrische Probleme anwenden; siehe z. B. Aufgabe 10.

21.3 Lokale Suche

Im Allgemeinen ist die lokale Suche das in der Praxis erfolgreichste Verfahren, um gute Lösungen für Instanzen des TSP zu erhalten. Die Idee der lokalen Suche ist die folgende. Wir beginnen mit einer mittels irgendeiner Heuristik gewonnenen Tour. Dann versuchen wir, diese mit gewissen „lokalen“ Änderungen zu verbessern. Z. B. könnten wir unsere Tour durch das Entfernen zweier Kanten in zwei Stücke teilen und diese dann einer anderen Tour hinzufügen.

Die lokale Suche ist eher ein algorithmisches Prinzip als ein Algorithmus. Insbesondere müssen zwei Entscheidungen vorweg getroffen werden:

- Welche sind die erlaubten Änderungen, d. h. wie wird die Nachbarschaft einer Lösung definiert?
- Wann werden diese Änderungen vorgenommen? (Eine Möglichkeit wäre, nur Verbesserungen zu erlauben.)

Um ein konkretes Beispiel geben zu können, beschreiben wir nun den bekannten k -OPT-ALGORITHMUS für das TSP. Sei $k \geq 2$ eine feste ganze Zahl.

k -OPT-ALGORITHMUS

Input: Eine Instanz (K_n, c) des TSP.

Output: Eine Tour T .

-
- ① Sei T irgendeine Tour.
 - ② Sei \mathcal{S} die Familie der k -elementigen Teilmengen von $E(T)$.
 - ③ **For** alle $S \in \mathcal{S}$ und alle Touren T' mit $E(T') \supseteq E(T) \setminus S$ **do:**
If $c(E(T')) < c(E(T))$ **then** setze $T := T'$ und **go to** ②.
-

Eine schnellere Implementierung dieses Algorithmus wurde von de Berg et al. [2016] vorgeschlagen.

Eine Tour heißt **k -opt**, falls sie mit dem k -OPT-ALGORITHMUS nicht weiter verbessert werden kann. Für jedes feste k gibt es Instanzen des TSP und k -opt Touren, die nicht $(k+1)$ -opt sind. Z. B. ist die Tour in Abb. 21.4 2-opt aber nicht 3-opt (bezüglich der euklidischen Distanz). Sie kann durch den Austausch dreier Kanten verbessert werden; die Tour (a, b, e, c, d, a) ist optimal.

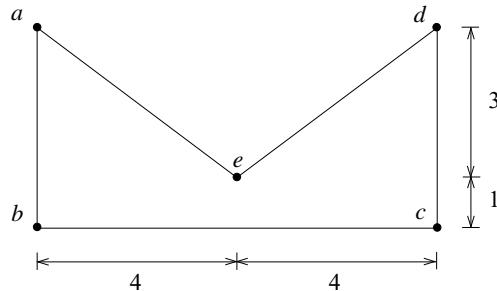


Abbildung 21.4.

Die rechte Tour in Abb. 21.5 ist 3-opt bezüglich der links angezeigten Gewichte. Die nicht eingezeichneten Kanten haben Gewicht 4. Ein Viereraustausch ergibt jedoch sofort die optimale Lösung. Beachte, dass die Dreiecksungleichung gilt.

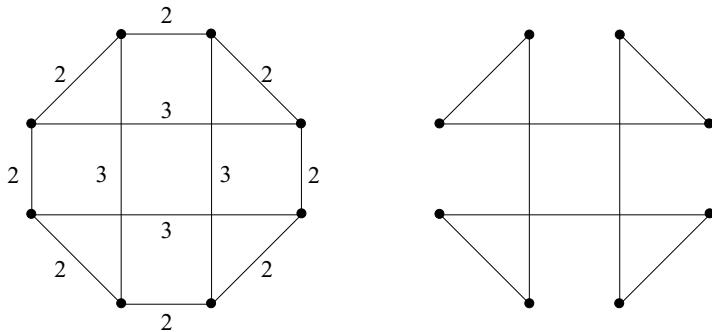


Abbildung 21.5.

Die Lage ist in der Tat noch viel schlimmer: Eine mit dem k -OPT-ALGORITHMUS für eine Instanz mit n Städten berechnete Tour kann um den Faktor $\frac{1}{4}n^{\frac{1}{2k}}$ länger sein als die optimale Tour (für alle k und unendlich viele n). Andererseits ist die Länge einer 2-opt Tour niemals schlechter als $4\sqrt{n}$ mal der Optimalwert. Die Worst-Case-Laufzeit des k -OPT-ALGORITHMUS ist jedoch für alle k exponentiell, und das gilt sogar für 2-OPT auf euklidischen Instanzen. Diese Resultate stammen von Chandra, Karloff und Tovey [1999] und Englert, Rögl und Vöcking [2014].

Eine weitere Frage ist, wie man k vorweg wählen soll. Natürlich werden Instanzen (K_n, c) mit dem k -OPT-ALGORITHMUS mit $k = n$ optimal gelöst, aber die Laufzeit wächst exponentiell mit k . Oft ist $k = 3$ eine gute Wahl. Lin und Kernighan [1973] haben eine brauchbare Heuristik vorgeschlagen, bei der k nicht fest ist, sondern von dem Algorithmus bestimmt wird. Diese Idee basiert auf dem folgenden Begriff:

Definition 21.15. Gegeben sei eine Instanz (K_n, c) des TSP und eine Tour T . Ein alternierender Spaziergang ist eine Knotenfolge $P = (x_0, x_1, \dots, x_{2m})$ mit

der Eigenschaft, dass $\{x_i, x_{i+1}\} \neq \{x_j, x_{j+1}\}$ für alle $0 \leq i < j < 2m$ und dass wir für $i = 0, \dots, 2m - 1$ haben: $\{x_i, x_{i+1}\} \in E(T)$ genau dann, wenn i gerade ist. Es heißt **P geschlossen**, falls zusätzlich $x_0 = x_{2m}$. Der **Gewinn** $g(P)$ von P wird folgendermaßen definiert:

$$g(P) := \sum_{i=0}^{m-1} (c(\{x_{2i}, x_{2i+1}\}) - c(\{x_{2i+1}, x_{2i+2}\})).$$

Es heißt **P gut**, falls $g((x_0, \dots, x_{2i})) > 0$ für alle $i \in \{1, \dots, m\}$. Wir benutzen die Abkürzung $E(P) = \{\{x_i, x_{i+1}\} : i = 0, \dots, 2m - 1\}$.

Beachte, dass Knoten in einem alternierenden Spaziergang mehrfach auftreten können. In dem Beispiel von Abb. 21.4 ist (a, e, b, c, e, d, a) ein guter geschlossener alternierender Spaziergang. Für eine gegebene Tour T sind wir natürlich an denjenigen geschlossenen alternierenden Spaziergängen P interessiert, für die $E(T) \Delta E(P)$ wiederum eine Tour definiert.

Lemma 21.16. (Lin und Kernighan [1973]) *Gibt es einen geschlossenen alternierenden Spaziergang P mit $g(P) > 0$, so gilt*

- (a) $c(E(T) \Delta E(P)) = c(E(T)) - g(P)$;
- (b) *es gibt einen guten geschlossenen alternierenden Spaziergang Q mit $E(Q) = E(P)$.*

Beweis: Es folgt (a) aus der Definition. Zum Beweis von (b) sei $P = (x_0, x_1, \dots, x_{2m})$ und k der größte Index mit $g((x_0, \dots, x_{2k}))$ minimal. Wir behaupten, dass $Q := (x_{2k}, x_{2k+1}, \dots, x_{2m-1}, x_0, x_1, \dots, x_{2k})$ gut ist. Für $i = k + 1, \dots, m$ haben wir nach der Definition von k :

$$g((x_{2k}, x_{2k+1}, \dots, x_{2i})) = g((x_0, x_1, \dots, x_{2i})) - g((x_0, x_1, \dots, x_{2k})) > 0.$$

Für $i = 1, \dots, k$ haben wir wiederum nach der Definition von k :

$$\begin{aligned} & g((x_{2k}, x_{2k+1}, \dots, x_{2m-1}, x_0, x_1, \dots, x_{2i})) \\ &= g((x_{2k}, x_{2k+1}, \dots, x_{2m})) + g((x_0, x_1, \dots, x_{2i})) \\ &\geq g((x_{2k}, x_{2k+1}, \dots, x_{2m})) + g((x_0, x_1, \dots, x_{2k})) \\ &= g(P) > 0. \end{aligned}$$

Somit ist Q tatsächlich gut. □

Wir wenden uns nun der Beschreibung des Algorithmus zu. Für eine gegebene Tour T sucht er einen guten geschlossenen alternierenden Spaziergang P , ersetzt T durch $(V(T), E(T) \Delta E(P))$ und iteriert dann. Bei jeder Iteration prüft er vollständig alle Möglichkeiten, bis er einen guten geschlossenen alternierenden Spaziergang gefunden hat oder bis einer der beiden Parameter p_1 und p_2 ihn daran hindert. Dies wird in Abb. 21.6 dargestellt.

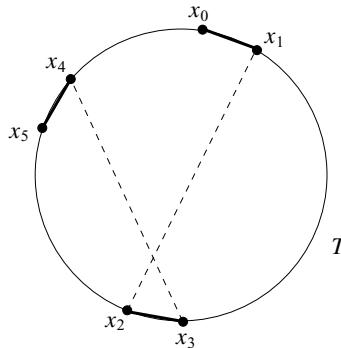


Abbildung 21.6.

LIN-KERNIGHAN-ALGORITHMUS

Input: Eine Instanz (K_n, c) des TSP. Ein aufspannender Teilgraph G von K_n . Zwei Parameter $p_1 \in \mathbb{N}$ (Backtrackingtiefe) und $p_2 \in \mathbb{N}$ (Unzulässigkeitstiefe).

Output: Eine Tour T .

- ① Sei T irgendeine Tour.
- ② Setze $X_0 := V(K_n)$, $i := 0$ und $g^* := 0$.
- ③ **If** $X_i = \emptyset$ und $g^* > 0$ **then**:
 Setze $T := (V(T), E(T) \Delta E(P^*))$ und **go to** ②.
If $X_i = \emptyset$ und $g^* = 0$ **then**:
 Setze $i := \min\{i - 1, p_1\}$. **If** $i < 0$ **then stop**, **else go to** ③.
- ④ Wähle $x_i \in X_i$ und setze $X_i := X_i \setminus \{x_i\}$.
If i ist ungerade, $i \geq 3$, $(V(T), E(T) \Delta E((x_0, x_1, \dots, x_{i-1}, x_i, x_0)))$ ist eine Tour und $g((x_0, x_1, \dots, x_{i-1}, x_i, x_0)) > g^*$ **then**:
 Setze $P^* := (x_0, x_1, \dots, x_{i-1}, x_i, x_0)$ und $g^* := g(P^*)$.
- ⑤ **If** i ist ungerade **then**:
 Setze $X_{i+1} := \{x \in \Gamma_G(x_i) \setminus \{x_0\} : \{x_i, x\} \notin E(T) \cup E((x_0, x_1, \dots, x_i)), g((x_0, x_1, \dots, x_{i-1}, x_i, x)) > g^*\}$.
If i ist gerade und $i \leq p_2$ **then**:
 Setze $X_{i+1} := \{x \in V(K_n) : \{x_i, x\} \in E(T) \setminus E((x_0, x_1, \dots, x_i))\}$.
If i ist gerade und $i > p_2$ **then**:
 Setze $X_{i+1} := \{x \in V(K_n) : \{x_i, x\} \in E(T) \setminus E((x_0, x_1, \dots, x_i)), (V(T), E(T) \Delta E((x_0, x_1, \dots, x_i, x, x_0)))$ ist eine Tour $\}$.
 Setze $i := i + 1$. **Go to** ③.

Beachte, dass in ④ Folgendes gilt: Ist $(V(T), E(T) \Delta E((x_0, x_1, \dots, x_{i-1}, x_i, x_0)))$ eine Tour, so ist $(x_0, x_1, \dots, x_{i-1}, x_i, x_0)$ ein geschlossener alternierender Spaziergang.

Lin und Kernighan haben die Parameter $p_1 = 5$ und $p_2 = 2$ vorgeschlagen. Dies sind die kleinsten Werte, welche gewährleisten, dass der Algorithmus einen günstigen Dreieraustausch findet:

Satz 21.17. (Lin und Kernighan [1973]) *Der LIN-KERNIGHAN-ALGORITHMUS*

- (a) *findet für $G = K_n$, $p_1 = \infty$ und $p_2 = \infty$ einen guten geschlossenen alternierenden Spaziergang P , so dass $(V(T), E(T) \Delta E(P))$ eine Tour ist, falls es einen gibt;*
- (b) *liefert für $G = K_n$, $p_1 = 5$ und $p_2 = 2$ eine Tour als Output, die 3-opt ist.*

Beweis: Sei T die Tour, mit welcher der Algorithmus terminiert. Dann war g^* die ganze Zeit gleich Null seit der letzten Tourenänderung. Damit folgt im Fall $p_1 = p_2 = \infty$, dass der Algorithmus alle guten alternierenden Spaziergänge vollständig durchgeprüft hat. Insbesondere gilt (a).

Ist $p_1 = 5$ und $p_2 = 2$, so hat der Algorithmus wenigstens alle guten geschlossenen alternierenden Spaziergänge der Länge 4 oder 6 vollständig durchgeprüft. Angenommen, es gäbe einen günstigen Zweier- oder Dreieraustausch mit resultierender Tour T' . Dann bilden die Kanten $E(T) \Delta E(T')$ einen geschlossenen alternierenden Spaziergang P mit höchstens sechs Kanten und $g(P) > 0$. Mit Lemma 21.16 können wir o. B. d. A. annehmen, dass P gut ist, somit hätte der Algorithmus P gefunden. Damit ist (b) bewiesen. \square

Wir weisen darauf hin, dass dieses Verfahren unmöglich einen „nicht-sequentiellen“ Austausch finden kann, wie z. B. der in Abb. 21.5 gezeigte Viereraustausch. In diesem Beispiel kann die Tour nicht mit dem LIN-KERNIGHAN-ALGORITHMUS verbessert werden, ein (nicht-sequentieller) Viereraustausch würde aber sofort die optimale Tour ergeben.

Demnach könnte eine Weiterentwicklung des LIN-KERNIGHAN-ALGORITHMUS folgendermaßen aussehen. Terminiert der Algorithmus, so versucht man (mittels gewisser Heuristiken), einen günstigen nicht-sequentiellen Viereraustausch zu finden. Gelingt dies, so fahren wir mit der neuen Tour fort, anderenfalls müssen wir aufhören.

Der Graph G sollte dünn sein (sonst könnte die Berechnung von X_{i+1} zu lange dauern), er sollte aber auch eine gute Tour enthalten. Z. B. funktioniert die Delaunay-Triangulierung oft gut für euklidische Instanzen.

Der LIN-KERNIGHAN-ALGORITHMUS ist viel effektiver als z. B. der 3-OPT-ALGORITHMUS. Er ist nicht nur mindestens so gut (und gewöhnlich sehr viel besser), auch seine erwartete Laufzeit (mit $p_1 = 5$ und $p_2 = 2$) braucht den Vergleich nicht zu scheuen: Lin und Kernighan berichten über eine empirische Laufzeit von ca. $O(n^{2.2})$. Es ist jedoch unwahrscheinlich, dass die Worst-Case-Laufzeit polynomiell ist; siehe Aufgabe 13 (Papadimitriou [1992]) für eine genaue Formulierung dieser Aussage (samt Beweis).

Fast alle für das TSP in der Praxis verwendeten Heuristiken zur lokalen Suche basieren auf diesem Algorithmus. Obwohl das ungünstigste (worst case) Verhalten schlechter ist als CHRISTOFIDES' ALGORITHMUS, liefert der LIN-KERNIGHAN-ALGORITHMUS typischerweise viel bessere Lösungen, normalerweise innerhalb einiger Prozent des Optimums. Für sehr effiziente Varianten siehe Applegate, Cook und Rohe [2003] und Helsgaun [2009].

Nach Aufgabe 13, Kapitel 9, gibt es keinen auf lokaler Suche basierenden Algorithmus für das TSP, der polynomielle Zeitkomplexität pro Iteration hat und immer eine optimale Lösung findet, sofern $P \neq NP$ (hier bedeutet eine Iteration die Zeit zwischen zwei Änderungen der aktuellen Tour). Wir werden nun zeigen, dass man sogar nicht einmal entscheiden kann, ob eine gegebene Tour optimal ist. Dazu betrachten wir zunächst die folgende Einschränkung des HAMILTON-KREIS Problems:

EINGESCHRÄNKTES HAMILTON-KREIS-PROBLEM

Instanz: Ein ungerichteter Graph G und ein hamiltonscher Weg in G .

Frage: Enthält G einen Hamilton-Kreis?

Lemma 21.18. Das EINGESCHRÄNKTE HAMILTON-KREIS-PROBLEM ist NP-vollständig.

Beweis: Für eine gegebene Instanz G des HAMILTON-KREIS Problems (welches NP-vollständig ist, siehe Satz 15.25) konstruieren wir eine äquivalente Instanz des EINGESCHRÄNKTNEN HAMILTON-KREIS-PROBLEMS.

Es sei $V(G) = \{1, \dots, n\}$. Nun nehmen wir n Kopien des in Abb. 21.7 dargestellten rautenförmigen Graphen und verbinden sie in senkrechter Reihenfolge mit den Kanten $\{S_i, N_{i+1}\}$ ($i = 1, \dots, n - 1$).

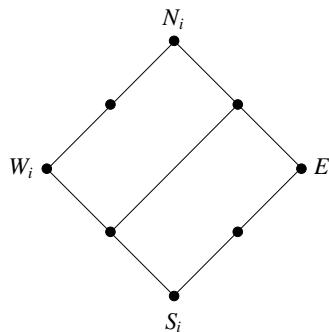


Abbildung 21.7.

Offensichtlich enthält der resultierende Graph einen hamiltonschen Weg von N_1 nach S_n . Wir fügen jetzt noch für jede Kante $\{i, j\} \in E(G)$ die beiden Kanten

$\{W_i, E_j\}$ und $\{W_j, E_i\}$ hinzu und nennen den resultierenden Graphen H . Offensichtlich induziert jeder Hamilton-Kreis in G einen Hamilton-Kreis in H .

Ferner durchläuft ein Hamilton-Kreis in H jeden der obigen rautenförmigen Graphen auf gleiche Weise: entweder alle von E_i nach W_i oder alle von S_i nach N_i . Letzteres ist aber unmöglich, somit ist H genau dann hamiltonsch, wenn G es ist. \square

Satz 21.19. (Papadimitriou und Steiglitz [1977]) *Das Problem: Entscheide, ob eine gegebene Tour für eine gegebene Instanz des METRISCHEN TSP optimal ist, ist coNP-vollständig.*

Beweis: Das Problem liegt in $coNP$, da eine optimale Tour als ein Zertifikat für Suboptimalität dient.

Wir werden nun das EINGESCHRÄNKTE HAMILTON-KREIS-PROBLEM in das Komplement unseres Problems transformieren. Für einen gegebenen Graphen G und einen hamiltonschen Weg P in G prüfen wir zunächst, ob die Enden von P durch eine Kante verbunden sind. Falls ja, so sind wir fertig. Andernfalls definieren wir

$$c_{ij} := \begin{cases} 1 & \text{für } \{i, j\} \in E(G) \\ 2 & \text{für } \{i, j\} \notin E(G). \end{cases}$$

Natürlich gilt die Dreiecksungleichung. Ferner definiert P eine Tour mit Kosten $n + 1$, die genau dann optimal ist, wenn es keinen Hamilton-Kreis in G gibt. \square

Korollar 21.20. *Es kann kein auf lokaler Suche basierender Algorithmus mit polynomieller Zeitkomplexität pro Iteration für das TSP exakt sein, sofern $P \neq NP$.*

Beweis: Ein exakter auf lokaler Suche basierender Algorithmus beinhaltet die Entscheidung, ob die Anfangstour optimal ist. \square

Die lokale Suche kann natürlich auch auf diverse andere kombinatorische Optimierungsprobleme angewendet werden. Der SIMPLEXALGORITHMUS kann auch als ein lokaler Suchalgorithmus betrachtet werden. Obwohl sich lokale Suchalgorithmen in der Praxis sehr gut bewährt haben, gibt es so gut wie keine theoretische Grundlage für ihre Effizienz, außer in einigen Spezialfällen (siehe z. B. Aufgabe 10, Kapitel 16, und die Abschnitte 22.6 und 22.8). Für viele NP -schwere Probleme und interessante Nachbarschaften von Lösungen (inklusive die in diesem Abschnitt besprochenen) ist nicht einmal bekannt, ob ein lokales Optimum in polynomieller Zeit berechnet werden kann; siehe Aufgabe 13. Das von Aarts und Lenstra [1997] herausgegebene Buch enthält weitere Beispiele von Heuristiken für die lokale Suche. Ferner besprechen Michiels, Aarts und Korst [2007] weitere theoretische Resultate zur lokalen Suche.

21.4 Das Traveling-Salesman-Polytop

Dantzig, Fulkerson und Johnson [1954] waren die Ersten, die eine TSP-Instanz von nicht trivialer Größe optimal lösten. Sie lösten zunächst eine LP-Relaxierung einer

geeigneten Formulierung des Problems als ganzzahliges LP und fügten dann schrittweise Schnittebenen hinzu. Damit begann die Analyse des Traveling-Salesman-Polytops:

Definition 21.21. Für $n \geq 3$ bezeichnen wir mit $Q(n)$ das **Traveling-Salesman-Polytop**, d. h. die konvexe Hülle der Inzidenzvektoren der Touren in dem vollständigen Graphen K_n .

Obwohl keine vollständige Beschreibung des Traveling-Salesman-Polytops bekannt ist, gibt es einige interessante Resultate, von denen manche sogar von praktischem Nutzen sind. Da $\sum_{e \in \delta(v)} x_e = 2$ für alle $v \in V(K_n)$ und $x \in Q(n)$, ist die Dimension von $Q(n)$ höchstens gleich $|E(K_n)| - |V(K_n)| = \binom{n}{2} - n = \frac{n(n-3)}{2}$. Um zu beweisen, dass $\dim(Q(n))$ in der Tat genau gleich $\frac{n(n-3)}{2}$ ist, benötigen wir das folgende graphentheoretische Lemma:

Lemma 21.22. Für jedes $k \geq 1$ haben wir:

- (a) Die Kantenmenge von K_{2k+1} kann in k Touren partitioniert werden;
- (b) Die Kantenmenge von K_{2k} kann in $k-1$ Touren und ein perfektes Matching partitioniert werden.

Beweis: (a): Es seien die Knoten mit $0, \dots, 2k-1, x$ durchnummeriert. Betrachte die folgenden Touren für $i = 0, \dots, k-1$ (alles soll modulo $2k$ gelten):

$$\begin{aligned} T_i &= (x, i, i+1, i-1, i+2, i-2, i+3, \dots, \\ &\quad i-k+2, i+k-1, i-k+1, i+k, x). \end{aligned}$$

Ein Beispiel ist in Abb. 21.8 gegeben. Da $|E(K_{2k+1})| = k(2k+1)$, genügt es zu zeigen, dass diese Touren paarweise kantendisjunkt sind. Dies ist klar bezüglich der Kanten, mit denen x inzident ist. Ferner sieht man leicht, dass $a+b \in \{2i, 2i+1\}$ für $\{a, b\} \in E(T_i)$ mit $a, b \neq x$.

(b): Es seien die Knoten mit $0, \dots, 2k-2, x$ durchnummeriert. Betrachte die folgenden Touren für $i = 0, \dots, k-2$ (alles soll modulo $2k-1$ gelten):

$$\begin{aligned} T_i &= (x, i, i+1, i-1, i+2, i-2, i+3, \dots, \\ &\quad i+k-2, i-k+2, i+k-1, i-k+1, x). \end{aligned}$$

Dasselbe Argument wie oben zeigt, dass diese Touren paarweise kantendisjunkt sind. Entfernen wir sie, so erhalten wir einen 1-regulären Graphen, der somit ein perfektes Matching liefert. \square

Satz 21.23. (Grötschel und Padberg [1979]) Es gilt

$$\dim(Q(n)) = \frac{n(n-3)}{2}.$$

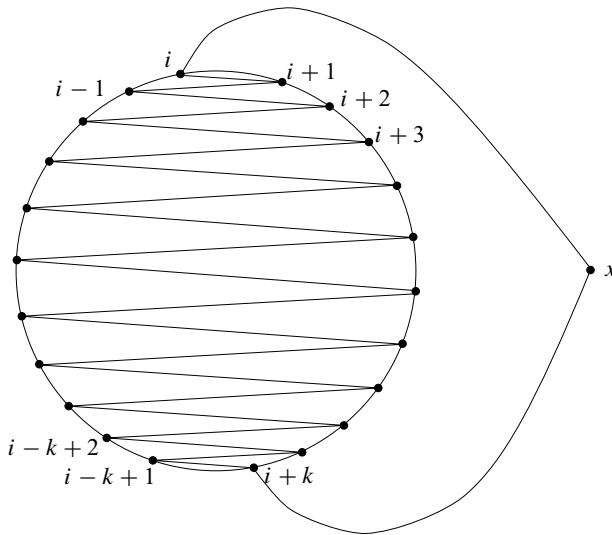


Abbildung 21.8.

Beweis: Die Aussage ist für $n = 3$ trivial. Sei nun $n \geq 4$ und $v \in V(K_n)$ ein beliebiger Knoten.

Fall 1: Es sei n gerade, etwa $n = 2k + 2$ für eine ganze Zahl $k \geq 1$. Nach Lemma 21.22(a) ist $K_n - v$ die Vereinigung von k paarweise kantendisjunkten Touren T_0, \dots, T_{k-1} . Es gehe T_{ij} aus T_i hervor, indem wir die j -te Kante $\{a, b\}$ durch die zwei Kanten $\{a, v\}, \{v, b\}$ ersetzen ($i = 0, \dots, k-1; j = 1, \dots, n-1$). Betrachte die Matrix, deren Zeilen die Inzidenzvektoren dieser $k(n-1)$ Touren sind. Dann bilden die Spalten, die denjenigen Kanten entsprechen, mit denen v nicht inzident ist, eine quadratische Matrix

$$\begin{pmatrix} A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ 0 & 0 & A & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix}, \quad \text{wobei } A = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & \cdots & 0 \end{pmatrix}.$$

Da diese Matrix nichtsingulär ist, sind die Inzidenzvektoren der $k(n-1)$ Touren linear unabhängig, womit $\dim(Q(n)) \geq k(n-1) - 1 = \frac{n(n-3)}{2}$ folgt.

Fall 2: Es sei n ungerade, etwa $n = 2k + 3$ für eine ganze Zahl $k \geq 1$. Nach Lemma 21.22(b) ist $K_n - v$ die Vereinigung von k paarweise kantendisjunkten Touren und einem perfekten Matching M . Aus den Touren konstruieren wir $k(n-1)$ Touren in K_n wie in Fall 1. Nun vervollständigen wir das perfekte Matching M auf beliebige Weise zu einer Tour T in K_{n-1} . Für jede Kante $e = \{a, b\}$ von M ersetzen wir e in T durch die zwei Kanten $\{a, v\}$ und $\{v, b\}$. Auf diese Weise erhalten wir weitere $k+1$ Touren. Wie oben sieht man, dass die Inzidenzvektoren

aller $k(n-1) + k + 1 = kn + 1$ Touren linear unabhängig sind, womit wir bewiesen haben, dass $\dim(Q(n)) \geq kn + 1 - 1 = \frac{n(n-3)}{2}$. \square

Die ganzzahligen Punkte von $Q(n)$, d. h. die Touren, können auf schöne Weise charakterisiert werden:

Proposition 21.24. *Die Inzidenzvektoren der Touren in K_n sind genau die ganzzahligen Vektoren x , welche die folgenden Bedingungen erfüllen:*

$$0 \leq x_e \leq 1 \quad (e \in E(K_n)); \quad (21.1)$$

$$\sum_{e \in \delta(v)} x_e = 2 \quad (v \in V(K_n)); \quad (21.2)$$

$$\sum_{e \in E(K_n[X])} x_e \leq |X| - 1 \quad (\emptyset \neq X \subset V(K_n)). \quad (21.3)$$

Beweis: Offensichtlich erfüllt der Inzidenzvektor einer jeden Tour diese Bedingungen. Andererseits ist jeder ganzzahlige Vektor, der (21.1) und (21.2) erfüllt, der Inzidenzvektor eines perfekten einfachen 2-Matchings, d. h. der Vereinigung paarweise knotendisjunkter Kreise, die sämtliche Knoten überdecken. Die Bedingungen (21.3) verhindern Kreise mit weniger als n Kanten. \square

Die Bedingungen (21.3) sind bekannt als **Subtour-Ungleichungen** und das durch (21.1), (21.2) und (21.3) definierte Polytop heißt das **Subtouren-Polytop**. Im Allgemeinen ist das Subtouren-Polytop nicht ganzzahlig; ein Gegenbeispiel wird in Abb. 21.9 gezeigt (die nicht eingezeichneten Kanten haben Gewicht 3): Die kürzeste Tour hat die Länge 10, aber die optimale gebrochene Lösung ($x_e = 1$, falls e Gewicht 1 hat, und $x_e = \frac{1}{2}$, falls e Gewicht 2 hat) hat Gesamtgewicht 9.

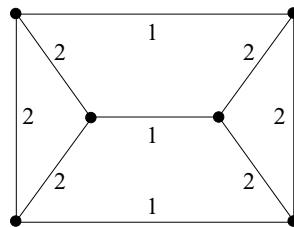


Abbildung 21.9.

Die folgenden drei äquivalenten Charakterisierungen des Subtouren-Polytops werden sich als nützlich erweisen:

Proposition 21.25. *Sei $V(K_n) = \{1, \dots, n\}$ und $x \in [0, 1]^{E(K_n)}$ mit $\sum_{e \in \delta(v)} x_e = 2$ für alle $v \in V(K_n)$. Dann sind die folgenden drei Aussagen äquivalent:*

$$\sum_{e \in E(K_n[X])} x_e \leq |X| - 1 \quad (\emptyset \neq X \subset V(K_n)); \quad (21.3)$$

$$\sum_{e \in E(K_n[X])} x_e \leq |X| - 1 \quad (\emptyset \neq X \subseteq V(K_n) \setminus \{1\}); \quad (21.4)$$

$$\sum_{e \in \delta(X)} x_e \geq 2 \quad (\emptyset \neq X \subset V(K_n)). \quad (21.5)$$

Beweis: Für jedes $\emptyset \neq X \subset V(K_n)$ haben wir

$$\begin{aligned} \sum_{e \in \delta(V(K_n) \setminus X)} x_e &= \sum_{e \in \delta(X)} x_e = \sum_{v \in X} \sum_{e \in \delta(v)} x_e - 2 \sum_{e \in E(K_n[X])} x_e \\ &= 2|X| - 2 \sum_{e \in E(K_n[X])} x_e, \end{aligned}$$

woraus die Äquivalenz von (21.3), (21.4) und (21.5) folgt. \square

Korollar 21.26. Das SEPARATIONS-PROBLEM für Subtour-Ungleichungen kann in polynomieller Zeit gelöst werden.

Beweis: Unter Verwendung von (21.5) und Betrachtung von x als Kantenkapazitätenvektor müssen wie entscheiden, ob es einen Schnitt in (K_n, x) mit Kapazität kleiner als 2 gibt. Somit reduziert sich das SEPARATIONS-PROBLEM auf das Problem: Finde einen kapazitätsminimalen Schnitt in einem ungerichteten Graphen mit nichtnegativen Kapazitäten. Nach Satz 8.42 kann dieses Problem in $O(n^3)$ -Zeit gelöst werden. \square

Da jede Tour ein perfektes einfaches 2-Matching ist, enthält die konvexe Hülle aller perfekten einfachen 2-Matchings das Traveling-Salesman-Polytop. Somit haben wir nach Satz 12.3:

Proposition 21.27. Jedes $x \in Q(n)$ erfüllt die Ungleichungen

$$\sum_{e \in E(K_n[X]) \cup F} x_e \leq |X| + \frac{|F| - 1}{2}, \quad X \subseteq V(K_n) \text{ und } F \subseteq \delta(X) \text{ mit } |F| \text{ ungerade.} \quad (21.6)$$

Die Bedingungen (21.6) heißen **2-Matching-Ungleichungen**. Ist F ein Matching, so genügt es, die Ungleichungen (21.6) zu betrachten; die weiteren 2-Matching-Ungleichungen folgen dann (Aufgabe 17). Für die 2-Matching-Ungleichungen kann das SEPARATIONS-PROBLEM nach Satz 12.21 in polynomieller Zeit gelöst werden. Somit können wir eine lineare Funktion auf dem durch (21.1), (21.2), (21.3) und (21.6) definierten Polytop mit der ELLIPSOIDMETHODE (Satz 4.21) in polynomieller Zeit lösen (Aufgabe 16). Die 2-Matching-Ungleichungen werden durch die sogenannten **Kamm-Ungleichungen** verallgemeinert, siehe Abb. 21.10:

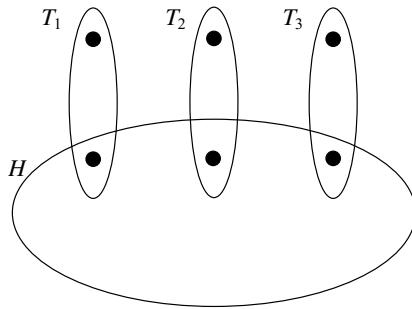


Abbildung 21.10.

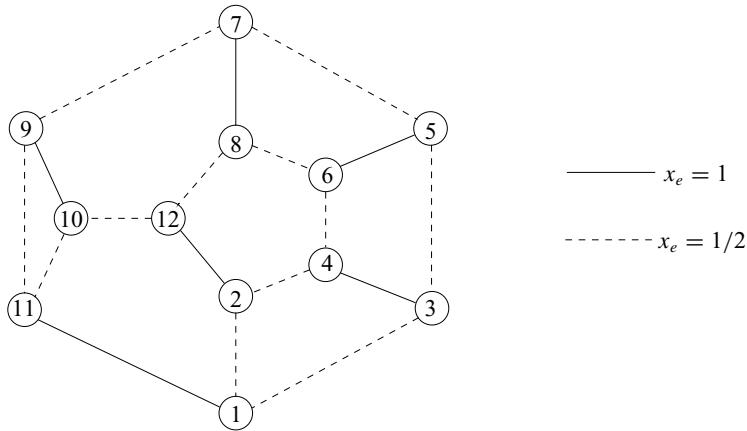


Abbildung 21.11.

Proposition 21.28. (Chvátal [1973], Grötschel und Padberg [1979]) Seien $T_1, \dots, T_s \subseteq V(K_n)$ s paarweise disjunkte Mengen, $s \geq 3$ ungerade und $H \subseteq V(K_n)$ mit $T_i \cap H \neq \emptyset$ und $T_i \setminus H \neq \emptyset$ für $i = 1, \dots, s$. Dann erfüllt jedes $x \in Q(n)$ die Ungleichung

$$\sum_{e \in \delta(H)} x_e + \sum_{i=1}^s \sum_{e \in \delta(T_i)} x_e \geq 3s + 1. \quad (21.7)$$

Beweis: Sei x der Inzidenzvektor irgendeiner Tour. Für jedes $i \in \{1, \dots, s\}$ muss die Tour sowohl in $T_i \setminus H$ als auch in $T_i \cap H$ ankommen und diese beiden Mengen auch wieder verlassen. Demnach folgt

$$\sum_{e \in \delta(T_i)} x_e + \sum_{e \in \delta(H) \cap E(K_n[T_i])} x_e \geq 3.$$

Addieren wir diese s Ungleichungen, so erhalten wir

$$\sum_{e \in \delta(H)} x_e + \sum_{i=1}^s \sum_{e \in \delta(T_i)} x_e \geq 3s.$$

Da die linke Seite eine gerade Zahl ist, folgt der Satz. \square

Die in Abb. 21.11 gezeigte gebrochene Lösung x (Kanten e mit $x_e = 0$ sind nicht eingezeichnet) ist ein Beispiel einer verletzten Kamm-Ungleichung in K_{12} : Betrachte $H = \{1, 2, 3, 4, 5, 6\}$, $T_1 = \{1, 11\}$, $T_2 = \{2, 12\}$ und $T_3 = \{5, 6, 7, 8\}$. Man prüft leicht nach, dass die entsprechende Kamm-Ungleichung verletzt wird. Beachte, dass die Ungleichungen (21.1), (21.2), (21.3) und (21.6) erfüllt sind und dass x ein Optimum bezüglich der Gewichte $c(e) := 1 - x_e$ (Gesamtgewicht 3) ist, während die optimalen Touren Gewicht $\frac{7}{2}$ haben.

Wir erwähnen hier nur noch eine weitere Klasse von Ungleichungen: die **Cliquenbaum-Ungleichungen**:

Satz 21.29. (Grötschel und Pulleyblank [1986]) *Seien H_1, \dots, H_r paarweise disjunkte Teilmengen von $V(G)$ (die Griffe) und T_1, \dots, T_s ($s \geq 1$) paarweise disjunkte nichtleere echte Teilmengen von $V(G)$ (die Zinken), so dass*

- für jeden Griff gilt, dass die Anzahl der Zinken, mit denen der Griff nichleeren Durchschnitt hat, ungerade und mindestens gleich 3 ist;
- jede Zinke mindestens einen zu keinem Griff gehörenden Knoten enthält;
- $G := K_n[H_1 \cup \dots \cup H_r \cup T_1 \cup \dots \cup T_s]$ zusammenhängend ist, aber $G - (T_i \cap H_j)$ immer dann unzusammenhängend ist, wenn $T_i \cap H_j \neq \emptyset$.

Sei t_j die Anzahl der Griffe, mit denen T_j nichleeren Durchschnitt hat ($j = 1, \dots, s$). Dann erfüllt jedes $x \in Q(n)$ die Ungleichung

$$\sum_{i=1}^r \sum_{e \in E(K_n[H_i])} x_e + \sum_{j=1}^s \sum_{e \in E(K_n[T_j])} x_e \leq \sum_{i=1}^r |H_i| + \sum_{j=1}^s (|T_j| - t_j) - \frac{s+1}{2}. \quad (21.8)$$

Den Beweis dieses Satzes führen wir hier nicht, da er recht technisch ist. Es sind (21.3) und (21.6) spezielle Cliquenbaum-Ungleichungen (Aufgabe 18). Sie sind weiter verallgemeinert worden, z. B. zu Bipartitions-Ungleichungen von Boyd und Cunningham [1991]. Für das SEPARATIONS-PROBLEM für die Cliquenbaum-Ungleichungen (21.8) mit einer festen Anzahl von Griffen und Zinken gibt es einen polynomiellen Algorithmus (Carr [1997]), aber für allgemeine Cliquenbaum-Ungleichungen ist keiner bekannt. Sogar für das SEPARATIONS-PROBLEM für Kamm-Ungleichungen ist kein polynomieller Algorithmus bekannt.

All die Ungleichungen (21.1), (21.4) (beschränkt auf den Fall $3 \leq |X| \leq n-3$) und (21.6) (beschränkt auf den Fall F ist ein Matching) definieren paarweise verschiedene Facetten von $Q(n)$ (für $n \geq 6$). Der Beweis, dass die trivialen Ungleichungen (21.1) Facetten definieren, besteht darin, $\dim(Q(n))$ linear unabhängige Touren mit $x_e = 1$ (und ebenso mit $x_e = 0$) für irgendeine feste Kante e zu finden. Der Beweis ähnelt dem von Satz 21.23; wir verweisen auf Grötschel und Padberg [1979]. Sogar all die Ungleichungen (21.8) definieren Facetten von $Q(n)$

($n \geq 6$). Der Beweis ist recht kompliziert, siehe Grötschel und Padberg [1979] oder Grötschel und Pulleyblank [1986].

Die Anzahl der Facetten von $Q(n)$ wächst schnell: Schon $Q(10)$ hat mehr als 50 Milliarden Facetten. Jedes Polyeder mit Projektion $Q(n)$ (d. h. jede erweiterte Formulierung) hat $2^{\Omega(n^{1/2})}$ Facetten (Fiorini et al. [2015]). Diese Schranke wurde von Rothvoß [2017] auf $2^{\Omega(n)}$ verbessert. Jedes semidefinite Programm mit Projektion $Q(n)$ hat $2^{\Omega(n^{1/13})}$ Variablen (Lee, Raghavendra und Steurer [2015]).

Es ist keine vollständige Beschreibung von $Q(n)$ bekannt und es ist sehr unwahrscheinlich, dass es eine solche gibt. Betrachte das folgende Problem:

TSP-FACETTEN

Instanz: Eine ganze Zahl $n \geq 3$ und eine ganzzahlige Ungleichung $ax \leq \beta$.

Frage: Definiert die gegebene Ungleichung eine Facette von $Q(n)$?

Das folgende Resultat zeigt, dass es unwahrscheinlich ist, dass es eine vollständige Beschreibung des Traveling-Salesman-Polytops gibt:

Satz 21.30. (Karp und Papadimitriou [1982]) *Ist TSP-FACETTEN in NP, so gilt $NP = coNP$.*

Ferner ist das folgende Problem NP -vollständig: Entscheide, ob zwei gegebene Knoten von $Q(n)$ benachbart sind, d. h. ob es eine Seitenfläche der Dimension 1 gibt, die beide enthält (Papadimitriou [1978]).

21.5 Untere Schranken

Angenommen, wir haben eine Tour heuristisch gefunden, z. B. mit dem LIN-KERNIGHAN-ALGORITHMUS. Wir wissen nicht im Voraus, ob diese Tour optimal ist oder wenigstens in der Nähe des Optimums liegt. Kann man auf irgendeine Weise feststellen, dass unsere Tour um höchstens x Prozent vom Optimum entfernt ist? Anders ausgedrückt: Gibt es eine untere Schranke für das Optimum?

Untere Schranken kann man finden, indem man eine LP-Relaxierung einer Formulierung des TSP als ganzzahliges LP betrachtet, z. B. mittels der Ungleichungen (21.1), (21.2), (21.3) und (21.6). Dieses LP lässt sich jedoch nicht leicht lösen (obwohl es einen polynomiellen Algorithmus mittels der ELLIPSOIDMETHODE gibt). Eine vernünftigere untere Schranke erhält man, wenn man nur (21.1), (21.2) und (21.6) nimmt, d. h. indem man ein perfektes einfaches 2-Matching minimalen Gewichtes findet (siehe Aufgabe 1, Kapitel 12).

Die bisher effizienteste Methode verwendet Lagrange-Relaxierung (siehe Abschnitt 5.6). Held und Karp [1970, 1971] waren die Ersten, die Lagrange-Relaxierung auf das TSP anwendeten. Ihr Ansatz basiert auf dem folgenden Begriff:

Definition 21.31. Gegeben sei ein vollständiger Graph K_n mit $V(K_n) = \{1, \dots, n\}$. Ein **1-Baum** ist ein Graph bestehend aus einem aufspannenden Baum auf den Knoten $\{2, \dots, n\}$ und zwei Kanten, mit denen der Knoten 1 inzident ist.

Die Touren sind gerade die 1-Bäume T mit $|\delta_T(i)| = 2$ für $i = 1, \dots, n$. Wir kennen aufspannende Bäume gut und 1-Bäume sind nicht sehr anders; z.B. haben wir:

Proposition 21.32. *Die konvexe Hülle der Inzidenzvektoren aller 1-Bäume ist die Menge der Vektoren $x \in [0, 1]^{E(K_n)}$ mit*

$$\sum_{e \in E(K_n)} x_e = n, \quad \sum_{e \in \delta(1)} x_e = 2, \quad \sum_{e \in E(K_n[X])} x_e \leq |X| - 1 \quad (\emptyset \neq X \subseteq \{2, \dots, n\}).$$

Beweis: Die Aussage folgt direkt mit Satz 6.13. □

Wir weisen darauf hin, dass eine lineare Zielfunktion leicht über der Menge der 1-Bäume optimiert werden kann: Man braucht nur einen aufspannenden Baum minimalen Gewichtes auf $\{2, \dots, n\}$ zu finden (siehe Abschnitt 6.1) und die zwei leichtesten Kanten, mit denen der Knoten 1 inzident ist, hinzuzufügen. Lagrange-Relaxierung liefert nun die folgende untere Schranke:

Proposition 21.33. (Held und Karp [1970]) *Gegeben sei eine Instanz (K_n, c) des TSP mit $V(K_n) = \{1, \dots, n\}$ und $\lambda = (\lambda_2, \dots, \lambda_n) \in \mathbb{R}^{n-1}$. Dann ist*

$$LR(K_n, c, \lambda) := \min \left\{ c(E(T)) + \sum_{i=2}^n (|\delta_T(i)| - 2) \lambda_i : T \text{ ist ein 1-Baum} \right\}$$

eine untere Schranke für die Länge einer optimalen Tour, und diese kann in der zur Lösung des MINIMUM-SPANNING-TREE-PROBLEMS auf $n - 1$ Knoten benötigten Zeit berechnet werden.

Beweis: Eine optimale Tour T ist ein 1-Baum mit $|\delta_T(i)| = 2$ für alle i , also ist $LR(K_n, c, \lambda)$ eine untere Schranke. Für gegebenes $\lambda = (\lambda_2, \dots, \lambda_n)$ wählen wir λ_1 beliebig und ersetzen die Gewichte c durch $c'(\{i, j\}) := c(\{i, j\}) + \lambda_i + \lambda_j$ ($1 \leq i < j \leq n$). Jetzt müssen wir nur noch einen 1-Baum minimalen Gewichtes bezüglich c' finden. □

Beachte, dass die Lagrange-Multiplikatoren λ_i ($i = 2, \dots, n$) nicht auf die nichtnegativen Zahlen beschränkt sind, da die zusätzlichen Bedingungen $|\delta_T(i)| = 2$ Gleichungen sind. Die λ_i können mittels eines Subgradienten-Verfahrens bestimmt werden; siehe Abschnitt 5.6. Der größtmögliche Wert

$$HK(K_n, c) := \max\{LR(K_n, c, \lambda) : \lambda \in \mathbb{R}^{n-1}\}$$

(das Lagrange-Dual) heißt die **Held-Karp-Schranke** für (K_n, c) . Wir haben:

Satz 21.34. (Held und Karp [1970]) *Für jede Instanz (K_n, c) des TSP mit $V(K_n) = \{1, \dots, n\}$ gilt*

$$\begin{aligned}
HK(K_n, c) = \min \left\{ cx : 0 \leq x_e \leq 1 \quad (e \in E(K_n)), \right. \\
\sum_{e \in \delta(v)} x_e = 2 \quad (v \in V(K_n)), \\
\left. \sum_{e \in E(K_n[I])} x_e \leq |I| - 1 \quad (\emptyset \neq I \subseteq \{2, \dots, n\}) \right\}.
\end{aligned}$$

Beweis: Die Aussage folgt direkt mit Satz 5.37 und Proposition 21.32. \square

Mit anderen Worten, die Held-Karp-Schranke ist gleich dem optimalen LP-Zielfunktionswert über dem Subtouren-Polytop (siehe Proposition 21.25). Dies hilft uns bei der Abschätzung der Qualität der Held-Karp-Schranke für das METRISCHE TSP. Wir verwenden auch wieder die Idee von CHRISTOFIDES' ALGORITHMUS:

Satz 21.35. (Wolsey [1980]) *Für jede Instanz des METRISCHEN TSP ist die Held-Karp-Schranke mindestens gleich $\frac{2}{3}$ der Länge einer optimalen Tour.*

Beweis: Sei (K_n, c) eine Instanz des METRISCHEN TSP und T ein 1-Baum minimalen Gewichtes in (K_n, c) . Es gilt

$$c(E(T)) = LR(K_n, c, 0) \leq HK(K_n, c).$$

Es bestehe $W \subseteq V(K_n)$ aus den Knoten mit ungeradem Grad in T . Da jeder Vektor x im Subtouren-Polytop von (K_n, c) die Ungleichung $\sum_{e \in \delta(X)} x_e \geq 2$ für alle $\emptyset \neq X \subset V(K_n)$ erfüllt, enthält das Polyeder

$$\left\{ x : x_e \geq 0 \text{ für alle } e \in E(K_n), \sum_{e \in \delta(X)} x_e \geq 2 \text{ für alle } X \text{ mit } |X \cap W| \text{ ungerade} \right\}$$

das Subtouren-Polytop. Mit Satz 21.34 folgt dann

$$\begin{aligned}
& \min \left\{ cx : x_e \geq 0 \text{ für alle } e \in E(K_n), \sum_{e \in \delta(X)} x_e \geq 1 \text{ für alle } X \text{ mit } |X \cap W| \text{ ungerade} \right\} \\
& \leq \frac{1}{2} HK(K_n, c).
\end{aligned}$$

Beachte jedoch: Nach Satz 12.18 ist die linke Seite das kleinstmögliche Gewicht eines W -Joins J in (K_n, c) . Somit gilt $c(E(T)) + c(J) \leq \frac{3}{2} HK(K_n, c)$. Da der Graph $G := (V(K_n), E(T) \cup J)$ zusammenhängend und eulersch ist, ist $\frac{3}{2} HK(K_n, c)$ eine obere Schranke für die Länge einer optimalen Tour (nach Lemma 21.3). \square

Ein anderer Beweis stammt von Shmoys und Williamson [1990]. Es ist nicht bekannt, ob diese Schranke die bestmögliche ist. Für das in Abb. 21.9 auf S. 645

dargestellte Beispiel (nicht eingezeichnete Kanten haben Gewicht 3) ist die Held-Karp-Schranke (gleich 9) kleiner als die Länge einer optimalen Tour (gleich 10). Es gibt Instanzen des METRISCHEN TSP mit $\frac{HK(K_n,c)}{OPT(K_n,c)}$ beliebig nahe an $\frac{3}{4}$ (Aufgabe 19). Anders ausgedrückt: Die Ganzzahligkeitslücke des Subtouren-Polytops für das METRISCHE TSP liegt zwischen $\frac{4}{3}$ und $\frac{3}{2}$.

In der Praxis ist die Held-Karp-Schranke normalerweise viel besser; siehe z. B. Johnson, McGeoch und Rothberg [1996] oder Applegate et al. [2006].

21.6 Branch-and-Bound

Branch-and-Bound ist ein Verfahren, mit dem man eine vollständige Enumeration aller Lösungen simulieren kann, ohne sie alle einzeln betrachten zu müssen. Für viele *NP*-schwere kombinatorische Optimierungsprobleme ist Branch-and-Bound der beste Ansatz, um zu einer optimalen Lösung zu gelangen. Er stammt von Land und Doig [1960] und wurde zum ersten Mal von Little et al. [1963] auf das TSP angewendet.

Um den BRANCH-AND-BOUND Ansatz auf ein kombinatorisches Optimierungsproblem (etwa eine Minimierung) anzuwenden, benötigt man zwei Schritte:

- Der „Branch“-Schritt: Eine gegebene Teilmenge der möglichen Lösungen (Touren im Falle des TSP) kann in mindestens zwei nichtleere Teilmengen partitioniert werden;
- Der „Bound“-Schritt: Für eine durch Branch-Iterationen gewonnene Teilmenge kann eine untere Schranke für die Kosten irgendeiner in ihr liegenden Lösung berechnet werden.

Das allgemeine Verfahren läuft wie folgt ab:

BRANCH-AND-BOUND

Input: Eine Instanz eines Minimierungsproblems.

Output: Eine optimale Lösung S^* .

- ① Setze den Anfangsbauum $T := (\{\mathcal{S}\}, \emptyset)$ fest, wobei \mathcal{S} die Menge der zulässigen Lösungen ist.
Markiere \mathcal{S} als aktiv.
Setze die anfängliche obere Schranke $U := \infty$ fest (oder wende eine Heuristik an, um einen besseren Wert für die obere Schranke zu erhalten).
- ② Wähle einen aktiven Knoten X des Baumes T (gibt es keinen solchen, dann **stop**).
Markiere X als nicht aktiv.
(„Branch“) Bestimme eine Partition $X = X_1 \dot{\cup} \dots \dot{\cup} X_t$.

③ **For** jedes $i = 1, \dots, t$ **do**:

(„Bound“) Bestimme eine untere Schranke L für die Kosten einer jeden Lösung in X_i .

If $|X_i| = 1$ (etwa $X_i = \{S\}$) und $\text{cost}(S) < U$ **then**:

Setze $U := \text{cost}(S)$ und $S^* := S$.

If $|X_i| > 1$ und $L < U$ **then**:

Setze $T := (V(T) \cup \{X_i\}, E(T) \cup \{\{X, X_i\}\})$ und markiere X_i als aktiv.

④ **Go to** ②.

Es sollte klar sein, dass das obige Verfahren immer eine optimale Lösung findet. Die Implementierung (und die Effizienz) hängen natürlich stark vom aktuellen Problem ab. Wir werden hier eine mögliche Implementierung für das TSP besprechen.

Der einfachste Weg, den Branch-Schritt zu gestalten, ist: Wähle eine Kante e und setze $X = X_e \cup (X \setminus X_e)$, wobei X_e die Menge derjenigen Lösungen in X ist, welche die Kante e enthalten. Dann können wir jeden Knoten X des Baumes in der folgenden Form angeben:

$$\mathcal{S}_{A,B} = \{S \in \mathcal{S} : A \subseteq S, B \cap S = \emptyset\} \quad \text{für irgendwelche } A, B \subseteq E(G).$$

Für diese $X = \mathcal{S}_{A,B}$ kann man das TSP mit der zusätzlichen Bedingung, dass alle Kanten von A , aber keine von B , zu der Tour gehören, in der Form eines TSP ohne Nebenbedingungen schreiben, indem man die Gewichte c auf geeignete Weise modifiziert: Setze

$$c'_e := \begin{cases} c_e & \text{für } e \in A \\ c_e + C & \text{für } e \notin A \cup B \\ c_e + 2C & \text{für } e \in B \end{cases}$$

mit $C := \sum_{e \in E(G)} c_e + 1$. Dann sind die Touren in $\mathcal{S}_{A,B}$ genau diejenigen Touren, deren modifiziertes Gewicht weniger als $(n+1-|A|)C$ ist. Ferner ist die Differenz zwischen dem ursprünglichen und dem modifizierten Gewicht irgendeiner Tour in $\mathcal{S}_{A,B}$ genau gleich $(n-|A|)C$.

Die Held-Karp-Schranke (siehe Abschnitt 21.5) kann dazu verwendet werden, den Bound-Schritt zu implementieren.

Mit dem obigen BRANCH-AND-BOUND Verfahren für das TSP sind recht große Instanzen des TSP (bis zu ca. 100 Städten) gelöst worden.

BRANCH-AND-BOUND wird oft auch zur Lösung von ganzzahligen LPs herangezogen, insbesondere wenn die Variablen binär sind (d.h. nur die Werte 0 oder 1 annehmen können). Hier besteht der am nächsten liegende Branch-Schritt darin, dass man eine Variable wählt und die beiden Werte ausprobiert. Eine untere Schranke kann man leicht dadurch erhalten, dass man die LP-Relaxierung löst.

Im ungünstigsten Fall (worst case) ist BRANCH-AND-BOUND nicht besser als die vollständige Enumeration aller Lösungen. In der Praxis hängt die Effizienz

nicht nur davon ab, wie die Branch- und Bound-Schritte implementiert werden. Es ist auch wichtig, eine gute Strategie für die Wahl des aktiven Knotens X in ② des Algorithmus zu haben. Ferner kann eine gute Heuristik am Anfang (und somit eine gute anfängliche obere Schranke) sehr hilfreich sein, um den Branch-and-Bound-Baum T klein zu halten.

Zur Lösung einer Instanz des TSP wird BRANCH-AND-BOUND oft mit einer Schnittebenenmethode (siehe Abschnitt 5.5) verbunden, basierend auf den Resultaten von Abschnitt 21.4. Dazu geht man folgendermaßen vor. Da wir eine exponentiell wachsende Anzahl von Nebenbedingungen haben (die $Q(n)$ nicht einmal vollständig beschreiben), lösen wir zunächst das LP

$$\min \left\{ cx : 0 \leq x_e \leq 1 \ (e \in E(K_n)), \sum_{e \in \delta(v)} x_e = 2 \ (v \in V(K_n)) \right\},$$

d. h. mit den Nebenbedingungen (21.1) und (21.2). Dieses Polyeder enthält die perfekten einfachen 2-Matchings als ganzzahlige Vektoren. Angenommen, wir haben eine Lösung x^* des obigen LP. Dann unterscheiden wir drei Fälle:

- (a) Es ist x^* der Inzidenzvektor einer Tour;
- (b) Wir finden eine verletzte Subtour-Ungleichung (21.3), 2-Matching-Ungleichung (21.6), Kamm-Ungleichung (21.7) oder Cliquesbaum-Ungleichung (21.8);
- (c) Wir finden keine verletzte Nebenbedingung (insbesondere keine verletzte Subtour-Ungleichung), aber x^* ist nicht ganzzahlig.

Ist x^* ganzzahlig, aber nicht der Inzidenzvektor einer Tour, so gibt es nach Proposition 21.24 eine verletzte Subtour-Ungleichung.

Im Fall (a) sind wir fertig. Im Fall (b) fügen wir einfach die verletzte Ungleichung (oder eventuell die verletzten Ungleichungen) unserem LP hinzu und lösen das neue LP. Im Fall (c) haben wir nur eine (i. A. sehr gute) untere Schranke für die Länge einer Tour. Mit dieser Schranke (und der gebrochenen Lösung) können wir jetzt ein BRANCH-AND-BOUND Verfahren beginnen. Wegen der bestmöglichen unteren Schranke hoffen wir, viele der Variablen vorab festlegen zu können und dadurch die Anzahl der zu einer optimalen Lösung führenden Branch-Schritte erheblich zu reduzieren. Ferner können wir in jedem Knoten des Branch-and-Bound-Baumes wieder nach verletzten Ungleichungen suchen.

Dieses Methode heißt **Branch-and-Cut** und mit ihr wurden Instanzen des TSP mit mehr als 10 000 Städten optimal gelöst. Natürlich sind viele weitere hier nicht beschriebene ausgeklügelte Ideen notwendig, um eine effiziente Implementierung zu erhalten. Insbesondere sind gute Heuristiken zur Entdeckung verletzter Ungleichungen ausschlaggebend. Siehe Applegate et al. [2003, 2006] und Jünger und Naddef [2001] bezüglich weiterer Informationen und Literatur.

Diese Erfolge bei der optimalen Lösung großer Instanzen stehen im Gegensatz zu den schlechten Worst-Case-Laufzeiten. Woeginger [2002] hat einen Überblick über subexponentiell exakte Algorithmen für NP-schwere Probleme zusammengestellt; siehe auch Aufgabe 1.

Aufgaben

1. Man beschreibe einen exakten Algorithmus für das TSP mittels dynamischer Optimierung. Es seien die Knoten (Städte) mit $1, \dots, n$ durchnummeriert und es bezeichne $\gamma(A, x)$ die kleinstmöglichen Kosten eines $1-x$ -Weges P mit $V(P) = A \cup \{1\}$ für alle $A \subseteq \{2, 3, \dots, n\}$ und $x \in A$. Die Idee ist es nun, all diese Zahlen $\gamma(A, x)$ zu berechnen. Man vergleiche die Laufzeit dieses Algorithmus mit der naiven Enumeration aller Touren.

(Bellman [1962], Held und Karp [1962])

Bemerkung: Dies ist der exakte TSP-Algorithmus mit der bislang besten Worst-Case-Laufzeit. Für das EUKLIDISCHE TSP haben Hwang, Chang und Lee [1993] einen planaren Separatoren benutzenden exakten Algorithmus mit sub-exponentieller Laufzeit $O(c\sqrt{n} \log n)$ beschrieben.

2. Es seien die n Städte einer Instanz des TSP in m Gruppen aufgeteilt, mit der Eigenschaft, dass die Distanz zwischen zwei Städten genau dann gleich Null ist, wenn sie beide derselben Gruppe angehören.

- (a) Man beweise, dass es eine optimale Tour mit höchstens $m(m - 1)$ Kanten positiven Gewichtes gibt.
- (b) Man beweise, dass ein solches TSP mit festem m in polynomieller Zeit gelöst werden kann.

(Triesch, Nolles und Vygen [1994])

3. Man betrachte das folgende Problem. Ein von einem bestimmten Depot d_1 losfahrender LKW muss eine Anzahl von Kunden c_1, \dots, c_n besuchen und schließlich nach d_1 zurückkehren. Zwischen zwei Kundenbesuchen muss er eines der Depots d_1, \dots, d_k ansteuern. Die Distanzen zwischen den Kunden und Depots seien nichtnegativ und symmetrisch. Man möchte eine kürzeste Tour finden.

- (a) Man zeige, dass dieses Problem NP -vollständig ist.
- (b) Man zeige, dass es für festes k in polynomieller Zeit gelöst werden kann.

(*Hinweis:* Man benutze Aufgabe 2.)

(Triesch, Nolles und Vygen [1994])

4. Man betrachte das die Dreiecksungleichung erfüllende ASYMMETRISCHE TSP: Gegeben sei eine Zahl $n \in \mathbb{N}$ und Distanzen $c((i, j)) \in \mathbb{R}_+$ für $i, j \in \{1, \dots, n\}$, $i \neq j$, die die Dreiecksungleichung $c((i, j) + c((j, k)) \geq c((i, k))$ für alle paarweise verschiedenen $i, j, k \in \{1, \dots, n\}$ erfüllen. Man finde eine Permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$, die $\sum_{i=1}^{n-1} c((\pi(i), \pi(i+1))) + c((\pi(n), \pi(1)))$ minimiert.

Man beschreibe einen Algorithmus, der immer eine Lösung mit Kosten höchstens gleich $\log n$ mal dem Optimum findet.

Hinweis: Zunächst bestimme man einen Digraphen H mit $V(H) = \{1, \dots, n\}$ und $|\delta_H^-(v)| = |\delta_H^+(v)| = 1$ für alle $v \in V(H)$ und kleinstmöglichen Kosten $c(E(H))$. Dann kontrahiere man die Kreise in H und iteriere.

(Frieze, Galbiati und Maffioli [1982])

Bemerkung: Svensson, Tarnawski und Végh [2017] haben einen Approximationsalgorithmus mit konstanter Gütegarantie entwickelt.

5. Man beschreibe einen polynomiellen Algorithmus, der jede Instanz eines TSP, die der metrische Abschluss eines gewichteten Baumes ist, optimal löst.
- * 6. Man finde Instanzen des EUKLIDISCHEN TSP, für die der DOPPELBAUM-ALGORITHMUS eine Tour als Output liefert, deren Länge beliebig nahe bei zweimal dem Optimum liegt.
7. Sei G ein vollständiger bipartiter Graph mit Bipartition $V(G) = A \cup B$, wobei $|A| = |B|$. Sei $c : E(G) \rightarrow \mathbb{R}_+$ eine Kostenfunktion mit $c(\{a, b\}) + c(\{b, a'\}) + c(\{a', b'\}) \geq c(\{a, b'\})$ für alle $a, a' \in A$ und $b, b' \in B$. Man möchte einen Hamilton-Kreis in G mit minimalen Kosten finden. Dieses Problem heißt das METRISCHE BIPARTITE TSP.
- Man beweise für jedes k : Gibt es einen k -Approximationsalgorithmus für das METRISCHE BIPARTITE TSP, so gibt es auch einen k -Approximationsalgorithmus für das METRISCHE TSP.
 - Man finde einen 2-Approximationsalgorithmus für das METRISCHE BIPARTITE TSP. (*Hinweis:* Man kombiniere Aufgabe 27, Kapitel 13, mit der Idee des DOPPELBAUM-ALGORITHMUS.)
- (Frank et al. [1998], Chalasani, Motwani und Rao [1996])
- * 8. Man finde Instanzen des METRISCHEN TSP, für die CHRISTOFIDES' ALGORITHMUS eine Tour als Output liefert, deren Länge beliebig nahe bei $\frac{3}{2}$ mal dem Optimum liegt.
9. Man betrachte das (metrische) s - t -Weg-TSP: Gegeben sei eine Instanz des METRISCHEN TSP und zwei Knoten s und t , gesucht ist ein hamiltonscher s - t -Weg mit minimalem Gewicht.
- Man beschreibe einen 2-Approximationsalgorithmus durch Verallgemeinerung des DOPPELBAUM-ALGORITHMUS.
 - * (b) Man beschreibe einen $\frac{5}{3}$ -Approximationsalgorithmus durch Verallgemeinerung von CHRISTOFIDES' ALGORITHMUS.
- (Hoogeveen [1991])
- Bemerkung:* Traub und Vygen [2018] haben einen $(\frac{3}{2} + \epsilon)$ -Approximationsalgorithmus für beliebiges $\epsilon > 0$ angegeben.
10. Man zeige, dass die Resultate von Abschnitt 21.2 auf das EUKLIDISCHE STEINERBAUM-PROBLEM erweitert werden können. Man beschreibe ein Approximationsschema für dieses Problem.
11. Man beweise: Im LIN-KERNIGHAN-ALGORITHMUS enthält eine Menge X_i für ungerades i mit $i > p_2 + 3$ niemals mehr als ein Element.
12. Man betrachte das folgende Entscheidungsproblem:

ZWEITER HAMILTON-KREIS

Instanz: Ein Graph G und ein Hamilton-Kreis in G .

Frage: Gibt es einen weiteren Hamilton-Kreis in G ?

- (a) Man zeige, dass dieses Problem NP -vollständig ist. (*Hinweis:* Man beachte den Beweis von Lemma 21.18.)

- * (b) Man beweise: Für einen gegebenen 3-regulären Graphen G und $e \in E(G)$ ist die Anzahl der e enthaltenden Hamilton-Kreise gerade.
 - (c) Man zeige, dass ZWEITER HAMILTON-KREIS für 3-reguläre Graphen in P ist. (Dennoch ist für einen gegebenen 3-regulären Graphen G und einen Hamilton-Kreis in G kein polynomieller Algorithmus zur Bestimmung eines weiteren Hamilton-Kreises bekannt.)
13. Sei $(X, (S_x)_{x \in X}, c, \text{goal})$ ein diskretes Optimierungsproblem mit Umgebung $N_x(y) \subseteq S_x$ für $y \in S_x$ und $x \in X$. Angenommen, man kann Folgendes in polynomieller Zeit bewerkstelligen: Man finde für jedes $x \in X$ ein Element von S_x und für jedes $y \in S_x$ ein Element $y' \in N_x(y)$ mit geringeren Kosten, oder entscheide, dass dies nicht möglich ist. Dann sagt man: Das obige Problem, zusammen mit dieser Umgebung, gehört der Klasse *PLS* an. (*PLS* ist die Abkürzung für „polynomielle lokale Suche“). Man beweise: Gibt es ein Problem in *PLS* mit der Eigenschaft: *Das Problem, für eine gegebene Instanz ein lokales Optimum zu berechnen, ist NP-schwer*, so gilt $NP = coNP$.
- Hinweis:* Man konstruiere einen nicht-deterministischen Algorithmus für ein *coNP*-vollständiges Problem.
- (Johnson, Papadimitriou und Yannakakis [1988])
- Bemerkung:* Das TSP ist sowohl mit der k -opt als auch mit der Lin-Kernighan-Umgebung *PLS*-vollständig (Krentel [1989], Papadimitriou [1992]), d. h. kann man ein lokales Optimum in polynomieller Zeit finden, so kann man dies auch für jedes Problem-plus-Umgebung in *PLS* tun (und damit folgt ein weiterer Beweis von Satz 4.18, da der *SIMPLEXALGORITHMUS* korrekt ist).
14. Man zeige: $Q(n) = \{x \in \{0, 1\}^{E(K_n)} : x_e = y_{ij} + y_{ji} (e = \{i, j\} \in E(K_n)), \sum_{j \neq i} y_{ij} = \sum_{j \neq i} y_{ji} = 1 (i = 1, \dots, n), u_j - u_i \geq 1 - n(1 - y_{ij}) (1 \leq i \leq n, 2 \leq j \leq n, i \neq j), y_{ij} \in \{0, 1\} (1 \leq i, j \leq n, i \neq j)\}$.
- (Miller, Tucker und Zemlin [1960])
- Bemerkung:* Kaibel und Weltge [2015] haben bewiesen, dass jede Darstellung von $Q(n)$ zusätzliche Variablen (wie oben und in Aufgabe 15) oder exponentiell viele Nebenbedingungen hat (wie in Proposition 21.24).
15. Man beschreibe ein Polytop P mit einer polynomiellen Anzahl von Variablen und Nebenbedingungen (eine sogenannte erweiterte Formulierung) mit der Eigenschaft, dass die Projektion von P auf einige der Variablen das Subtouren-Polytop ist.
16. Man zeige, dass man jede lineare Funktion über dem durch (21.1), (21.2), (21.3) und (21.6) definierten Polytop optimieren kann.
- Hinweis:* Man verwende Satz 21.23, um die Dimension zu verringern, damit man ein volldimensionales Polytop erhält. Man finde einen inneren Punkt und wende Satz 4.21 an.
17. Man betrachte die 2-Matching-Ungleichungen (21.6) in Proposition 21.27. Man zeige, dass es keine Rolle spielt, ob man zusätzlich verlangt, dass F ein Matching ist: Jeder Vektor im Subtouren-Polytop, der die Ungleichungen (21.6) stets erfüllt, wenn F ein Matching ist, erfüllt alle Ungleichungen (21.6).

18. Man zeige, dass die Subtour-Ungleichungen (21.3), die 2-Matching-Ungleichungen (21.6) und die Kamm-Ungleichungen (21.7) Spezialfälle der Cliquesbaum-Ungleichungen (21.8) sind.
19. Man beweise, dass es Instanzen (K_n, c) des METRISCHEN TSP gibt, für die $\frac{HK(K_n, c)}{\text{OPT}(K_n, c)}$ beliebig nahe bei $\frac{3}{4}$ liegt.
Hinweis: Man ersetze die Kanten mit Gewicht 1 in Abb. 21.9 durch lange Wege und betrachte den metrischen Abschluss.
Bemerkung: Hougardy [2014] hat gezeigt, dass dies sogar für das EUKLIDISCHE TSP gilt.
20. Man betrachte das TSP mit n Städten. Sei c_w^* die Länge einer optimalen Tour bezüglich einer Gewichtsfunktion $w : E(K_n) \rightarrow \mathbb{R}_+$. Man beweise: Gilt $L_1 \leq c_{w_1}^*$ bzw. $L_2 \leq c_{w_2}^*$ für die Gewichtsfunktionen w_1 bzw. w_2 , so folgt $L_1 + L_2 \leq c_{w_1+w_2}^*$, wobei die Summe der zwei Gewichtsfunktionen komponentenweise gilt.
21. Sei T eine optimale Tour für eine Instanz (K_n, c) des METRISCHEN TSP und sei T' eine von T verschiedene kürzeste Tour. Man zeige, dass

$$\frac{c(E(T')) - c(E(T))}{c(E(T))} \leq \frac{2}{n}.$$

(Papadimitriou und Steiglitz [1978])

22. Sei $x \in [0, 1]^{E(K_n)}$ mit $\sum_{e \in \delta(v)} x_e = 2$ für alle $v \in V(K_n)$. Man beweise: Gibt es eine verletzte Subtour-Ungleichung, d. h. eine Menge $S \subset V(K_n)$ mit $\sum_{e \in \delta(S)} x_e < 2$, so gibt es auch eine mit $x_e < 1$ für alle $e \in \delta(S)$.
(Crowder und Padberg [1980])
23. Für eine Familie \mathcal{F} (nicht notwendigerweise paarweise verschiedener) Teilmengen von $\{1, \dots, n\}$ und einen Vektor $x \in \mathbb{R}^{E(K_n)}$ setze man $\mathcal{F}(x) := \sum_{X \in \mathcal{F}} \sum_{e \in \delta(X)} x_e$, und es bezeichne $\mu_{\mathcal{F}}$ das Minimum von $\mathcal{F}(x)$ bezüglich aller Inzidenzvektoren von Touren in K_n . Eine Ungleichung der Form $\mathcal{F}(x) \geq \mu_{\mathcal{F}}$ nennt man eine *Hypergraphen-Ungleichung*. Beispiele hierzu sind (21.5) und (21.7). Man zeige, dass das TSP-Polytop mittels Gradbedingungen und Hypergraphen-Ungleichungen beschrieben werden kann, d. h. es gibt Familien $\mathcal{F}_1, \dots, \mathcal{F}_k$, so dass $Q(n) =$

$$\left\{ x \in \mathbb{R}^{E(K_n)} : \sum_{e \in \delta(v)} x_e = 2 \ (v \in V(K_n)), \mathcal{F}_i(x) \leq \mu_{\mathcal{F}_i} \ (i = 1, \dots, k) \right\}.$$

Hinweis: Man schreibe jede facettenbestimmende Ungleichung um, unter Verwendung der Tatsache, dass $\sum_{e \in \delta(\{v, w\})} x_e = 4 - 2x_{\{v, w\}}$ für jedes die Gradbedingungen erfüllende x gilt.

(Applegate et al. [2006])

Literatur

Allgemeine Literatur:

- Applegate, D.L., Bixby, R., Chvátal, V., und Cook, W.J. [2006]: The Traveling Salesman Problem: A Computational Study. Princeton University Press 2006
- Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., und Schrijver, A. [1998]: Combinatorial Optimization. Wiley, New York 1998, Kapitel 7
- Gutin, G., und Punnen, A.P. [2002]: The Traveling Salesman Problem and Its Variations. Kluwer, Dordrecht 2002
- Jünger, M., Reinelt, G., und Rinaldi, G. [1995]: The traveling salesman problem. In: Handbooks in Operations Research and Management Science; Volume 7; Network Models (M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, Hrsg.), Elsevier, Amsterdam 1995
- Jungnickel, D. [2013]: Graphs, Networks and Algorithms. 4. Aufl. Springer, Berlin 2013, Kapitel 15
- Lawler, E.L., Lenstra J.K., Rinnooy Kan, A.H.G., und Shmoys, D.B. [1985]: The Traveling Salesman Problem. Wiley, Chichester 1985
- Papadimitriou, C.H., und Steiglitz, K. [1982]: Combinatorial Optimization; Algorithms and Complexity. Prentice-Hall, Englewood Cliffs 1982, Abschnitt 17.2, Kapitel 18 und 19
- Reinelt, G. [1994]: The Traveling Salesman; Computational Solutions for TSP Applications. Springer, Berlin 1994
- Vygen, J. [2012]: New approximation algorithms for the TSP. *OPTIMA* 90 (2012), 1–12

Zitierte Literatur:

- Aarts, E., und Lenstra, J.K. [1997]: Local Search in Combinatorial Optimization. Wiley, Chichester 1997
- Applegate, D., Bixby, R., Chvátal, V., und Cook, W.J. [2003]: Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems. *Mathematical Programming B* 97 (2003), 91–153
- Applegate, D., Cook, W.J., und Rohe, A. [2003]: Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing* 15 (2003), 82–92
- Arora, S. [1998]: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM* 45 (1998), 753–782
- Bellman, R. [1962]: Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM* 9 (1962), 61–63
- de Berg, M., Buchin, K., Jansen, B.M.P., und Woeginger, G. [2016]: Fine-grained complexity analysis of two classic TSP variants. *Proceedings of ICALP 2016*, Article 5
- Berman, P., und Karpinski, M. [2006]: 8/7-approximation algorithm for (1,2)-TSP. *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms* (2006), 641–648
- Boyd, S.C., und Cunningham, W.H. [1991]: Small traveling salesman polytopes. *Mathematics of Operations Research* 16 (1991), 259–271
- Burkard, R.E., Deineko, V.G., und Woeginger, G.J. [1998]: The travelling salesman and the PQ-tree. *Mathematics of Operations Research* 23 (1998), 613–623
- Carr, R. [1997]: Separating clique trees and bipartition inequalities having a fixed number of handles and teeth in polynomial time. *Mathematics of Operations Research* 22 (1997), 257–265

- Chalasani, P., Motwani, R., und Rao, A. [1996]: Algorithms for robot grasp and delivery. Proceedings of the 2nd International Workshop on Algorithmic Foundations of Robotics (1996), 347–362
- Chandra, B., Karloff, H., und Tovey, C. [1999]: New results on the old k -opt algorithm for the traveling salesman problem. SIAM Journal on Computing 28 (1999), 1998–2029
- Christofides, N. [1976]: Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh 1976
- Chvátal, V. [1973]: Edmonds' polytopes and weakly hamiltonian graphs. Mathematical Programming 5 (1973), 29–40
- Crowder, H., und Padberg, M.W. [1980]: Solving large-scale symmetric travelling salesman problems to optimality. Management Science 26 (1980), 495–509
- Dantzig, G., Fulkerson, R., und Johnson, S. [1954]: Solution of a large-scale traveling-salesman problem. Operations Research 2 (1954), 393–410
- Englert, M., Röglin, H., und Vöcking, B. [2014]: Worst case and probabilistic analysis of the 2-opt algorithm for the TSP. Algorithmica 68 (2014), 190–264
- Fiorini, S., Massar, S., Pokutta, S., Tiwary, H.R., und de Wolf, R. [2015]: Exponential lower bounds for polytopes in combinatorial optimization. Journal of the ACM 62 (2015), Article 17
- Frank, A., Triesch, E., Korte, B., und Vygen, J. [1998]: On the bipartite travelling salesman problem. Report No. 98866, Research Institute for Discrete Mathematics, University of Bonn, 1998
- Frieze, A.M., Galbiati, G., und Maffioli, F. [1982]: On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. Networks 12 (1982), 23–39
- Garey, M.R., Graham, R.L., und Johnson, D.S. [1976]: Some NP-complete geometric problems. Proceedings of the 8th Annual ACM Symposium on the Theory of Computing (1976), 10–22
- Grötschel, M., und Padberg, M.W. [1979]: On the symmetric travelling salesman problem. Mathematical Programming 16 (1979), 265–302
- Grötschel, M., und Pulleyblank, W.R. [1986]: Clique tree inequalities and the symmetric travelling salesman problem. Mathematics of Operations Research 11 (1986), 537–569
- Held, M., und Karp, R.M. [1962]: A dynamic programming approach to sequencing problems. Journal of SIAM 10 (1962), 196–210
- Held, M., und Karp, R.M. [1970]: The traveling-salesman problem and minimum spanning trees. Operations Research 18 (1970), 1138–1162
- Held, M., und Karp, R.M. [1971]: The traveling-salesman problem and minimum spanning trees; part II. Mathematical Programming 1 (1971), 6–25
- Helsgaun, K. [2009]: General k -opt submoves for the Lin-Kernighan TSP heuristic. Mathematical Programming Computation 1 (2009), 119–163
- Hoogeveen, J.A. [1991]: Analysis of Christofides' heuristic: some paths are more difficult than cycles. Operations Research Letters 10 (1991), 291–295
- Hougardy, S. [2014]: On the integrality ratio of the subtour LP for Euclidean TSP. Operations Research Letters 42 (2014), 495–499
- Hurkens, C.A.J., und Woeginger, G.J. [2004]: On the nearest neighbour rule for the traveling salesman problem. Operations Research Letters 32 (2004), 1–4
- Hwang, R.Z., Chang, R.C., und Lee, R.C.T. [1993]: The searching over separators strategy to solve some NP -hard problems in subexponential time. Algorithmica 9 (1993), 398–423
- Johnson, D.S., McGeoch, L.A., und Rothberg, E.E. [1996]: Asymptotic experimental analysis for the Held-Karp traveling salesman bound. Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (1996), 341–350

- Johnson, D.S., Papadimitriou, C.H., und Yannakakis, M. [1988]: How easy is local search? *Journal of Computer and System Sciences* 37 (1988), 79–100
- Jünger, M., und Naddef, D. [2001]: Computational Combinatorial Optimization. Springer, Berlin 2001
- Kaibel, V., und Weltge, S. [2015]: Lower bounds on the sizes of integer programs without additional variables. *Mathematical Programming B* 154 (2015), 407–425
- Karp, R.M. [1977]: Probabilistic analysis of partitioning algorithms for the TSP in the plane. *Mathematics of Operations Research* 2 (1977), 209–224
- Karp, R.M., und Papadimitriou, C.H. [1982]: On linear characterizations of combinatorial optimization problems. *SIAM Journal on Computing* 11 (1982), 620–632
- Karpinski, M., Lampis, M., und Schmied, R. [2013]: New inapproximability bounds for TSP. *Algorithms and Computation; Proceedings of ISAAC 2013; LNCS* 8283 (L. Cai, S.-W. Chen, T.-W. Lam, Hrsg.), Springer, Berlin 2013, pp. 568–578
- Klein, P.N. [2008]: A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights. *SIAM Journal on Computing* 37 (2008), 1926–1952
- Krentel, M.W. [1989]: Structure in locally optimal solutions. *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science* (1989), 216–221
- Land, A.H., und Doig, A.G. [1960]: An automatic method of solving discrete programming problems. *Econometrica* 28 (1960), 497–520
- Lee, J.R., Raghavendra, P., und Steurer, D. [2015]: Lower bounds on the size of semidefinite programming relaxations. *Proceedings of the 47th Annual ACM Symposium on Theory of Computing* (2015), 567–576
- Lin, S., und Kernighan, B.W. [1973]: An effective heuristic algorithm for the traveling salesman problem. *Operations Research* 21 (1973), 498–516
- Little, J.D.C., Murty, K.G., Sweeney, D.W., und Karel, C. [1963]: An algorithm for the traveling salesman problem. *Operations Research* 11 (1963), 972–989
- Michiels, W., Aarts, E., und Korst, J. [2007]: Theoretical Aspects of Local Search. Springer, Berlin 2007
- Miller, C.E., Tucker, A.W., und Zemlin, R.A. [1960]: Integer programming formulations of traveling salesman problems. *Journal of the ACM* 7 (1960), 326–329
- Mitchell, J. [1999]: Guillotine subdivisions approximate polygonal subdivisions: a simple polynomial-time approximation scheme for geometric TSP, k -MST, and related problems. *SIAM Journal on Computing* 28 (1999), 1298–1309
- Papadimitriou, C.H. [1977]: The Euclidean traveling salesman problem is NP -complete. *Theoretical Computer Science* 4 (1977), 237–244
- Papadimitriou, C.H. [1978]: The adjacency relation on the travelling salesman polytope is NP -complete. *Mathematical Programming* 14 (1978), 312–324
- Papadimitriou, C.H. [1992]: The complexity of the Lin-Kernighan heuristic for the traveling salesman problem. *SIAM Journal on Computing* 21 (1992), 450–465
- Papadimitriou, C.H., und Steiglitz, K. [1977]: On the complexity of local search for the traveling salesman problem. *SIAM Journal on Computing* 6 (1), 1977, 76–83
- Papadimitriou, C.H., und Steiglitz, K. [1978]: Some examples of difficult traveling salesman problems. *Operations Research* 26 (1978), 434–443
- Papadimitriou, C.H., und Yannakakis, M. [1993]: The traveling salesman problem with distances one and two. *Mathematics of Operations Research* 18 (1993), 1–12
- Rao, S.B., und Smith, W.D. [1998]: Approximating geometric graphs via “spanners” and “banyans”. *Proceedings of the 30th Annual ACM Symposium on Theory of Computing* (1998), 540–550
- Rosenkrantz, D.J., Stearns, R.E., und Lewis, P.M. [1977]: An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing* 6 (1977), 563–581

- Rothvoß, T. [2017]: The matching polytope has exponential extension complexity. *Journal of the ACM* 64 (2017), Article 41
- Sahni, S., und Gonzalez, T. [1976]: P -complete approximation problems. *Journal of the ACM* 23 (1976), 555–565
- Shmoys, D.B., und Williamson, D.P. [1990]: Analyzing the Held-Karp TSP bound: a monotonicity property with application. *Information Processing Letters* 35 (1990), 281–285
- Svensson, O., Tarnawski, J., und Végh, L. [2017]: A constant-factor approximation algorithm for the asymmetric traveling salesman problem. arXiv:1708.04215
- Traub, V., und Vygen, J. [2018]: Approaching $\frac{3}{2}$ for the s - t -path TSP. *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms* (2018), 1854–1864
- Triesch, E., Nolles, W., und Vygen, J. [1994]: Die Einsatzplanung von Zementmischern und ein Traveling Salesman Problem In: *Operations Research; Reflexionen aus Theorie und Praxis* (B. Werners, R. Gabriel, Hrsg.), Springer, Berlin 1994 [auf Deutsch]
- Woeginger, G.J. [2002]: Exact algorithms for NP -hard problems. *OPTIMA* 68 (2002), 2–8
- Wolsey, L.A. [1980]: Heuristic analysis, linear programming and branch and bound. *Mathematical Programming Study* 13 (1980), 121–134



22 Standortprobleme

Etliche betriebswirtschaftliche Entscheidungen hängen mit der Wahl und/oder Standortbestimmung von Betriebseinrichtungen zusammen, um der Nachfrage effizient gerecht zu werden. Beispiele hierzu sind Fabriken, Lagerhallen, Depots, Bibliotheken, Feuerwachen, Krankenhäuser und Basisstationen für Fernmeldedienste (z. B. Fernsehen und Mobiltelefonnetze). Diese Beispiele haben eine gemeinsame Eigenschaft: Es geht um die Platzierung von gewissen Einheiten, mit dem Ziel, die Nachfrage der Kunden optimal zu bedienen. Standortprobleme, die auch in vielen anderen weniger offensichtlichen Kontexten vorkommen, sind in der Tat weit verbreitet.

Das am meisten studierte Modell der diskreten Standortprobleme ist das sogenannte UNBESCHRÄNKTE STANDORTPROBLEM, welches wir in Abschnitt 22.1 besprechen werden. Obwohl es seit 1960 eingehend untersucht worden ist (siehe z. B. Stollsteimer [1963], Balinski und Wolfe [1963], Kuehn und Hamburger [1963] und Manne [1964]), wurde erst 1997 ein Approximationsalgorithmus gefunden. Seitdem sind einige gänzlich verschiedene Ansätze gebraucht worden, um eine obere Schranke für die Approximationsgüte zu bestimmen. Diese werden wir in diesem Kapitel zeigen; ferner werden wir auch allgemeinere Probleme besprechen, wie z. B. Versionen mit Kapazitäten, das k -MEDIAN-PROBLEM und das UNIVERSELLE STANDORTPROBLEM.

22.1 Das unbeschränkte Standortproblem

Das einfachste Problem, für welches wir viele Resultate beweisen werden, ist das UNBESCHRÄNKTE STANDORTPROBLEM. Es lautet wie folgt:

UNBESCHRÄNKTES STANDORTPROBLEM

Instanz: Eine endliche Menge \mathcal{D} von Kunden, eine endliche Menge \mathcal{F} von möglichen Standorten, Fixkosten $f_i \in \mathbb{R}_+$ für die Bereitstellung des Standortes $i \in \mathcal{F}$ und Servicekosten $c_{ij} \in \mathbb{R}_+$ für jedes $i \in \mathcal{F}$ und $j \in \mathcal{D}$.

Aufgabe: Bestimme eine Teilmenge X der Standorte (die **offenen** Standorte) und eine Zuordnung $\sigma : \mathcal{D} \rightarrow X$ der Kunden zu den offenen Standorten, so dass die Summe der Bereitstellungs- und Servicekosten

$$\sum_{i \in X} f_i + \sum_{j \in \mathcal{D}} c_{\sigma(j)j}$$

minimiert wird.

In vielen Anwendungen hängen die Servicekosten von einer Metrik c auf $\mathcal{D} \cup \mathcal{F}$ ab (sie sind z. B. proportional zu der geometrischen Entfernung oder der Fahrzeit). In diesem Fall gilt

$$c_{ij} + c_{i'j} + c_{i'j'} \geq c_{ij'} \quad \text{für alle } i, i' \in \mathcal{F} \text{ und } j, j' \in \mathcal{D}. \quad (22.1)$$

Gilt umgekehrt diese Bedingung, so können wir c folgendermaßen erweitern: $c_{ii} := 0$ und $c_{ii'} := \min_{j \in \mathcal{D}} (c_{ij} + c_{i'j})$ für $i, i' \in \mathcal{F}$, $c_{jj} := 0$ und $c_{jj'} := \min_{i \in \mathcal{F}} (c_{ij} + c_{ij'})$ für $j, j' \in \mathcal{D}$ und $c_{ji} := c_{ij}$ für $j \in \mathcal{D}$ und $i \in \mathcal{F}$. Auf diese Weise erhalten wir eine (Semi-)Metrik c auf $\mathcal{D} \cup \mathcal{F}$. Somit sprechen wir von *metrischen* Servicekosten, wenn (22.1) gilt. Das obige Problem, beschränkt auf Instanzen mit metrischen Servicekosten, heißt das METRISCHE UNBESCHRÄNKTE STANDORTPROBLEM.

Proposition 22.1. *Das METRISCHE UNBESCHRÄNKTE STANDORTPROBLEM ist stark NP-schwer.*

Beweis: Wir betrachten das MINIMUM-WEIGHT-SET-COVER-PROBLEM mit Einheitsgewichten (welches nach Korollar 15.24 stark NP-schwer ist). Jede Instanz (U, \mathcal{S}) kann folgendermaßen in eine Instanz des METRISCHEN UNBESCHRÄNKTE STANDORTPROBLEMS transformiert werden: Seien $\mathcal{D} := U$, $\mathcal{F} := \mathcal{S}$ und $f_i := 1$ für $i \in \mathcal{S}$, $c_{ij} := 1$ für $j \in i \in \mathcal{S}$ und $c_{ij} := 3$ für $j \in U \setminus \{i\}$ und $i \in \mathcal{S}$. Dann hat die resultierende Instanz für $k \leq |\mathcal{S}|$ eine Lösung mit Kosten $|\mathcal{D}| + k$ genau dann, wenn (U, \mathcal{S}) eine Überdeckung der Kardinalität k hat. \square

Die im obigen Beweis vorkommende Zahl 3 kann durch jede beliebige Zahl größer als 1 aber nicht größer als 3 ersetzt werden (sonst würde (22.1) verletzt werden). In der Tat zeigt eine ähnliche Konstruktion, dass metrische Servicekosten notwendig sind, um Approximationsalgorithmen zu erhalten: Setzen wir $c_{ij} := \infty$ für $j \in U \setminus \{i\}$ und $i \in \mathcal{S}$ im obigen Beweis, so sehen wir, dass aus jedem Approximationsalgorithmus für das UNBESCHRÄNKTE STANDORTPROBLEM die Existenz eines Approximationsalgorithmus für das MINIMUM-SET-COVER-PROBLEM mit derselben Approximationsgüte folgen würde (und für das MINIMUM-SET-COVER-PROBLEM gibt es für kein $k' \geq 1$ einen k' -Approximationsalgorithmus, sofern $P \neq NP$; siehe Abschnitt 16.1). Guha und Khuller [1999] und Sviridenko [unveröffentlicht] haben die obige Konstruktion erweitert um zu zeigen, dass aus einem 1,463-Approximationsalgorithmus für das METRISCHE UNBESCHRÄNKTE STANDORTPROBLEM (selbst wenn die Servicekosten nur 1 oder 3 betragen) $P = NP$ folgen würde (siehe Vygen [2005] zu weiteren Details).

Sei nun umgekehrt eine Instanz des UNBESCHRÄNKTE STANDORTPROBLEMS gegeben. Setzen wir $U := \mathcal{D}$, $\mathcal{S} = 2^{\mathcal{D}}$ und $c(D) := \min_{i \in \mathcal{F}} (f_i + \sum_{j \in D} c_{ij})$ für $D \subseteq \mathcal{D}$, so erhalten wir eine äquivalente Instanz des MINIMUM-WEIGHT-SET-COVER-PROBLEM. Obwohl diese Instanz exponentiell groß ist, können wir den GREEDY-ALGORITHMUS FÜR SET-COVER anwenden und erhalten in polynomischer Zeit eine Lösung mit Kosten höchstens gleich $(1 + \frac{1}{2} + \dots + \frac{1}{|\mathcal{D}|})$ mal dem Optimum (siehe Satz 16.3), wie es von Hochbaum [1982] vorgeschlagen wurde:

Dazu müssen wir bei jedem Schritt ein Paar $(D, i) \in 2^{\mathcal{D}} \times \mathcal{F}$ mit minimalem $f_i + \sum_{j \in D} c_{ij}$ und i offen finden, dann alle Kunden in D diesem i zuordnen und sie

von nun an ignorieren. Obwohl es exponentiell viele Wahlen gibt, ist es leicht, eine beste zu finden, da es genügt, Paare (D_k^i, i) für $i \in \mathcal{F}$ und $k \in \{1, \dots, |\mathcal{D}|\}$ zu betrachten, wobei D_k^i die Menge der ersten k Kunden in einer linearen Ordnung mit wachsendem c_{ij} ist. Offensichtlich können andere Paare nicht effektiver sein.

Jain et al. [2003] haben gezeigt, dass die Approximationsgüte dieses Greedy-Algorithmus sogar für metrische Instanzen gleich $\Omega(\log n / \log \log n)$ ist, wobei $n = |\mathcal{D}|$. Tatsächlich war vor der Arbeit von Shmoys, Tardos und Aardal [1997] kein Approximationsalgorithmus mit konstanter Gütegarantie bekannt, sogar nicht bei metrischen Servicekosten. Seitdem hat sich die Situation dramatisch verändert. In den folgenden Abschnitten werden wir ganz unterschiedliche Approximationsalgorithmen für das METRISCHE UNBESCHRÄNKTE STANDORTPROBLEM erläutern.

Ein noch eingeschränkteres Problem entsteht, wenn man die Standorte und Kunden als Punkte der Ebene darstellt und die Servicekosten durch die geometrischen Entfernungen gegeben sind. Hier haben Arora, Raghavan und Rao [1998] gezeigt, dass das Problem ein Approximationsschema hat, d. h. einen k -Approximationsalgorithmus für jedes $k > 1$, wie bei dem Algorithmus in Abschnitt 21.2. Dieses Ergebnis wurde von Kollaopoulos und Rao [2007] verbessert, aber der Algorithmus scheint immer noch zu langsam für praktische Zwecke zu sein.

Von nun an setzen wir allgemeine metrische Servicekosten voraus. Für eine gegebene Instanz $\mathcal{D}, \mathcal{F}, f_i, c_{ij}$ und eine gegebene nichtleere Teilmenge X von Standorten kann eine beste Zuordnung $\sigma : \mathcal{D} \rightarrow X$ mit $c_{\sigma(j)j} = \min_{i \in X} c_{ij}$ leicht berechnet werden. Demnach werden wir oft eine nichtleere Menge $X \subseteq \mathcal{F}$ eine zulässige Lösung nennen. Diese hat die Bereitstellungskosten $c_F(X) := \sum_{i \in X} f_i$ und die Servicekosten $c_S(X) := \sum_{j \in \mathcal{D}} \min_{i \in X} c_{ij}$. Ziel ist es, eine nichtleere Teilmenge $X \subseteq \mathcal{F}$ mit $c_F(X) + c_S(X)$ minimal zu finden.

22.2 Rundung von LP-Lösungen

Das UNBESCHRÄNKTE STANDORTPROBLEM kann folgendermaßen als ganzzahliges LP formuliert werden:

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij} \\ \text{bzgl.} \quad & x_{ij} \leq y_i \quad (i \in \mathcal{F}, j \in \mathcal{D}) \\ & \sum_{i \in \mathcal{F}} x_{ij} = 1 \quad (j \in \mathcal{D}) \\ & x_{ij} \in \{0, 1\} \quad (i \in \mathcal{F}, j \in \mathcal{D}) \\ & y_i \in \{0, 1\} \quad (i \in \mathcal{F}). \end{aligned}$$

Relaxieren wir die Ganzzahligkeitsnebenbedingungen, so erhalten wir das LP:

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij} \\
\text{bzgl.} \quad & x_{ij} \leq y_i \quad (i \in \mathcal{F}, j \in \mathcal{D}) \\
& \sum_{i \in \mathcal{F}} x_{ij} = 1 \quad (j \in \mathcal{D}) \\
& x_{ij} \geq 0 \quad (i \in \mathcal{F}, j \in \mathcal{D}) \\
& y_i \geq 0 \quad (i \in \mathcal{F}),
\end{aligned} \tag{22.2}$$

welches zuerst von Balinski [1965] formuliert wurde. Das duale LP lautet:

$$\begin{aligned}
\max \quad & \sum_{j \in \mathcal{D}} v_j \\
\text{bzgl.} \quad & v_j - w_{ij} \leq c_{ij} \quad (i \in \mathcal{F}, j \in \mathcal{D}) \\
& \sum_{j \in \mathcal{D}} w_{ij} \leq f_i \quad (i \in \mathcal{F}) \\
& w_{ij} \geq 0 \quad (i \in \mathcal{F}, j \in \mathcal{D}).
\end{aligned} \tag{22.3}$$

LP-Rundungsalgorithmen lösen diese LPs (siehe Satz 4.18) und runden die gebrochene Lösung des primalen LP auf geeignete Weise. Shmoys, Tardos und Aardal [1997] haben mit dieser Methode den ersten Approximationsalgorithmus mit konstanter Gütegarantie gefunden:

SHMOYS-TARDOS-AARDAL-ALGORITHMUS

Input: Eine Instanz $(\mathcal{D}, \mathcal{F}, (f_i)_{i \in \mathcal{F}}, (c_{ij})_{i \in \mathcal{F}, j \in \mathcal{D}})$ des UNBESCHRÄNKten STANDORTPROBLEMS.
Output: Eine Lösung $X \subseteq \mathcal{F}$ und $\sigma : \mathcal{D} \rightarrow X$.

- ① Berechne eine optimale Lösung (x^*, y^*) für (22.2) und eine optimale Lösung (v^*, w^*) für (22.3).
- ② Sei $k := 1$, $X := \emptyset$ und $U := \mathcal{D}$.
- ③ Sei $j_k \in U$ mit $v_{j_k}^*$ minimal.
Sei $i_k \in \mathcal{F}$ mit $x_{i_k j_k}^* > 0$ und f_{i_k} minimal. Setze $X := X \cup \{i_k\}$.
Sei $N_k := \{j \in U : \text{es gibt ein } i \in \mathcal{F} \text{ mit } x_{i j_k}^* > 0, x_{i j}^* > 0\}$.
Setze $\sigma(j) := i_k$ für alle $j \in N_k$.
Setze $U := U \setminus N_k$.
- ④ Setze $k := k + 1$.
If $U \neq \emptyset$ **then go to** ③.

Satz 22.2. (Shmoys, Tardos und Aardal [1997]) *Der obige Algorithmus ist ein 4-Approximationsalgorithmus für das METRISCHE UNBESCHRÄNKTE STANDORTPROBLEM.*

Beweis: Aus $x_{ij}^* > 0$ folgt mit den Bedingungen des komplementären Schlupfes (Korollar 3.23), dass $v_j^* - w_{ij}^* = c_{ij}$, und somit gilt $c_{ij} \leq v_j^*$. Demnach sind die Servicekosten für Kunde $j \in N_k$ höchstens gleich

$$c_{ikj} \leq c_{ij} + c_{ijk} + c_{ikjk} \leq v_j^* + 2v_{jk}^* \leq 3v_j^*,$$

wobei i ein Standort mit $x_{ij}^* > 0$ und $x_{ijk}^* > 0$ ist.

Für die Bereitstellungskosten f_{ik} gibt es die folgende Schranke:

$$f_{ik} \leq \sum_{i \in \mathcal{F}} x_{ijk}^* f_i = \sum_{i \in \mathcal{F}: x_{ijk}^* > 0} x_{ijk}^* f_i \leq \sum_{i \in \mathcal{F}: x_{ijk}^* > 0} y_i^* f_i.$$

Aus $x_{ijk}^* > 0$ folgt $x_{ijk'}^* = 0$ für $k \neq k'$, also sind die gesamten Bereitstellungskosten höchstens gleich $\sum_{i \in \mathcal{F}} y_i^* f_i$.

Addition ergibt die Gesamtkosten $3 \sum_{j \in \mathcal{D}} v_j^* + \sum_{i \in \mathcal{F}} y_i^* f_i$, und diese sind höchstens viermal der LP-Wert und somit höchstens viermal das Optimum. \square

Diese Gütegarantie wurde von Chudak und Shmoys [2003] auf 1,736 verbessert und auf 1,582 von Sviridenko [2002].

22.3 Primal-duale Algorithmen

Jain und Vazirani [2001] haben einen anderen Approximationsalgorithmus vorgeschlagen, nämlich einen klassischen primal-dualen Algorithmus: Er berechnet gleichzeitig zulässige primale und duale Lösungen (für die in Abschnitt 22.2 angegebenen LPs). Die primale ist ganzzahlig und die Approximationsgüte folgt aus approximativen Bedingungen des komplementären Schlupfes.

Man kann diesen Algorithmus so sehen, dass er stetig alle dualen Variablen (beginnend mit Null) erhöht und v_j festsetzt, falls $j \in \mathcal{D}$ vorläufig verbunden wurde. Sei $w_{ij} := \max\{0, v_j - c_{ij}\}$ zu irgendeinem Zeitpunkt. Anfangs ist kein Standort offen. Wir öffnen Standorte vorläufig und verbinden Kunden, falls folgende Ereignisse stattfinden:

- Es gilt $v_j = c_{ij}$ für einen vorläufig offenen Standort i und nicht verbundenen Kunden j .
Dann setze $\sigma(j) := i$ und setze v_j fest.
- Es gilt $\sum_{j \in \mathcal{D}} w_{ij} = f_i$ für einen Standort i , der (noch) nicht vorläufig offen ist.
Dann öffne i vorläufig. Für alle nicht verbundenen Kunden $j \in \mathcal{D}$ mit $v_j \geq c_{ij}$ setze $\sigma(j) := i$ und setze v_j fest.

Mehrere solche Ereignisse können gleichzeitig eintreten. Diese werden dann in beliebiger Reihenfolge bearbeitet. Auf diese Weise fahren wir fort, bis alle Kunden verbunden sind.

Sei nun V die Menge der vorläufig offenen Standorte und E die Menge der Paare $\{i, i'\}$ verschiedener vorläufig offener Standorte, so dass es einen Kunden j mit $w_{ij} > 0$ und $w_{i'j} > 0$ gibt. Wähle eine inklusionsmaximale stabile Menge X in dem Graphen (V, E) , öffne die Standorte in X , und verbinde jeden Kunden mit einem, ihm am nächsten, offenen Standort.

Man kann X aber auch während der vorläufigen Öffnung von Standorten auf Greedy-Weise wählen. Damit kann der Algorithmus folgendermaßen formal beschrieben werden. Es sei Y die Menge der (noch) nicht vorläufig offenen Standorte.

JAIN-VAZIRANI-ALGORITHMUS

Input: Eine Instanz $(\mathcal{D}, \mathcal{F}, (f_i)_{i \in \mathcal{F}}, (c_{ij})_{i \in \mathcal{F}, j \in \mathcal{D}})$ des UNBESCHRÄNKTE STANDORTPROBLEMS.

Output: Eine Lösung $X \subseteq \mathcal{F}$ und $\sigma : \mathcal{D} \rightarrow X$.

- ① Setze $X := \emptyset$, $Y := \mathcal{F}$ und $U := \mathcal{D}$.
- ② Setze $t_1 := \min\{c_{ij} : i \in \mathcal{F} \setminus Y, j \in U\}$.
Setze $t_2 := \min\{\tau : \exists i \in Y : \omega(i, \tau, U) = f_i\}$, wobei
 $\omega(i, \tau, U) := \sum_{j \in U} \max\{0, \tau - c_{ij}\} + \sum_{j \in \mathcal{D} \setminus U} \max\{0, v_j - c_{ij}\}$.
Setze $t := \min\{t_1, t_2\}$.
- ③ **For** $i \in \mathcal{F} \setminus Y$ und $j \in U$ mit $c_{ij} = t$ **do**:
Setze $v_j := t$ und $U := U \setminus \{j\}$.
- ④ **For** $i \in Y$ mit $\omega(i, t, U) = f_i$ **do**:
Setze $Y := Y \setminus \{i\}$.
If es gibt keine $i' \in X$ und $j \in \mathcal{D} \setminus U$ mit $v_j > c_{ij}$ und $v_j > c_{i'j}$
then setze $X := X \cup \{i\}$.
For $j \in U$ mit $c_{ij} \leq t$ **do**: Setze $v_j := t$ und $U := U \setminus \{j\}$.
- ⑤ **If** $U \neq \emptyset$ **then go to** ②.
- ⑥ **For** $j \in \mathcal{D}$ **do**: Sei $i \in X$ mit $c_{ij} = \min\{c_{i'j} : i' \in X\}$. Setze $\sigma(j) := i$.

Satz 22.3. (Jain und Vazirani [2001]) Für metrische Instanzen I öffnet der JAIN-VAZIRANI-ALGORITHMUS eine Menge X von Standorten mit $3c_F(X) + c_S(X) \leq 3 \text{OPT}(I)$. Insbesondere ist er ein 3-Approximationsalgorithmus für das METRISCHE UNBESCHRÄNKTE STANDORTPROBLEM. Er kann mit Laufzeit $O(m \log m)$ implementiert werden, wobei $m = |\mathcal{F}| |\mathcal{D}|$.

Beweis: Beachte zunächst, dass t während des Verlaufs des Algorithmus nicht abnimmt.

Der Algorithmus berechnet eine primale Lösung X , ferner σ und die Zahlen v_j , $j \in \mathcal{D}$, welche zusammen mit $w_{ij} := \max\{0, v_j - c_{ij}\}$, $i \in \mathcal{F}$, $j \in \mathcal{D}$, eine zulässige Lösung für das duale LP (22.3) bilden. Somit gilt $\sum_{j \in \mathcal{D}} v_j \leq \text{OPT}(I)$. Für jeden offenen Standort i sind alle Kunden j mit $w_{ij} > 0$ mit i verbunden und

es gilt $f_i = \sum_{j \in \mathcal{D}} w_{ij}$. Ferner behaupten wir, dass die Servicekosten für jeden Kunden j höchstens gleich $3(v_j - w_{\sigma(j)j})$ sind.

Wir unterscheiden zwei Fälle. Ist $c_{\sigma(j)j} = v_j - w_{\sigma(j)j}$, so folgt die Behauptung. Sei anderenfalls $c_{\sigma(j)j} > v_j$ und $w_{\sigma(j)j} = 0$. Dann folgt $c_{ij} > v_j$ für alle $i \in X$. Dies bedeutet: Wenn j in ③ oder ④ aus U entfernt wird, gibt es ein $i \in \mathcal{F} \setminus (Y \cup X)$ mit $c_{ij} \leq v_j$ und ein $j' \in \mathcal{D} \setminus U$ und ein $i' \in X$ mit $w_{ij'} > 0$ und $w_{i'j'} > 0$. Somit folgt $c_{ij'} = v_{j'} - w_{ij'} < v_{j'}$ und $c_{i'j'} = v_{j'} - w_{i'j'} < v_{j'}$. Beachte, dass $v_{j'} \leq v_j$, da j' vor j aus U entfernt wird. Damit folgt nun $c_{\sigma(j)j} \leq c_{ij} \leq c_{i'j'} + c_{ij'} + c_{ij} < v_{j'} + v_{j'} + v_j \leq 3v_j = 3(v_j - w_{\sigma(j)j})$.

Zur Laufzeit beachten wir, dass die Anzahl der Iterationen höchstens gleich $|\mathcal{D}| + 1$ ist, da bei jeder Iteration mindestens ein Kunde aus U entfernt wird, außer eventuell der ersten, falls $f_i = 0$ für ein $i \in \mathcal{F}$. Die Gesamtzeit für die Berechnung von t_1 in ② und für die Ausführung von ③ ist $O(m \log m)$, falls wir alle c_{ij} ein für alle Mal vorweg sortieren. Beachte als Nächstes, dass $t_2 = \min \left\{ \frac{t_2^i}{|U_i|} : i \in Y \right\}$, wobei

$$t_2^i = f_i + \sum_{j \in \mathcal{D} \setminus U : v_j > c_{ij}} (c_{ij} - v_j) + \sum_{j \in U_i} c_{ij}$$

und U_i die Menge der nicht verbundenen Kunden ist, deren Servicekosten bezüglich i höchstens gleich dem neuen Wert von t sind. Da wir gerade diese Zahl berechnen wollen, gehen wir folgendermaßen vor.

Wir speichern t_2 , t_2^i und $|U_i|$ ($i \in Y$) während des gesamten Verlaufs; am Anfang ist $t_2 = \infty$, $t_2^i = f_i$ und $|U_i| = 0$ für alle i . Wird ein neuer Kunde j verbunden und ist $v_j > c_{ij}$ für ein $i \in Y$, so wird t_2^i um v_j und $|U_i|$ um 1 verringert, wodurch eventuell auch eine Änderung in t_2 verursacht wird. Wir müssen aber auch $|U_i|$ um 1 und t_2^i um c_{ij} erhöhen (und eventuell auch t_2 ändern), falls t den Wert c_{ij} für ein $i \in Y$ und ein $j \in U$ erreicht. Dies kann dadurch bewerkstelligt werden, dass wir t_1 in ② durch $t_1 := \min\{c_{ij} : i \in \mathcal{F}, j \in U\}$ neu definieren und diese Aktualisierungen vor ⑤ für alle $i \in Y$ und $j \in U$ mit $c_{ij} = t$ erledigen. Beachte, dass es insgesamt $O(m)$ solche Aktualisierungen gibt, von denen jede konstante Zeit benötigt.

Die **If**-Aussage in ④ kann in $O(|\mathcal{D}|)$ -Zeit implementiert werden, wenn wir für jedes $j \in \mathcal{D} \setminus U$ den Standort $i' \in X$ mit $v_{j'} > c_{i'j'}$ (falls vorhanden; es gibt höchstens einen) speichern. \square

Ein besserer primal-dualer Algorithmus ist von Jain et al. [2003] vorgeschlagen worden. Eine Idee besteht darin, die Zulässigkeit der dualen Variablen zu relaxieren. Dann interpretieren wir die dualen Variablen als die Budgets der Kunden, die sie zur Zahlung der Servicekosten und zur Bezugssumme der Bereitstellungskosten der Standorte benötigen. Ein Standort wird geöffnet, wenn die zugesagten Zuschüsse die Bereitstellungskosten decken. Für verbundene Kunden erhöht sich das Budget nicht weiter, aber sie können weiterhin anderen Standorten gewisse Summen anbieten, falls diese näher liegen und eine alternative Verbindung Servicekosten sparen würde. Der Algorithmus läuft wie folgt ab.

DUAL-FITTING-ALGORITHMUS

Input: Eine Instanz $(\mathcal{D}, \mathcal{F}, (f_i)_{i \in \mathcal{F}}, (c_{ij})_{i \in \mathcal{F}, j \in \mathcal{D}})$ des UNBESCHRÄNKTE STANDORTPROBLEMS.

Output: Eine Lösung $X \subseteq \mathcal{F}$ und $\sigma : \mathcal{D} \rightarrow X$.

- ① Sei $X := \emptyset$ und $U := \mathcal{D}$.
- ② Setze $t_1 := \min\{c_{ij} : i \in X, j \in U\}$.
Setze $t_2 := \min\{\tau : \exists i \in \mathcal{F} \setminus X : \omega(i, \tau, U) = f_i\}$, wobei
 $\omega(i, \tau, U) := \sum_{j \in U} \max\{0, \tau - c_{ij}\} + \sum_{j \in \mathcal{D} \setminus U} \max\{0, c_{\sigma(j)j} - c_{ij}\}$.
Setze $t := \min\{t_1, t_2\}$.
- ③ **For** $i \in X$ und $j \in U$ mit $c_{ij} = t$ **do**:
 Setze $\sigma(j) := i$, $v_j := t$ und $U := U \setminus \{j\}$.
- ④ **For** $i \in \mathcal{F} \setminus X$ mit $\omega(i, t, U) = f_i$ **do**:
 Setze $X := X \cup \{i\}$.
 For $j \in \mathcal{D} \setminus U$ mit $c_{ij} < c_{\sigma(j)j}$ **do**: Setze $\sigma(j) := i$.
 For $j \in U$ mit $c_{ij} \leq t$ **do**: Setze $\sigma(j) := i$, $v_j := t$ und $U := U \setminus \{j\}$.
- ⑤ **If** $U \neq \emptyset$ **then go to** ②.

Satz 22.4. Der obige Algorithmus berechnet Zahlen v_j , $j \in \mathcal{D}$, und eine zulässige Lösung X, σ mit Kosten $\sum_{j \in \mathcal{D}} v_j$. Er kann mit Laufzeit $O(|\mathcal{F}|^2 |\mathcal{D}|)$ implementiert werden.

Beweis: Die erste Aussage ist klar. Die Laufzeit kann man wie bei dem JAIN-VAZIRANI-ALGORITHMUS beweisen. Wir müssen jedoch alle t_2^i bei jeder Verbindungsänderung eines Kunden aktualisieren, d. h. immer dann, wenn ein neuer Standort geöffnet wird. \square

Wir werden eine Zahl γ bestimmen, für die $\sum_{j \in D} v_j \leq \gamma (f_i + \sum_{j \in D} c_{ij})$ für alle Paare $(i, D) \in \mathcal{F} \times 2^{\mathcal{D}}$ gilt (d. h. es ist $(\frac{v_j}{\gamma})_{j \in \mathcal{D}}$ eine zulässige Lösung des in Aufgabe 3 gegebenen dualen LP). Daraus folgt dann die Approximationsgüte γ . Natürlich müssen wir metrische Servicekosten voraussetzen.

Betrachte $i \in \mathcal{F}$ und $D \subseteq \mathcal{D}$ mit $|D| = d$. Nummeriere die Kunden in D in der Reihenfolge ihres Entfernens aus U während des Verlaufs des Algorithmus; o. B. d. A. können wir annehmen, dass $D = \{1, \dots, d\}$. Es gilt $v_1 \leq v_2 \leq \dots \leq v_d$.

Sei $k \in D$. Beachte, dass k im Algorithmus zum Zeitpunkt $t = v_k$ verbunden wird, und betrachte die Lage zu dem Zeitpunkt, wo t in ② zum ersten Mal auf v_k gesetzt wird. Für $j = 1, \dots, k-1$ setzen wir

$$r_{j,k} := \begin{cases} c_{i(j,k)j} & \text{falls } j \text{ zu diesem Zeitpunkt mit } i(j, k) \in \mathcal{F} \text{ verbunden ist,} \\ v_k & \text{sonst, d. h. falls } v_j = v_k. \end{cases}$$

Nun geben wir gültige Ungleichungen für diese Variablen an. Zunächst haben wir für $j = 1, \dots, d-2$,

$$r_{j,j+1} \geq r_{j,j+2} \geq \dots \geq r_{j,d}, \quad (22.4)$$

da die Servicekosten bei einer Verbindungsänderung abnehmen. Als Nächstes haben wir für $k = 1, \dots, d$,

$$\sum_{j=1}^{k-1} \max\{0, r_{j,k} - c_{ij}\} + \sum_{l=k}^d \max\{0, v_k - c_{il}\} \leq f_i. \quad (22.5)$$

Um dies zu sehen, betrachten wir zwei Fälle. Ist $i \in \mathcal{F} \setminus X$ zu dem betrachteten Zeitpunkt, so gilt (22.5) nach Wahl von t in ②. Andernfalls ist i vorher zu X hinzugefügt worden und zu jenem Zeitpunkt galt $\sum_{j \in U} \max\{0, v_j - c_{ij}\} + \sum_{j \in \mathcal{D} \setminus U} \max\{0, c_{\sigma(j),j} - c_{ij}\} = f_i$. Nach jenem Zeitpunkt kann die linke Seite nur kleiner werden.

Schließlich haben wir für $1 \leq j < k \leq d$,

$$v_k \leq r_{j,k} + c_{ij} + c_{ik}; \quad (22.6)$$

für $r_{j,k} = v_k$ ist diese Ungleichung trivial, sonst folgt sie nach Wahl von t_1 in ②: Beachte, dass die rechte Seite von (22.6) wegen der metrischen Servicekosten mindestens gleich $c_{i(j,k)k}$ ist und dass Standort $i(j, k)$ zum betrachteten Zeitpunkt offen ist.

Zum Beweis der Approximationsgüte betrachten wir das folgende Optimierungsproblem für $\gamma_F \geq 1$ und $d \in \mathbb{N}$. Da wir eine Aussage für alle Instanzen machen wollen, betrachten wir f_i , c_{ij} und v_j ($j = 1, \dots, d$) und $r_{j,k}$ ($1 \leq j < k \leq d$) als Variablen:

$$\begin{aligned} \max \quad & \frac{\sum_{j=1}^d v_j - \gamma_F f_i}{\sum_{j=1}^d c_{ij}} \\ \text{bzgl.} \quad & \begin{aligned} v_j &\leq v_{j+1} & (1 \leq j < d) \\ r_{j,k} &\geq r_{j,k+1} & (1 \leq j < k < d) \\ r_{j,k} + c_{ij} + c_{ik} &\geq v_k & (1 \leq j < k \leq d) \end{aligned} \\ & \sum_{j=1}^{k-1} \max\{0, r_{j,k} - c_{ij}\} + \\ & \sum_{l=k}^d \max\{0, v_k - c_{il}\} \leq f_i & (1 \leq k \leq d) \\ & \sum_{j=1}^d c_{ij} > 0 \\ & f_i \geq 0 \\ & v_j, c_{ij} \geq 0 & (1 \leq j \leq d) \\ & r_{j,k} \geq 0 & (1 \leq j < k \leq d). \end{aligned} \quad (22.7)$$

Beachte, dass dieses Optimierungsproblem leicht als LP formuliert werden kann (Aufgabe 6). Aus seinen optimalen Zielfunktionswerten folgt die Approximationsgüte für den DUAL-FITTING-ALGORITHMUS:

Satz 22.5. *Sei $\gamma_F \geq 1$, sei ferner γ_S das Supremum der optimalen Zielfunktionswerte des LP (22.7) bezüglich aller $d \in \mathbb{N}$. Gegeben sei eine Instanz des METRISCHEN UNBESCHRÄNKTE STANDORTPROBLEMS mit einer Lösung $X^* \subseteq \mathcal{F}$.*

Dann sind die Kosten der mit dem DUAL-FITTING-ALGORITHMUS für diese Instanz berechneten Lösung höchstens gleich $\gamma_{FCF}(X^*) + \gamma_{SCS}(X^*)$.

Beweis: Der Algorithmus liefert Zahlen v_j und implizit die Zahlen $r_{j,k}$ für alle $j, k \in \mathcal{D}$ mit $v_j \leq v_k$ und $j \neq k$. Für jedes Paar $(i, D) \in \mathcal{F} \times 2^{\mathcal{D}}$ erfüllen die Zahlen f_i, c_{ij}, v_j und $r_{j,k}$ die Bedingungen (22.4), (22.5) und (22.6), und bilden somit eine zulässige Lösung für (22.7), sofern $\sum_{j=1}^d c_{ij} \neq 0$. Damit gilt $\sum_{j=1}^d v_j - \gamma_F f_i \leq \gamma_S \sum_{j=1}^d c_{ij}$ (ist $c_{ij} = 0$ für alle $j \in D$, so folgt dies sofort aus (22.5) und (22.6)). Wählen wir nun $\sigma^* : \mathcal{D} \rightarrow X^*$ mit $c_{\sigma^*(j)j} = \min_{i \in X^*} c_{ij}$ und summieren über alle Paare $(i, \{j \in \mathcal{D} : \sigma^*(j) = i\})$ ($i \in X^*$), so erhalten wir

$$\sum_{j \in \mathcal{D}} v_j \leq \gamma_F \sum_{i \in X^*} f_i + \gamma_S \sum_{j \in \mathcal{D}} c_{\sigma^*(j)j} = \gamma_{FCF}(X^*) + \gamma_{SCS}(X^*).$$

Da die von dem Algorithmus berechnete Lösung Gesamtkosten höchstens gleich $\sum_{j \in \mathcal{D}} v_j$ hat, ist der Satz bewiesen. \square

Um dies anwenden zu können, beachten wir:

Lemma 22.6. Betrachte das LP (22.7) für ein $d \in \mathbb{N}$.

- (a) Für $\gamma_F = 1$ ist der optimale Zielfunktionswert höchstens gleich 2.
- (b) Für $\gamma_F = 1,61$ ist der optimale Zielfunktionswert höchstens gleich 1,61 (Jain et al. [2003]).
- (c) Für $\gamma_F = 1,11$ ist der optimale Zielfunktionswert höchstens gleich 1,78 (Mahdian, Ye und Zhang [2006]).

Beweis: Wir beweisen hier nur (a). Für eine zulässige Lösung gilt

$$\begin{aligned} d \left(f_i + \sum_{j=1}^d c_{ij} \right) &\geq \sum_{k=1}^d \left(\sum_{j=1}^{k-1} r_{j,k} + \sum_{l=k}^d v_k \right) \\ &\geq \sum_{k=1}^d dv_k - (d-1) \sum_{j=1}^d c_{ij}. \end{aligned} \tag{22.8}$$

Daraus folgt

$$d \sum_{j=1}^d v_j \leq df_i + (2d-1) \sum_{j=1}^d c_{ij}, \quad \text{d.h. } \sum_{j=1}^d v_j \leq f_i + 2 \sum_{j=1}^d c_{ij}.$$

\square

Die Beweise von (b) und (c) sind recht lang und technisch. Aus (a) folgt sofort, dass $(\frac{v_j}{2})_{j \in \mathcal{D}}$ eine zulässige duale Lösung ist und dass der DUAL-FITTING-ALGORITHMUS ein 2-Approximationsalgorithmus ist. Aus (b) folgt eine Approximationsgüte von 1,61. Noch bessere Ergebnisse lassen sich durch eine Kombination

des DUAL-FITTING-ALGORITHMUS mit Skalierung und Greedy-Augmentierung erzielen: Methoden, die wir im nächsten Abschnitt besprechen werden.

Das folgende Korollar, welches wir später benötigen, fasst einige Folgerungen aus Satz 22.5 und Lemma 22.6 zusammen:

Korollar 22.7. *Sei $(\gamma_F, \gamma_S) \in \{(1, 2), (1, 61, 1, 61), (1, 11, 1, 78)\}$. Gegeben sei eine Instanz des METRISCHEN UNBESCHRÄNKten STANDORTPROBLEMS und eine Lösung $\emptyset \neq X^* \subseteq \mathcal{F}$. Dann sind die Kosten einer mit dem DUAL-FITTING-ALGORITHMUS für diese Instanz berechneten Lösung höchstens gleich $\gamma_F c_F(X^*) + \gamma_S c_S(X^*)$.* \square

22.4 Skalierung und Greedy-Augmentierung

Viele approximative Resultate sind asymmetrisch bezüglich der Bereitstellungs- und Servicekosten. Oft können die Servicekosten durch die Öffnung weiterer Standorte verringert werden. Dies kann in einigen Fällen tatsächlich zu einer besseren Approximationsgüte führen.

Proposition 22.8. *Sei $\emptyset \neq X, X^* \subseteq \mathcal{F}$. Dann gilt $\sum_{i \in X^*} (c_S(X) - c_S(X \cup \{i\})) \geq c_S(X) - c_S(X^*)$.*

Insbesondere gibt es ein $i \in X^$ mit $\frac{c_S(X) - c_S(X \cup \{i\})}{f_i} \geq \frac{c_S(X) - c_S(X^*)}{c_F(X^*)}$.*

Beweis: Für $j \in \mathcal{D}$ sei $\sigma(j) \in X$ mit $c_{\sigma(j)j} = \min_{i \in X} c_{ij}$. Sei ferner $\sigma^*(j) \in X^*$ mit $c_{\sigma^*(j)j} = \min_{i \in X^*} c_{ij}$. Dann gilt $c_S(X) - c_S(X \cup \{i\}) \geq \sum_{j \in \mathcal{D}: \sigma^*(j)=i} (c_{\sigma(j)j} - c_{ij})$ für alle $i \in X^*$. Summieren wir, so folgt das Lemma. \square

Die **Greedy-Augmentierung** einer Menge X bedeutet den folgenden Vorgang: Wähle schrittweise ein $\frac{c_S(X) - c_S(X \cup \{i\})}{f_i}$ maximierendes Element $i \in \mathcal{F}$ und füge es zu X hinzu, bis $c_S(X) - c_S(X \cup \{i\}) \leq f_i$ für alle $i \in \mathcal{F}$. Wir benötigen das folgende Lemma:

Lemma 22.9. (Charikar und Guha [2005]) *Sei $\emptyset \neq X, X^* \subseteq \mathcal{F}$. Wende die Greedy-Augmentierung auf X an. Sei $Y \supseteq X$ die resultierende Menge. Dann gilt*

$$\begin{aligned} c_F(Y) + c_S(Y) &\leq \\ c_F(X) + c_F(X^*) \ln \left(\max \left\{ 1, \frac{c_S(X) - c_S(X^*)}{c_F(X^*)} \right\} \right) &+ c_F(X^*) + c_S(X^*). \end{aligned}$$

Beweis: Gilt $c_S(X) \leq c_F(X^*) + c_S(X^*)$, so gilt die obige Ungleichung offensichtlich, selbst wenn X an der Stelle von Y steht. Die Kosten werden durch Greedy-Augmentierung niemals erhöht.

Andernfalls sei $X = X_0, X_1, \dots, X_k$ die Folge der augmentierten Mengen, wobei k der erste Index mit $c_S(X_k) \leq c_F(X^*) + c_S(X^*)$ ist. Mittels Umnummerierung der Standorte können wir erreichen, dass $X_i \setminus X_{i-1} = \{i\}$ ($i = 1, \dots, k$). Nach Proposition 22.8 gilt für $i = 1, \dots, k$:

$$\frac{c_S(X_{i-1}) - c_S(X_i)}{f_i} \geq \frac{c_S(X_{i-1}) - c_S(X^*)}{c_F(X^*)}.$$

Somit gilt $f_i \leq c_F(X^*) \frac{c_S(X_{i-1}) - c_S(X_i)}{c_S(X_{i-1}) - c_S(X^*)}$ (beachte, dass $c_S(X_{i-1}) > c_S(X^*)$) und

$$c_F(X_k) + c_S(X_k) \leq c_F(X) + c_F(X^*) \sum_{i=1}^k \frac{c_S(X_{i-1}) - c_S(X_i)}{c_S(X_{i-1}) - c_S(X^*)} + c_S(X_k).$$

Da die rechte Seite mit wachsendem $c_S(X_k)$ zunimmt (Die Ableitung ist $1 - \frac{c_F(X^*)}{c_S(X_{k-1}) - c_S(X^*)} > 0$), wird die rechte Seite nicht kleiner, wenn wir $c_S(X_k)$ durch $c_F(X^*) + c_S(X^*)$ ersetzen. Unter Benutzung der Ungleichung $x - 1 \geq \ln x$ für $x > 0$ bekommen wir

$$\begin{aligned} c_F(X_k) + c_S(X_k) &\leq c_F(X) + c_F(X^*) \sum_{i=1}^k \left(1 - \frac{c_S(X_i) - c_S(X^*)}{c_S(X_{i-1}) - c_S(X^*)} \right) + c_S(X_k) \\ &\leq c_F(X) - c_F(X^*) \sum_{i=1}^k \ln \frac{c_S(X_i) - c_S(X^*)}{c_S(X_{i-1}) - c_S(X^*)} + c_S(X_k) \\ &= c_F(X) - c_F(X^*) \ln \frac{c_S(X_k) - c_S(X^*)}{c_S(X) - c_S(X^*)} + c_S(X_k) \\ &= c_F(X) + c_F(X^*) \ln \frac{c_S(X) - c_S(X^*)}{c_F(X^*)} + c_F(X^*) + c_S(X^*). \end{aligned}$$

□

Damit können wir einige der früheren Ergebnisse für die Approximationsgüte verbessern. Gelegentlich ist es von Vorteil, Greedy-Augmentierung mit Skalierung zu kombinieren. Dazu haben wir das folgende allgemeine Resultat:

Satz 22.10. *Es seien $\beta, \gamma_S, \gamma_F$ positive Konstanten und A ein Algorithmus, der für jede Instanz eine Lösung X berechnet, so dass $\beta c_F(X) + c_S(X) \leq \gamma_F c_F(X^*) + \gamma_S c_S(X^*)$ für jedes $\emptyset \neq X^* \subseteq \mathcal{F}$. Ferner sei $\delta \geq \frac{1}{\beta}$.*

Skaliert man dann die Bereitstellungskosten um δ , wendet A auf die modifizierte Instanz an und anschließend die Greedy-Augmentierung auf das Ergebnis bezüglich der ursprünglichen Instanz, so erhält man eine Lösung mit Kosten höchstens gleich $\max\{\frac{\gamma_F}{\beta} + \ln(\beta\delta), 1 + \frac{\gamma_S - 1}{\beta\delta}\}$ mal dem Optimum.

Beweis: Sei X^* die Menge der offenen Standorte einer optimalen Lösung für die ursprüngliche Instanz. Dann gilt $\beta c_F(X) + c_S(X) \leq \gamma_F c_F(X^*) + \gamma_S c_S(X^*)$.

Ist $c_S(X) \leq c_S(X^*) + c_F(X^*)$, so folgt $\beta\delta(c_F(X) + c_S(X)) \leq \gamma_F\delta c_F(X^*) + \gamma_S c_S(X^*) + (\beta\delta - 1)(c_S(X^*) + c_F(X^*))$, also ist X eine Lösung mit Kosten höchstens gleich $\max\{1 + \frac{\gamma_F\delta - 1}{\beta\delta}, 1 + \frac{\gamma_S - 1}{\beta\delta}\}$ mal dem Optimum. Beachte, dass $1 + \frac{\gamma_F\delta - 1}{\beta\delta} \leq \frac{\gamma_F}{\beta} + \ln(\beta\delta)$, da $1 - \frac{1}{x} \leq \ln x$ für alle $x > 0$.

Anderenfalls wenden wir Greedy-Augmentierung auf X an und erhalten eine Lösung mit Kosten höchstens gleich

$$\begin{aligned} & c_F(X) + c_F(X^*) \ln \frac{c_S(X) - c_S(X^*)}{c_F(X^*)} + c_F(X^*) + c_S(X^*) \\ & \leq c_F(X) + c_F(X^*) \ln \frac{(\gamma_S - 1)c_S(X^*) + \gamma_F \delta c_F(X^*) - \beta \delta c_F(X)}{c_F(X^*)} \\ & \quad + c_F(X^*) + c_S(X^*). \end{aligned}$$

Die Ableitung der rechten Seite bezüglich $c_F(X)$ ist

$$1 - \frac{\beta \delta c_F(X^*)}{(\gamma_S - 1)c_S(X^*) + \gamma_F \delta c_F(X^*) - \beta \delta c_F(X)},$$

und diese verschwindet, falls $c_F(X) = \frac{\gamma_F - \beta}{\beta} c_F(X^*) + \frac{\gamma_S - 1}{\beta \delta} c_S(X^*)$. Demnach erhalten wir eine Lösung mit Kosten höchstens gleich

$$\left(\frac{\gamma_F}{\beta} + \ln(\beta \delta) \right) c_F(X^*) + \left(1 + \frac{\gamma_S - 1}{\beta \delta} \right) c_S(X^*).$$

□

Mit Korollar 22.7 können wir dieses Ergebnis auf den DUAL-FITTING-ALGORITHMUS mit $\beta = \gamma_F = 1$ und $\gamma_S = 2$ anwenden: Nehmen wir $\delta = 1,76$, so erhalten wir die Approximationsgüte 1,57. Mit $\beta = 1$, $\gamma_F = 1,11$ und $\gamma_S = 1,78$ (siehe Korollar 22.7) geht es sogar noch besser:

Korollar 22.11. (Mahdian, Ye und Zhang [2006]) *Der folgende Algorithmus: Multipliziere alle Bereitstellungskosten mit $\delta = 1,504$, wende den DUAL-FITTING-ALGORITHMUS an, skaliere alle Bereitstellungskosten zurück und wende Greedy-Augmentierung an, hat die Approximationsgüte 1,52.* □

Byrka und Aardal [2007] haben gezeigt, dass die Approximationsgüte dieses Algorithmus nicht besser als 1,494 sein kann. Sie haben auch einen Algorithmus mit der Approximationsgüte 1,500 gefunden, indem sie das obigen Verfahren mit einem LP-Rundungsalgorithmus kombinierten. (Byrka und Aardal [2010]). Diese wurde wiederum von Li [2013] auf 1,488 verbessert, die momentan beste bekannte Approximationsgüte.

Für den Spezialfall, dass sämtliche Servicekosten zwischen 1 und 3 liegen, liefert Greedy-Augmentierung eine noch bessere Approximationsgüte. Sei α die Lösung der Gleichung $\alpha + 1 = \ln \frac{2}{\alpha}$; es ist $0,463 \leq \alpha \leq 0,4631$. Man überprüft leicht, dass $\alpha = \frac{\alpha}{\alpha+1} \ln \frac{2}{\alpha} = \max \left\{ \frac{\xi}{\xi+1} \ln \frac{2}{\xi} : \xi > 0 \right\}$.

Satz 22.12. (Guha und Khuller [1999]) *Betrachte das UNBESCHRÄNKTE STANDORTPROBLEM beschränkt auf Instanzen, für die sämtliche Servicekosten in dem Intervall $[1, 3]$ liegen. Für dieses Problem gibt es einen $(1 + \alpha + \epsilon)$ -Approximationsalgorithmus für jedes $\epsilon > 0$.*

Beweis: Sei $\epsilon > 0$ und setze $k := \lceil \frac{1}{\epsilon} \rceil$. Enumeriere alle Lösungen $X \subseteq \mathcal{F}$ mit $|X| \leq k$.

Nun berechnen wir eine andere Lösung wie folgt. Zunächst öffnen wir einen Standort i mit minimalen Bereitstellungskosten f_i , dann wenden wir Greedy-Augmentierung an und erhalten eine Lösung Y . Wir behaupten, dass die beste Lösung Kosten höchstens gleich $1 + \alpha + \epsilon$ mal dem Optimum hat.

Sei X^* eine optimale Lösung und $\xi = \frac{c_F(X^*)}{c_S(X^*)}$. Wir können $|X^*| > k$ annehmen, da wir sonst X^* bereits oben gefunden haben. Es folgt $c_F(\{i\}) \leq \frac{1}{k} c_F(X^*)$. Ferner gilt $c_S(\{i\}) \leq 3|\mathcal{D}| \leq 3c_S(X^*)$, da die Servicekosten zwischen 1 und 3 liegen.

Nach Lemma 22.9 sind die Kosten von Y höchstens gleich

$$\begin{aligned} & \frac{1}{k} c_F(X^*) + c_F(X^*) \ln \left(\max \left\{ 1, \frac{2c_S(X^*)}{c_F(X^*)} \right\} \right) + c_F(X^*) + c_S(X^*) \\ &= c_S(X^*) \left(\frac{\xi}{k} + \xi \ln \left(\max \left\{ 1, \frac{2}{\xi} \right\} \right) + \xi + 1 \right) \\ &\leq c_S(X^*) (1 + \xi) \left(1 + \epsilon + \frac{\xi}{\xi + 1} \ln \left(\max \left\{ 1, \frac{2}{\xi} \right\} \right) \right) \\ &\leq (1 + \alpha + \epsilon)(1 + \xi)c_S(X^*) \\ &= (1 + \alpha + \epsilon)(c_F(X^*) + c_S(X^*)). \end{aligned}$$

□

In Anbetracht des folgenden Satzes scheint diese Approximationsgüte die bestmögliche zu sein:

Satz 22.13. *Gibt es ein $\epsilon > 0$ und einen $(1 + \alpha - \epsilon)$ -Approximationsalgorithmus für das UNBESCHRÄNKTE STANDORTPROBLEM beschränkt auf Instanzen mit sämtlichen Servicekosten gleich 1 oder 3, so gilt $P = NP$.*

Dieser Satz ist von Sviridenko [unveröffentlicht] bewiesen worden (basierend auf Resultaten von Feige [1998] und Guha und Khuller [1999]), und ist in der Überblicksarbeit von Vygen [2005] enthalten.

22.5 Beschränkung der Standortanzahl

Das k -STANDORTPROBLEM ist das UNBESCHRÄNKTE STANDORTPROBLEM mit der zusätzlichen Bedingung, dass nicht mehr als k Standorte geöffnet werden dürfen, wobei k eine zur Instanz gehörende natürliche Zahl ist. Der Spezialfall, wo sämtliche Bereitstellungskosten verschwinden, ist das bekannte k -MEDIAN-PROBLEM. In diesem Abschnitt werden wir einen Approximationsalgorithmus für das METRISCHE k -STANDORTPROBLEM beschreiben.

Für Probleme, die durch das Weglassen von Nebenbedingungen eines gewissen Typs viel einfacher werden, wird oft die Lagrange-Relaxierung (siehe Abschnitt 5.6) verwendet. Hier relaxieren wir die Schranke für die Anzahl der offenen Standorte und fügen allen Bereitstellungskosten eine Konstante λ hinzu.

Satz 22.14. (Jain und Vazirani [2001]) *Gibt es eine Konstante γ_S und einen polynomiellen Algorithmus A , so dass A für jede Instanz des METRISCHEN UNBESCHRÄNKTN STANDORTPROBLEMS eine Lösung X mit $c_F(X) + c_S(X) \leq c_F(X^*) + \gamma_S c_S(X^*)$ für jedes $\emptyset \neq X^* \subseteq \mathcal{F}$ berechnet, dann gibt es einen $(2\gamma_S)$ -Approximationsalgorithmus für das METRISCHE k -STANDORTPROBLEM mit ganzzähligen Daten.*

Beweis: Gegeben sei eine Instanz des METRISCHEN k -STANDORTPROBLEMS. Wir nehmen an, dass sämtliche Servicekosten ganze Zahlen aus $\{0, 1, \dots, c_{\max}\}$ und sämtliche Bereitstellungskosten ganze Zahlen aus $\{0, 1, \dots, f_{\max}\}$ sind.

Zunächst prüfen wir, ob es eine Lösung mit verschwindenden Kosten gibt und, falls ja, bestimmen wir eine. Dies ist leicht; siehe den Beweis von Lemma 22.15. Somit nehmen wir nun an, dass die Kosten einer jeden Lösung mindestens gleich 1 sind. Sei X^* eine optimale Lösung (wir werden sie nur zur Analyse gebrauchen).

Sei $A(\lambda) \subseteq \mathcal{F}$ die von A berechnete Lösung für die Instanz, bei der sämtliche Bereitstellungskosten um λ erhöht worden sind und die Bedingung bezüglich der Standortanzahl weggelassen wurde. Dann gilt $c_F(A(\lambda)) + |A(\lambda)|\lambda + c_S(A(\lambda)) \leq c_F(X^*) + |X^*|\lambda + \gamma_S c_S(X^*)$ und für alle $\lambda \geq 0$ folgt

$$c_F(A(\lambda)) + c_S(A(\lambda)) \leq c_F(X^*) + \gamma_S c_S(X^*) + (k - |A(\lambda)|)\lambda. \quad (22.9)$$

Ist $|A(0)| \leq k$, so ist $A(0)$ eine zulässige Lösung mit Kosten höchstens gleich γ_S mal dem Optimum, womit wir fertig sind.

Ist andererseits $|A(0)| > k$, so beachte, dass $|A(f_{\max} + \gamma_S |\mathcal{D}| c_{\max} + 1)| = 1 \leq k$. Setze $\lambda' := 0$ und $\lambda'' := f_{\max} + \gamma_S |\mathcal{D}| c_{\max} + 1$, und wende binäre Suche an, unter Erhaltung der Eigenschaft $|A(\lambda'')| \leq k < |A(\lambda')|$. Nach $O(\log |\mathcal{D}| + \log f_{\max} + \log c_{\max})$ Iterationen, wobei wir in jeder Iteration einen der beiden Werte λ', λ'' auf deren arithmetischen Mittelwert setzen, je nachdem, ob $|A(\frac{\lambda' + \lambda''}{2})| \leq k$ oder nicht, haben wir $\lambda'' - \lambda' \leq \frac{1}{|\mathcal{D}|^2}$. (Beachte, dass diese binäre Suche funktioniert, obwohl $\lambda \mapsto |A(\lambda)|$ i. A. nicht monoton ist.)

Ist $|A(\lambda'')| = k$, so folgt aus (22.9), dass $A(\lambda'')$ eine zulässige Lösung mit Kosten höchstens gleich γ_S mal dem Optimum ist. Damit sind wir fertig. Wir werden jedoch nicht immer ein solches λ'' antreffen, weil $\lambda \mapsto |A(\lambda)|$ nicht immer monoton ist und um mehr als 1 springen kann (Archer, Rajagopalan und Shmoys [2003] haben gezeigt, wie man dieses Problem durch Stören der Kosten beseitigen kann; sie konnten dies aber nicht in polynomieller Zeit bewerkstelligen).

Also betrachten wir $X := A(\lambda')$ und $Y := A(\lambda'')$ und nehmen für die weitere Betrachtung an, dass $|X| > k > |Y|$. Ferner seien $\alpha := \frac{k - |Y|}{|X| - |Y|}$ und $\beta := \frac{|X| - k}{|X| - |Y|}$.

Wähle eine Teilmenge X' von X mit $|X'| = |Y|$, so dass $\min_{i \in X'} c_{ii'} = \min_{i \in X} c_{ii'}$ für jedes $i' \in Y$, wobei $c_{ii'} := \min_{j \in \mathcal{D}} (c_{ij} + c_{i'j})$.

Wir öffnen entweder alle Elemente von X' (mit der Wahrscheinlichkeit α) oder alle Elemente von Y (mit der Wahrscheinlichkeit $\beta = 1 - \alpha$). Zusätzlich wählen wir zufällig und gleichverteilt $k - |Y|$ Standorte aus der Menge $X \setminus X'$. Von den gewählten Standorten öffnen wir dann diejenigen, die nicht schon offen sind (beachte, dass X und Y nicht notwendigerweise disjunkt sind). Damit ist der Erwartungswert der gesamten Bereitstellungskosten höchstens $\alpha c_F(X) + \beta c_F(Y)$.

Sei $j \in \mathcal{D}$ und sei i' einer der am nächsten gelegenen Standorte in X und i'' einer der am nächsten gelegenen Standorte in Y . Verbinde j mit i' , falls i' offen ist, sonst mit i'' , falls i'' offen ist. Ist weder i' noch i'' offen, so verbinde j mit einem Standort $i''' \in X'$, der $c_{i''i'''} \leq c_{i'i''}$ minimiert.

Für $i' \in X'$ erhalten wir hiermit den Erwartungswert $\alpha c_{i'j} + \beta c_{i''j}$ für die Servicekosten und für $i' \in X \setminus X'$ höchstens den Erwartungswert

$$\begin{aligned} & \alpha c_{i'j} + (1 - \alpha)\beta c_{i''j} + (1 - \alpha)(1 - \beta)c_{i'''j} \\ & \leq \alpha c_{i'j} + \beta^2 c_{i''j} + \alpha\beta \left(c_{i''j} + \min_{j' \in \mathcal{D}} (c_{i''j'} + c_{i'''j'}) \right) \\ & \leq \alpha c_{i'j} + \beta^2 c_{i''j} + \alpha\beta(c_{i''j} + c_{i''j} + c_{i'j}) \\ & = \alpha(1 + \beta)c_{i'j} + \beta(1 + \alpha)c_{i''j}. \end{aligned}$$

Somit ist der Erwartungswert der Gesamt servicekosten höchstens gleich

$$(1 + \max\{\alpha, \beta\})(\alpha c_S(X) + \beta c_S(Y)) \leq \left(2 - \frac{1}{|\mathcal{D}|}\right)(\alpha c_S(X) + \beta c_S(Y)).$$

Insgesamt ist dann unter Benutzung von (22.9) der Erwartungswert der Kosten höchstens gleich

$$\begin{aligned} & \left(2 - \frac{1}{|\mathcal{D}|}\right) \left(\alpha(c_F(X) + c_S(X)) + \beta(c_F(Y) + c_S(Y)) \right) \\ & \leq \left(2 - \frac{1}{|\mathcal{D}|}\right) \left(c_F(X^*) + \gamma_S c_S(X^*) + (\lambda'' - \lambda') \frac{(|X| - k)(k - |Y|)}{|X| - |Y|} \right) \\ & \leq \left(2 - \frac{1}{|\mathcal{D}|}\right) \left(c_F(X^*) + \gamma_S c_S(X^*) + (\lambda'' - \lambda') \frac{|X| - |Y|}{4} \right) \\ & \leq \left(2 - \frac{1}{|\mathcal{D}|}\right) \left(c_F(X^*) + \gamma_S c_S(X^*) + \frac{1}{4|\mathcal{D}|} \right) \\ & \leq \left(2 - \frac{1}{|\mathcal{D}|}\right) \left(1 + \frac{1}{4|\mathcal{D}|} \right) (c_F(X^*) + \gamma_S c_S(X^*)) \\ & \leq \left(2 - \frac{1}{2|\mathcal{D}|}\right) (c_F(X^*) + \gamma_S c_S(X^*)), \end{aligned}$$

also höchstens gleich $2\gamma_S(c_F(X^*) + c_S(X^*))$.

Beachte, dass der Erwartungswert der Kosten leicht zu berechnen ist, sogar unter der Bedingung, dass eine Standortmenge Z mit Wahrscheinlichkeit 1 geöffnet wird und weitere $k - |Z|$ zufällig gewählte Standorte einer anderen Standortmenge geöffnet werden. Somit kann man diesen Algorithmus mittels der Methode der bedingten Wahrscheinlichkeiten derandomisieren: Zunächst öffne man X' oder Y , je nachdem, wo die Schranke für den Erwartungswert der Kosten höchstens gleich $\left(2 - \frac{1}{|\mathcal{D}|}\right)(\alpha(c_F(X) + c_S(X)) + \beta(c_F(Y) + c_S(Y)))$ ist, und dann öffne man schrittweise Standpunkte aus $X \setminus X'$, so dass diese Schranke weiter gilt. \square

Insbesondere erhalten wir mit dem DUAL-FITTING-ALGORITHMUS (Korollar 22.7) einen 4-Approximationsalgorithmus für das METRISCHE k -STANDORTPROBLEM mit ganzzahligen Daten. Der erste Approximationsalgorithmus mit konstanter Gütegarantie für das METRISCHE k -STANDORTPROBLEM stammt von Charikar et al. [2002].

Die Laufzeit der binären Suche ist schwach polynomiell und funktioniert nur für ganzzahlige Daten. Wir können sie aber streng polynomiell machen, indem wir die Inputdaten diskretisieren:

Lemma 22.15. *Für jede gegebene Instanz I des METRISCHEN k -STANDORTPROBLEMS zusammen mit einem $\gamma_{\max} \geq 1$ und einem $0 < \epsilon \leq 1$ können wir entscheiden, ob $\text{OPT}(I) = 0$, und falls nicht, eine andere Instanz I' in $O(|\mathcal{F}||\mathcal{D}| \log(|\mathcal{F}||\mathcal{D}|))$ -Zeit erzeugen, so dass sämtliche Service- und Bereitstellungskosten ganze Zahlen aus $\{0, 1, \dots, \lceil \frac{2\gamma_{\max}(k+|\mathcal{D}|)^3}{\epsilon} \rceil\}$ sind und dass für jedes $1 \leq \gamma \leq \gamma_{\max}$ gilt: Jede Lösung für I' mit Kosten höchstens gleich $\gamma \text{OPT}(I')$ ist eine Lösung für I mit Kosten höchstens gleich $\gamma(1+\epsilon)\text{OPT}(I)$.*

Beweis: Sei $n := k + |\mathcal{D}|$. Für eine gegebene Instanz I berechnen wir zunächst eine obere und eine untere Schranke für $\text{OPT}(I)$, die sich um höchstens den Faktor $2n^2 - 1$ unterscheiden: Für jedes $B \in \{f_i : i \in \mathcal{F}\} \cup \{c_{ij} : i \in \mathcal{F}, j \in \mathcal{D}\}$ betrachten wir den bipartiten Graphen $G_B := (\mathcal{D} \cup \mathcal{F}, \{(i, j) : i \in \mathcal{F}, j \in \mathcal{D}, f_i \leq B, c_{ij} \leq B\})$.

Das kleinste B , für welches die Elemente von \mathcal{D} zu höchstens k paarweise verschiedenen mindestens einen Standort enthaltenden Zusammenhangskomponenten von G_B gehören, ist eine untere Schranke für $\text{OPT}(I)$. Diese Zahl B kann in $O(|\mathcal{F}||\mathcal{D}| \log(|\mathcal{F}||\mathcal{D}|))$ -Zeit mittels einer unkomplizierten Variante von KRUSKALS ALGORITHMUS für aufspannende Bäume mit minimalen Kosten gefunden werden.

Ferner können wir für dieses B in jeder ein Element von \mathcal{D} enthaltenden Zusammenhangskomponente von G_B einen beliebigen Standort wählen und jeden Kunden so verbinden, dass die Servicekosten höchstens $(2|\mathcal{D}| - 1)B$ betragen (unter der Bedingung, dass diese metrisch sind). Somit ist $\text{OPT}(I) \leq kB + (2|\mathcal{D}| - 1)|\mathcal{D}|B < (2n^2 - 1)B$, sofern $B \neq 0$. Im Falle $B = 0$ wären wir fertig.

Demnach brauchen wir Bereitstellungs- und Servicekosten nur bis zu einer Höhe von $B' := 2\gamma_{\max}n^2B$ zu berücksichtigen. Wir erhalten I' aus I , indem wir jedes c_{ij} auf $\lceil \frac{\min\{B', c_{ij}\}}{\delta} \rceil$ und jedes f_i auf $\lceil \frac{\min\{B', f_i\}}{\delta} \rceil$ setzen, wobei $\delta = \frac{\epsilon B}{n}$. Nun sind alle Inputzahlen ganze Zahlen aus $\{0, 1, \dots, \lceil \frac{2\gamma_{\max}n^3}{\epsilon} \rceil\}$.

Es gilt

$$\text{OPT}(I') \leq \frac{\text{OPT}(I)}{\delta} + n = \frac{\text{OPT}(I) + \epsilon B}{\delta} < \frac{(2n^2 - 1)B + \epsilon B}{\delta} \leq \frac{2n^2 B}{\delta} = \frac{B'}{\gamma_{\max} \delta}.$$

Somit enthält eine Lösung für I' mit Kosten höchstens gleich $\gamma \text{OPT}(I')$ kein Element mit Kosten $\lceil \frac{B'}{\delta} \rceil$ und ist demnach eine Lösung für I mit Kosten höchstens gleich

$$\delta\gamma \text{OPT}(I') \leq \gamma(\text{OPT}(I) + \epsilon B) \leq \gamma(1 + \epsilon)\text{OPT}(I).$$

□

Korollar 22.16. Es gibt einen streng polynomiellen 4-Approximationsalgorithmus für das METRISCHE k -STANDORTPROBLEM.

Beweis: Wende Lemma 22.15 mit $\gamma_{\max} = 4$ und $\epsilon = \frac{1}{4|\mathcal{D}|}$ an und dann Satz 22.14 mit dem DUAL-FITTING-ALGORITHMUS auf die resultierende Instanz. Nach Korollar 22.7 ist $\gamma_S = 2$ und wir erhalten für jedes $\emptyset \neq X^* \subseteq \mathcal{F}$ eine Lösung mit Gesamtkosten höchstens gleich

$$\left(2 - \frac{1}{2|\mathcal{D}|}\right) \left(1 + \frac{1}{4|\mathcal{D}|}\right) (c_F(X^*) + \gamma_{SCS}(X^*)) \leq 4(c_F(X^*) + c_S(X^*)).$$

□

Zhang [2007] hat einen 3.733-Approximationsalgorithmus für das METRISCHE k -STANDORTPROBLEM gefunden. Dieser Algorithmus benutzt lokale Suchmethoden ähnlich denen, die wir im nächsten Abschnitt besprechen werden.

22.6 Lokale Suche

Wie wir bereits in Abschnitt 21.3 besprochen haben, ist die lokale Suche eine Methode, die in der Praxis oft erfolgreich eingesetzt wird, obwohl normalerweise keine gute Approximationsgüte berechnet werden kann. Es war somit überraschend, als gezeigt wurde, dass Standortprobleme gut mittels lokaler Suche approximiert werden können. Dies wurde zuerst von Korupolu, Plaxton und Rajaraman [2000] untersucht und führte anschließend zu mehreren starken Resultaten. Einige davon werden wir in den nächsten beiden Abschnitten besprechen.

Für das METRISCHE k -MEDIAN-PROBLEM liefert die lokale Suche die bislang beste Approximationsgüte. Bevor wir diese Tatsache erläutern, werden wir den einfachsten aller auf der lokalen Suche basierenden Algorithmen betrachten: Wir beginnen mit einer beliebigen zulässigen Lösung (d. h. einer Menge von k Standorten) und verbessern sie dann durch einzelne Vertauschungen. Beachte, dass wir nur die Servicekosten zu betrachten brauchen, da die Bereitstellungskosten im k -MEDIAN-PROBLEM verschwinden. Ferner können wir o. B. d. A. annehmen, dass eine Lösung genau k Standorte enthält.

Satz 22.17. (Arya et al. [2004]) Betrachte eine Instanz des METRISCHEN k -MEDIAN-PROBLEMS. Sei X eine zulässige Lösung und X^* eine optimale Lösung. Gilt $c_S((X \setminus \{x\}) \cup \{y\}) \geq c_S(X)$ für alle $x \in X$ und $y \in X^*$, so folgt $c_S(X) \leq 5c_S(X^*)$.

Beweis: Wir betrachten optimale Zuordnungen σ und σ^* der Kunden zu den k Standorten in X bzw. X^* . Wir sagen: $x \in X$ erfasst $y \in X^*$, falls $|\{j \in \mathcal{D} : \sigma(j) =$

$x, \sigma^*(j) = y\} | > \frac{1}{2} |\{j \in \mathcal{D} : \sigma^*(j) = y\}|$. Jedes $y \in X^*$ wird von höchstens einem $x \in X$ erfasst.

Sei $\pi : \mathcal{D} \rightarrow \mathcal{D}$ eine Bijektion, für die für alle $j \in \mathcal{D}$ gilt:

- $\sigma^*(\pi(j)) = \sigma^*(j)$;
- gilt $\sigma(\pi(j)) = \sigma(j)$, so wird $\sigma^*(j)$ von $\sigma(j)$ erfasst.

Eine solche Abbildung π kann man leicht dadurch erhalten, dass man für jedes $y \in X^*$ die Elemente von $\{j \in \mathcal{D} : \sigma^*(j) = y\} = \{j_0, \dots, j_{t-1}\}$ so ordnet, dass Kunden j mit identischem $\sigma(j)$ einander folgen, und wir $\pi(j_k) := j_{k'}$ setzen, wobei $k' = (k + \lfloor \frac{t}{2} \rfloor) \bmod t$.

Eine *Vertauschung* sei ein Element von $X \times X^*$. Für eine gegebene Vertauschung (x, y) ist x der Ursprung und y das Ziel. Wir werden nun k Vertauschungen definieren, so dass jedes $y \in X^*$ genau einmal als Ziel vorkommt.

Erfasst ein $x \in X$ nur einen Standort $y \in X^*$, so betrachten wir die Vertauschung (x, y) . Gibt es l solche Vertauschungen, so sind sowohl in X als auch in X^* genau $k-l$ Elemente übrig. Einige der restlichen Elemente von X (höchstens $\frac{k-l}{2}$) könnten mindestens zwei Standorte von X^* erfassen, diese werden wir aber nicht betrachten. Für jeden übrig gebliebenen Standort $y \in X^*$ wählen wir ein $x \in X$, so dass x keinen Standort erfasst und jedes $x \in X$ der Ursprung von höchstens zwei solchen Vertauschungen ist.

Diese Vertauschungen analysieren wir nun der Reihe nach. Betrachte die Vertauschung (x, y) und setze $X' := (X \setminus \{x\}) \cup \{y\}$. Dann folgt $c_S(X') \geq c_S(X)$. Transformiere $\sigma : \mathcal{D} \rightarrow X$ in eine neue Zuordnung $\sigma' : \mathcal{D} \rightarrow X'$ mittels der folgenden Neuzuordnungen:

Kunden $j \in \mathcal{D}$ mit $\sigma^*(j) = y$ werden y zugeordnet. Kunden $j \in \mathcal{D}$ mit $\sigma(j) = x$ und $\sigma^*(j) = y' \in X^* \setminus \{y\}$ werden $\sigma(\pi(j))$ zugeordnet; beachte, dass $\sigma(\pi(j)) \neq x$, da y' nicht von x erfasst wird. Für alle weiteren Kunden bleibt die Zuordnung unverändert.

Nach Definition von σ gilt $c_{\sigma(\pi(j))j} \geq \min_{i \in X} c_{ij} = c_{\sigma(j)j}$. Damit haben wir

$$\begin{aligned} 0 &\leq c_S(X') - c_S(X) \\ &\leq \sum_{j \in \mathcal{D}: \sigma^*(j)=y} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + \sum_{j \in \mathcal{D}: \sigma(j)=x, \sigma^*(j) \neq y} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}) \\ &\leq \sum_{j \in \mathcal{D}: \sigma^*(j)=y} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + \sum_{j \in \mathcal{D}: \sigma(j)=x} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}). \end{aligned}$$

Nun summieren wir über alle Vertauschungen. Beachte, dass jeder Standort von X^* das Ziel genau einer Vertauschung ist, somit ist die Summe der ersten Terme gleich $c_S(X^*) - c_S(X)$. Ferner ist jedes $x \in X$ der Ursprung von höchstens zwei Vertauschungen. Da π eine Bijektion ist, folgt damit

$$\begin{aligned}
0 &\leq \sum_{j \in \mathcal{D}} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + 2 \sum_{j \in \mathcal{D}} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}) \\
&\leq c_S(X^*) - c_S(X) + 2 \sum_{j \in \mathcal{D}} (c_{\sigma^*(j)j} + c_{\sigma^*(j)\pi(j)} + c_{\sigma(\pi(j))\pi(j)} - c_{\sigma(j)j}) \\
&= c_S(X^*) - c_S(X) + 2 \sum_{j \in \mathcal{D}} (c_{\sigma^*(j)j} + c_{\sigma^*(\pi(j))\pi(j)}) \\
&= c_S(X^*) - c_S(X) + 4c_S(X^*).
\end{aligned}$$

□

Somit ist ein lokales Optimum eine 5-Approximation. Hieraus ergibt sich jedoch keine Aussage über die Laufzeit bis zu einem lokalen Optimum; es ist vorstellbar, dass die Anzahl der Schritte bis zu einem lokalen Optimum exponentiell sein könnte. Mittels Diskretisierung der Kosten erhalten wir aber eine streng polynomiale Laufzeit:

Korollar 22.18. *Sei $0 < \epsilon \leq 1$. Dann ist der folgende Algorithmus ein streng polynomieller $(5+\epsilon)$ -Approximationsalgorithmus für das METRISCHE k -MEDIAN-PROBLEM: Transformiere die Instanz wie in Lemma 22.15 mit $\gamma_{\max} = 5$ und $\frac{\epsilon}{5}$ statt ϵ , beginne mit einer beliebigen Menge von k Standorten und führe Servicekosten senkende Vertauschungen so lange wie möglich aus.*

Beweis: Da alle Servicekosten der neuen Instanz ganze Zahlen aus $\{0, 1, \dots, \lceil \frac{50(k+|\mathcal{D}|)^3}{\epsilon} \rceil\}$ sind, können wir höchstens $|\mathcal{D}| \lceil \frac{50(k+|\mathcal{D}|)^3}{\epsilon} \rceil$ aufeinander folgende die Gesamtsevicekosten senkende Vertauschungen ausführen. □

Die Verwendung von Mehrfachvertauschungen verbessert die Approximationsgüte deutlich:

Satz 22.19. (Arya et al. [2004]) *Betrachte eine Instanz des METRISCHEN k -MEDIAN-PROBLEMS. Sei $p \in \mathbb{N}$, X eine zulässige Lösung und X^* eine optimale Lösung. Gilt $c_S((X \setminus A) \cup B) \geq c_S(X)$ für alle $A \subseteq X$ und $B \subseteq X^*$ mit $|A| = |B| \leq p$, so folgt $c_S(X) \leq (3 + \frac{2}{p})c_S(X^*)$.*

Beweis: Seien σ und σ^* wiederum optimale Zuordnungen der Kunden zu den k Standorten in X bzw. X^* . Für jedes $A \subseteq X$ sei $C(A)$ die Menge der von A erfassten Standorte in X^* , d.h.

$$C(A) := \left\{ y \in X^* : |\{j \in \mathcal{D} : \sigma(j) \in A, \sigma^*(j) = y\}| > \frac{1}{2} |\{j \in \mathcal{D} : \sigma^*(j) = y\}| \right\}.$$

Wir partitionieren $X = A_1 \dot{\cup} \dots \dot{\cup} A_r$ und $X^* = B_1 \dot{\cup} \dots \dot{\cup} B_r$ wie folgt:

Sei $\{x \in X : C(\{x\}) \neq \emptyset\} =: \{x_1, \dots, x_s\} =: \bar{X}$.

Setze $r := \max\{s, 1\}$.

For $i = 1$ **to** $r - 1$ **do:**

 Setze $A_i := \{x_i\}$.

While $|A_i| < |C(A_i)|$ **do:**

 Füge ein Element $x \in X \setminus (A_1 \cup \dots \cup A_i \cup \bar{X})$ zu A_i hinzu.

 Setze $B_i := C(A_i)$.

 Setze $A_r := X \setminus (A_1 \cup \dots \cup A_{r-1})$ und $B_r := X^* \setminus (B_1 \cup \dots \cup B_{r-1})$.

Es ist klar, dass dieser Algorithmus $|A_i| = |B_i| \geq 1$ für $i = 1, \dots, r$ gewährleistet, und dass die Mengen A_1, \dots, A_r paarweise disjunkt sind und die Mengen B_1, \dots, B_r auch. Beachte, dass wir immer ein Element hinzufügen können, falls $|A_i| < |C(A_i)|$, da dann

$$\begin{aligned} & |X \setminus (A_1 \cup \dots \cup A_i \cup \bar{X})| \\ = & |X| - |A_1| - \dots - |A_i| - |\{x_{i+1}, \dots, x_r\}| \\ > & |X^*| - |C(A_1)| - \dots - |C(A_i)| - |C(\{x_{i+1}\})| - \dots - |C(\{x_r\})| \\ = & |X^* \setminus (C(A_1) \cup \dots \cup C(A_i) \cup C(\{x_{i+1}\}) \cup \dots \cup C(\{x_r\}))| \\ \geq & 0. \end{aligned}$$

Sei $\pi : \mathcal{D} \rightarrow \mathcal{D}$ eine Bijektion, so dass für alle $j \in \mathcal{D}$ Folgendes gilt:

- $\sigma^*(\pi(j)) = \sigma^*(j)$;
- ist $\sigma(\pi(j)) = \sigma(j)$, so wird $\sigma^*(j)$ von $\sigma(j)$ erfasst;
- ist $\sigma(j) \in A_i$ und $\sigma(\pi(j)) \in A_i$ für ein $i \in \{1, \dots, r\}$, so wird $\sigma^*(j)$ von A_i erfasst.

Eine solche Abbildung π kann man fast genauso wie im Beweis von Satz 22.17 erhalten.

Nun definieren wir eine Menge von Vertauschungen (A, B) mit $|A| = |B| \leq p$, $A \subseteq X$ und $B \subseteq X^*$. Jeder Vertauschung wird ein positives Gewicht zugewiesen. Die Vertauschung (A, B) bedeutet, dass X durch $X' := (X \setminus A) \cup B$ ersetzt wird; wir sagen, dass A die Ursprungsmenge und B die Zielmenge ist.

Für jedes $i \in \{1, \dots, r\}$ mit $|A_i| \leq p$ betrachten wir die Vertauschung (A_i, B_i) mit Gewicht 1. Für jedes $i \in \{1, \dots, r\}$ mit $|A_i| = q > p$ betrachten wir die Vertauschung $(\{x\}, \{y\})$ mit Gewicht $\frac{1}{q-1}$ für jedes $x \in A_i \setminus \{x_i\}$ und $y \in B_i$. Jedes $y \in X^*$ kommt in der Zielmenge von Vertauschungen mit Gesamtgewicht 1 vor und jedes $x \in X$ in der Ursprungsmenge von Vertauschungen mit Gesamtgewicht höchstens gleich $\frac{p+1}{p}$.

Wie bei den einzelnen Vertauschungen ordnen wir Kunden wie folgt neu zu. Genauer: Für eine Vertauschung (A, B) ordnen wir alle $j \in \mathcal{D}$ mit $\sigma^*(j) \in B$ dem Standort $\sigma^*(j)$ und alle $j \in \mathcal{D}$ mit $\sigma^*(j) \notin B$ und $\sigma(j) \in A$ dem Standort $\sigma(\pi(j))$ zu. Beachte, dass $B \supseteq C(A)$ für jede der betrachteten Vertauschungen (A, B) . Für alle $j \in \mathcal{D}$ mit $\sigma(j) \in A$ und $\sigma^*(j) \notin B$ folgt somit $\sigma(\pi(j)) \notin A$. Damit können wir die durch die Vertauschungen verursachte Kostensteigerung unter

Hinzunahme der nach Definition von σ gültigen Ungleichung $c_{\sigma(\pi(j))j} \geq c_{\sigma(j)j}$ wie folgt beschränken:

$$\begin{aligned} 0 &\leq c_S(X') - c_S(X) \\ &\leq \sum_{j \in \mathcal{D}: \sigma^*(j) \in B} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + \sum_{j \in \mathcal{D}: \sigma(j) \in A, \sigma^*(j) \notin B} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}) \\ &\leq \sum_{j \in \mathcal{D}: \sigma^*(j) \in B} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + \sum_{j \in \mathcal{D}: \sigma(j) \in A} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}). \end{aligned}$$

Nehmen wir nun die gewichtete Summe über alle Vertauschungen, so folgt, da π eine Bijektion ist:

$$\begin{aligned} 0 &\leq \sum_{j \in \mathcal{D}} (c_{\sigma^*(j)j} - c_{\sigma(j)j}) + \frac{p+1}{p} \sum_{j \in \mathcal{D}} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}) \\ &\leq c_S(X^*) - c_S(X) + \frac{p+1}{p} \sum_{j \in \mathcal{D}} (c_{\sigma^*(j)j} + c_{\sigma^*(j)\pi(j)} + c_{\sigma(\pi(j))\pi(j)} - c_{\sigma(j)j}) \\ &= c_S(X^*) - c_S(X) + \frac{p+1}{p} \sum_{j \in \mathcal{D}} (c_{\sigma^*(j)j} + c_{\sigma^*(\pi(j))\pi(j)}) \\ &= c_S(X^*) - c_S(X) + 2 \frac{p+1}{p} c_S(X^*). \end{aligned}$$

□

Arya et al. [2004] haben ferner gezeigt, dass diese Approximationsgüte die bestmögliche ist. Wie bei Korollar 22.18 implizieren Lemma 22.15 und Satz 22.19 einen $(3 + \epsilon)$ -Approximationsalgorithmus für jedes $\epsilon > 0$. Dies ist die momentan beste Approximationsgüte für das METRISCHE k -MEDIAN-PROBLEM.

Wir können ähnliche Methoden auf das METRISCHE UNBESCHRÄNKTE STANDORTPROBLEM anwenden und erhalten damit einen einfachen auf lokaler Suche basierenden Approximationsalgorithmus:

Satz 22.20. (Arya et al. [2004]) *Betrachte eine Instanz des METRISCHEN UNBESCHRÄNKTN STANDORTPROBLEMS. Seien X und X^* beliebige zulässige Lösungen. Ist für jedes $x \in X$ und jedes $y \in \mathcal{F} \setminus X$ weder $X \setminus \{x\}$ noch $X \cup \{y\}$ noch $(X \setminus \{x\}) \cup \{y\}$ besser als X , so gilt $c_S(X) \leq c_F(X^*) + c_S(X^*)$ und $c_F(X) \leq c_F(X^*) + 2c_S(X^*)$.*

Beweis: Wir benutzen dieselbe Notation wie in den vorangegangenen Beweisen. Insbesondere seien σ und σ^* optimale Zuordnungen der Kunden zu X bzw. X^* .

Der Beweis der ersten Ungleichung folgt leicht, indem wir für jedes $y \in X^*$ die Operation des Hinzufügens von y zu X betrachten, die die Kosten um höchstens $f_y + \sum_{j \in \mathcal{D}: \sigma^*(j)=y} (c_{\sigma^*(j)j} - c_{\sigma(j)j})$ steigert. Summieren wir diese Werte, so folgt, dass $c_F(X^*) + c_S(X^*) - c_S(X)$ nichtnegativ ist.

Sei wiederum $\pi : \mathcal{D} \rightarrow \mathcal{D}$ eine Bijektion, für die Folgendes für alle $j \in \mathcal{D}$ gilt:

- $\sigma^*(\pi(j)) = \sigma^*(j)$;
- ist $\sigma(\pi(j)) = \sigma(j)$, so wird $\sigma^*(j)$ von $\sigma(j)$ erfasst und $\pi(j) = j$.

Eine solche Abbildung π erhält man wie im Beweis von Satz 22.17, nachdem man Folgendes ausgeführt hat: Setze $\pi(j) := j$ für $|\{j \in \mathcal{D} : \sigma^*(j) = y, \sigma(j) = x\}| - |\{j \in \mathcal{D} : \sigma^*(j) = y, \sigma(j) \neq x\}|$ Elemente $j \in \mathcal{D}$ mit $\sigma^*(j) = y$ und $\sigma(j) = x$ für beliebige $x \in X$, $y \in X^*$, wobei y von x erfasst wird.

Um die Bereitstellungskosten von X zu beschränken, sei $x \in X$ und setze $\mathcal{D}_x := \{j \in \mathcal{D} : \sigma(j) = x\}$. Erfasst x kein $y \in X^*$, so ziehen wir in Betracht, x wegzulassen und jedes $j \in \mathcal{D}_x$ neu $\sigma(\pi(j)) \in X \setminus \{x\}$ zuzuordnen. Damit folgt

$$0 \leq -f_x + \sum_{j \in \mathcal{D}_x} (c_{\sigma(\pi(j))j} - c_{xj}). \quad (22.10)$$

Ist die Menge $C(\{x\})$ der von x erfassten Standorte nicht leer, so sei $y \in C(\{x\})$ ein nächster Standort in $C(\{x\})$ (d. h. $\min_{j \in \mathcal{D}} (c_{xj} + c_{yj})$ ist minimal). Wir betrachten das Hinzufügen eines jeden Standortes $y' \in C(\{x\}) \setminus \{y\}$, der die Kosten um mindestens Null und höchstens dem folgenden Betrag steigert:

$$f_{y'} + \sum_{j \in \mathcal{D}_x : \sigma^*(j) = y', \pi(j) = j} (c_{\sigma^*(j)j} - c_{xj}). \quad (22.11)$$

Ferner betrachten wir die Vertauschung $(\{x\}, \{y\})$. Für $j \in \mathcal{D}_x$ ordnen wir j neu $\sigma(\pi(j))$ zu, falls $\pi(j) \neq j$, und y sonst.

Die neuen Servicekosten für $j \in \mathcal{D}_x$ sind höchstens gleich $c_{\sigma(\pi(j))j}$ im ersten Fall, höchstens gleich $c_{\sigma^*(j)j}$, falls $\pi(j) = j$ und $\sigma^*(j) = y$, und

$$c_{yj} \leq c_{xj} + \min_{k \in \mathcal{D}} (c_{xk} + c_{yk}) \leq c_{xj} + \min_{k \in \mathcal{D}} (c_{xk} + c_{\sigma^*(j)k}) \leq 2c_{xj} + c_{\sigma^*(j)j}$$

sonst, wobei die mittlere Ungleichung gilt, weil $\sigma^*(j)$ von x erfasst wird, falls $\pi(j) = j$.

Insgesamt erhöht die Vertauschung von x mit y die Kosten um mindestens Null und höchstens

$$\begin{aligned} & f_y - f_x - \sum_{j \in \mathcal{D}_x} c_{xj} + \sum_{j \in \mathcal{D}_x : \pi(j) \neq j} c_{\sigma(\pi(j))j} \\ & + \sum_{j \in \mathcal{D}_x : \pi(j) = j, \sigma^*(j) = y} c_{\sigma^*(j)j} + \sum_{j \in \mathcal{D}_x : \pi(j) = j, \sigma^*(j) \neq y} (2c_{xj} + c_{\sigma^*(j)j}). \end{aligned} \quad (22.12)$$

Addieren wir (22.11) und (22.12), die beide nicht negativ sind, so erhalten wir

$$\begin{aligned} 0 & \leq \sum_{y' \in C(\{x\})} f_{y'} - f_x + \sum_{j \in \mathcal{D}_x : \pi(j) \neq j} (c_{\sigma(\pi(j))j} - c_{xj}) \\ & + \sum_{j \in \mathcal{D}_x : \pi(j) = j, \sigma^*(j) = y} (c_{\sigma^*(j)j} - c_{xj}) + \sum_{j \in \mathcal{D}_x : \pi(j) = j, \sigma^*(j) \neq y} 2c_{\sigma^*(j)j} \\ & \leq \sum_{y' \in C(\{x\})} f_{y'} - f_x + \sum_{j \in \mathcal{D}_x : \pi(j) \neq j} (c_{\sigma(\pi(j))j} - c_{xj}) + 2 \sum_{j \in \mathcal{D}_x : \pi(j) = j} c_{\sigma^*(j)j}. \end{aligned} \quad (22.13)$$

Summieren wir (22.10) bzw. (22.13) über alle $x \in X$, so erhalten wir

$$\begin{aligned}
0 &\leq \sum_{x \in X} \sum_{y' \in C(\{x\})} f_{y'} - c_F(X) + \sum_{j \in \mathcal{D}: \pi(j) \neq j} (c_{\sigma(\pi(j))j} - c_{\sigma(j)j}) \\
&\quad + 2 \sum_{j \in \mathcal{D}: \pi(j)=j} c_{\sigma^*(j)j} \\
&\leq c_F(X^*) - c_F(X) + \sum_{j \in \mathcal{D}: \pi(j) \neq j} (c_{\sigma^*(j)j} + c_{\sigma^*(j)\pi(j)} + c_{\sigma(\pi(j))\pi(j)} - c_{\sigma(j)j}) \\
&\quad + 2 \sum_{j \in \mathcal{D}: \pi(j)=j} c_{\sigma^*(j)j} \\
&= c_F(X^*) - c_F(X) + 2c_S(X^*).
\end{aligned}$$

□

Mit Lemma 22.15 impliziert dies einen $(3 + \epsilon)$ -Approximationsalgorithmus für jedes $\epsilon > 0$. Kombinieren wir dies mit Satz 22.10, so erhalten wir einen 2,375-Approximationsalgorithmus (Aufgabe 12). Charikar und Guha [2005] haben dieselbe Approximationsgüte für einen sehr ähnlichen auf lokaler Suche basierenden Algorithmus berechnet.

22.7 Beschränkte Standortprobleme

Ein Hauptvorteil der auf lokaler Suche basierenden Algorithmen ist ihre Flexibilität; sie können auf beliebige Kostenfunktionen und sogar bei zusätzlichen komplizierten Nebenbedingungen angewendet werden. Für die meisten Standortprobleme, bei denen die Standorte beschränkte Kapazität haben, ist die lokale Suche die einzige momentan bekannte Methode mit einer Approximationsgüte.

Man hat mehrere verschiedene beschränkte Standortprobleme studiert. Mahdian und Pál [2003] haben das folgende allgemeine Problem definiert, welches einige wichtige Spezialfälle enthält:

UNIVERSELLES STANDORTPROBLEM

Instanz: Eine endliche Kundenmenge \mathcal{D} und eine endliche Menge \mathcal{F} von möglichen Standorten; Eine Metrik c auf $V := \mathcal{D} \cup \mathcal{F}$, d. h. Distanzen $c_{ij} \geq 0$ ($i, j \in V$) mit $c_{ii} = 0$, $c_{ij} = c_{ji}$ und $c_{ij} + c_{jk} \geq c_{ik}$ für alle $i, j, k \in V$; Nachfragen $d_j \geq 0$ für alle $j \in \mathcal{D}$ und links-stetige, nicht fallende Kostenfunktionen $f_i : \mathbb{R}_+ \rightarrow \mathbb{R}_+ \cup \{\infty\}$ für alle $i \in \mathcal{F}$.

Aufgabe: Bestimme $x_{ij} \in \mathbb{R}_+$ für $i \in \mathcal{F}$ und $j \in \mathcal{D}$ mit $\sum_{i \in \mathcal{F}} x_{ij} = d_j$ für alle $j \in \mathcal{D}$, so dass $c(x) := c_F(x) + c_S(x)$ minimiert wird, wobei

$$c_F(x) := \sum_{i \in \mathcal{F}} f_i \left(\sum_{j \in \mathcal{D}} x_{ij} \right) \quad \text{und} \quad c_S(x) := \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij}.$$

Man kann $f_i(z)$ als die Kosten für die Bereitstellung der Kapazität z im Standort i interpretieren. Wir müssen noch angeben, wie die Funktionen f_i gegeben sind: Wir setzen ein Orakel voraus, das $f_i(u)$ und $\max\{\delta \in \mathbb{R} : u + \delta \geq 0, f_i(u + \delta) - f_i(u) + c|\delta| \leq t\}$ für jedes $i \in \mathcal{F}$, $u, c \in \mathbb{R}_+$ und $t \in \mathbb{R}$ berechnet. Dies ist eine natürliche Annahme, da es trivial ist, ein solches Orakel für die wichtigsten Spezialfälle des UNIVERSELLEN STANDORTPROBLEMS zu implementieren. Diese sind:

- das METRISCHE UNBESCHRÄNKTE STANDORTPROBLEM. Hier ist $d_j = 1$ ($j \in \mathcal{D}$) und $f_i(0) = 0$, $f_i(z) = t_i$ für ein $t_i \in \mathbb{R}_+$ und alle $z > 0$ ($i \in \mathcal{F}$).
- das METRISCHE BESCHRÄNKTE STANDORTPROBLEM. Hier ist $f_i(0) = 0$, $f_i(z) = t_i$ für $0 < z \leq u_i$ und $f_i(z) = \infty$ für $z > u_i$, wobei $u_i, t_i \in \mathbb{R}_+$ ($i \in \mathcal{F}$).
- das METRISCHE SCHWACH-BESCHRÄNKTE STANDORTPROBLEM. Hier ist $d_j = 1$ ($j \in \mathcal{D}$) und $f_i(z) = \lceil \frac{z}{u_i} \rceil t_i$ für ein $u_i \in \mathbb{N}$, $t_i \in \mathbb{R}_+$ und alle $z \geq 0$ ($i \in \mathcal{F}$).

Beachte, dass es für den ersten und den dritten Spezialfall immer eine ganzzahlige optimale Lösung gibt. Während dies für den ersten Spezialfall trivial ist, folgt es für den dritten leicht, indem wir eine beliebige optimale Lösung y nehmen und das folgende Resultat auf $d_j = 1$ für $j \in \mathcal{D}$ und $z_i = \max\{z : f_i(z) \leq f_i(\sum_{j \in \mathcal{D}} y_{ij})\} \in \mathbb{Z}_+$ für $i \in \mathcal{F}$ anwenden:

Proposition 22.21. Seien \mathcal{D} und \mathcal{F} endliche Mengen, $d_j \geq 0$ ($j \in \mathcal{D}$), $z_i \geq 0$ ($i \in \mathcal{F}$) und $c_{ij} \geq 0$ ($i \in \mathcal{F}, j \in \mathcal{D}$) mit $\sum_{j \in \mathcal{D}} d_j \leq \sum_{i \in \mathcal{F}} z_i$. Dann kann man eine optimale Lösung für

$$\min \left\{ \sum_{i \in \mathcal{F}, j \in \mathcal{D}} c_{ij} x_{ij} : x \geq 0, \sum_{i \in \mathcal{F}} x_{ij} = d_j \ (j \in \mathcal{D}), \sum_{j \in \mathcal{D}} x_{ij} \leq z_i \ (i \in \mathcal{F}) \right\} \quad (22.14)$$

in $O(n^3 \log n)$ -Zeit finden, wobei $n = |\mathcal{D}| + |\mathcal{F}|$. Sind die d_j und z_i alle ganzzahlig, so gibt es eine ganzzahlige optimale Lösung.

Beweis: Es ist (22.14) äquivalent mit der wie folgt definierten Instanz (G, b, c) des HITCHCOCK-PROBLEMS: $G := (A \dot{\cup} B, A \times B)$, $A := \{v_j : j \in \mathcal{D}\} \dot{\cup} \{0\}$, $B := \{w_i : i \in \mathcal{F}\}$, $b(v_j) := d_j$ für $j \in \mathcal{D}$, $b(w_i) = -z_i$ für $i \in \mathcal{F}$, $b(0) := \sum_{i \in \mathcal{F}} z_i - \sum_{j \in \mathcal{D}} d_j$ und $c(v_j, w_i) := c_{ij}$, $c(0, w_i) := 0$ für $i \in \mathcal{F}$ und $j \in \mathcal{D}$. Demnach kann (22.14) mittels Satz 9.19 in $O(n^3 \log n)$ -Zeit gelöst werden. Ist b ganzzahlig, so liefert sowohl der MINIMUM-MEAN-CYCLE-CANCELLING-ALGORITHMUS als auch der SUKZESSIVE-KÜRZESTE-WEGE-ALGORITHMUS eine ganzzahlige optimale Lösung. \square

Die schwach-beschränkte Version kann recht leicht mittels einer zuerst von Jain und Vazirani [2001] vorgeschlagenen Methode auf die unbeschränkte Version reduziert werden:

Satz 22.22. (Mahdian, Ye und Zhang [2006]) Seien γ_F und γ_S Konstanten und A ein polynomieller Algorithmus mit der Eigenschaft, dass A für jede Instanz des METRISCHEN UNBESCHRÄNKTE STANDORTPROBLEMS eine Lösung X mit $c_F(X) + c_S(X) \leq \gamma_{FCF}(X^*) + \gamma_{SCS}(X^*)$ für jedes $\emptyset \neq X^* \subseteq \mathcal{F}$ berechnet. Dann gibt es einen $(\gamma_F + \gamma_S)$ -Approximationsalgorithmus für das METRISCHE SCHWACH-BESCHRÄNKTE STANDORTPROBLEM.

Beweis: Betrachte eine Instanz $I = (\mathcal{F}, \mathcal{D}, (c_{ij})_{i \in \mathcal{F}, j \in \mathcal{D}}, (f_i)_{i \in \mathcal{F}})$ des METRISCHEN SCHWACH-BESCHRÄNKTE STANDORTPROBLEMS, wobei $f_i(z) = \lceil \frac{z}{u_i} \rceil t_i$ für $i \in \mathcal{F}$ und $z \in \mathbb{R}_+$. Wir transformieren nun diese Instanz in die Instanz $I' = (\mathcal{F}, \mathcal{D}, (f'_i)_{i \in \mathcal{F}}, (c'_{ij})_{i \in \mathcal{F}, j \in \mathcal{D}})$ des METRISCHEN UNBESCHRÄNKTE STANDORTPROBLEMS, indem wir $f'_i := t_i$ und $c'_{ij} := c_{ij} + \frac{t_i}{u_i}$ für $i \in \mathcal{F}$ und $j \in \mathcal{D}$ setzen. (Beachte, dass c' eine Metrik ist, falls c eine ist.)

Wir wenden nun A auf I' an und bestimmen eine Lösung $X \subseteq \mathcal{F}$ und eine Zuordnung $\sigma : \mathcal{D} \rightarrow X$. Setze $x_{ij} := 1$, falls $\sigma(j) = i$, und $x_{ij} := 0$ sonst. Sei $\sigma^* : \mathcal{D} \rightarrow \mathcal{F}$ eine optimale Lösung für I und $X^* := \{i \in \mathcal{F} : \text{es gibt ein } j \in \mathcal{D} \text{ mit } \sigma^*(j) = i\}$ die Menge der mindestens einmal geöffneten Standorte. Dann gilt:

$$\begin{aligned} c_F(x) + c_S(x) &= \sum_{i \in X} \left\lceil \frac{|\{j \in \mathcal{D} : \sigma(j) = i\}|}{u_i} \right\rceil t_i + \sum_{j \in \mathcal{D}} c_{\sigma(j)} j \\ &\leq \sum_{i \in X} t_i + \sum_{j \in \mathcal{D}} c'_{\sigma(j)} j \\ &\leq \gamma_F \sum_{i \in X^*} t_i + \gamma_S \sum_{j \in \mathcal{D}} c'_{\sigma^*(j)} j \\ &\leq (\gamma_F + \gamma_S) \sum_{i \in X^*} \left\lceil \frac{|\{j \in \mathcal{D} : \sigma^*(j) = i\}|}{u_i} \right\rceil t_i + \gamma_S \sum_{j \in \mathcal{D}} c_{\sigma^*(j)} j. \end{aligned}$$

□

Korollar 22.23. Das METRISCHE SCHWACH-BESCHRÄNKTE STANDORTPROBLEM hat einen 2,89-Approximationsalgorithmus.

Beweis: Wende Satz 22.22 auf den DUAL-FITTING-ALGORITHMUS an (Korollar 22.7(c)); hier sind $\gamma_F = 1,11$ und $\gamma_S = 1,78$. □

Siehe Aufgabe 11 bezüglich einer besseren Approximationsgüte.

Haben wir es mit harten Kapazitäten zu tun (d. h. Standorte dürfen nicht mehrfach geöffnet werden), so müssen wir die Aufteilung der Kundennachfragen zulassen, d. h. die Zuteilung zu mehreren offenen Standorten: Erlauben wir dies nicht, so können wir kein Resultat erwarten, da sogar die Entscheidung, ob es überhaupt eine zulässige Lösung gibt, bereits NP-vollständig ist (dies enthält das PARTITION-Problem; siehe Korollar 15.28).

Der erste Approximationsalgorithmus für das METRISCHE BESCHRÄNKTE STANDORTPROBLEM stammt von Pál, Tardos und Wexler [2001], die damit ein früheres spezielleres Resultat von Korupolu, Plaxton und Rajaraman [2000] erweiterten. Die Approximationsgüte wurde dann von Zhang, Chen und Ye [2005] auf 5,83 verbessert. Für den Spezialfall, dass alle Standorte dieselben Bereitstellungs-kosten haben, haben Levi, Shmoys und Swamy [2012] einen 5-Approximations-algorithmus durch Rundung einer LP-Relaxierung gefunden. Für das allgemeine METRISCHE BESCHRÄNKTE STANDORTPROBLEM haben An, Singh and Svensson [2017] den ersten Approximationsalgorithmus mit konstanter Gütegarantie angegeben, der auf dem Runden einer LP-Relaxierung basiert.

Die Ergebnisse von Pál, Tardos und Wexler [2001] wurden von Mahdian und Pál [2003] auf das UNIVERSELLE STANDORTPROBLEM verallgemeinert. Sie haben einen 7,88-Approximationsalgorithmus entwickelt. Im nächsten Abschnitt besprechen wir einen auf lokaler Suche basierenden Algorithmus mit der Approximationsgüte 6,702 für das UNIVERSELLE STANDORTPROBLEM. Zunächst notieren wir das folgende Resultat.

Lemma 22.24. (Mahdian und Pál [2003]) *Jede Instanz des UNIVERSELLEN STANDORTPROBLEMS hat eine optimale Lösung.*

Beweis: Gibt es keine Lösung mit endlichen Kosten, so ist jede Lösung eine optimale. Andernfalls sei $(x^i)_{i \in \mathbb{N}}$ eine Folge von Lösungen, deren Kosten gegen das Infimum $c^* \in \mathbb{R}_+$ der Menge der Kosten von zulässigen Lösungen konvergiert. Da diese Folge beschränkt ist, gibt es eine Teilfolge $(x^{i_j})_{j \in \mathbb{N}}$, die gegen eine Lösung x^* konvergiert. Es ist x^* eine zulässige Lösung, und da alle f_i links-stetig und nicht fallend sind, folgt $c(x^*) = c(\lim_{j \rightarrow \infty} x^{i_j}) \leq \lim_{j \rightarrow \infty} c(x^{i_j}) = c^*$, d. h. x^* ist eine optimale Lösung. \square

22.8 Das universelle Standortproblem

In diesem auf Vygen [2007] basierenden Abschnitt besprechen wir einen auf lokaler Suche basierenden Algorithmus für das UNIVERSELLE STANDORTPROBLEM. Dieser benutzt zwei Operationen. Die erste Operation ist $\text{ADD}(t, \delta)$ mit $t \in \mathcal{F}$ und $\delta \in \mathbb{R}_+$, und besteht aus dem Ersetzen der aktuellen zulässigen Lösung x durch eine optimale Lösung y für das folgende Problem:

$$\min \left\{ c_S(y) : y_{ij} \geq 0 (i \in \mathcal{F}, j \in \mathcal{D}), \sum_{i \in \mathcal{F}} y_{ij} = d_j (j \in \mathcal{D}), \right. \\ \left. \sum_{j \in \mathcal{D}} y_{ij} \leq \sum_{j \in \mathcal{D}} x_{ij} (i \in \mathcal{F} \setminus \{t\}), \sum_{j \in \mathcal{D}} y_{tj} \leq \sum_{j \in \mathcal{D}} x_{tj} + \delta \right\}. \quad (22.15)$$

Mit $c^x(t, \delta) := c_S(y) - c_S(x) + f_t(\sum_{j \in \mathcal{D}} x_{tj} + \delta) - f_t(\sum_{j \in \mathcal{D}} x_{tj})$ bezeichnen wir die geschätzten Kosten dieser Operation; es ist $c^x(t, \delta)$ eine obere Schranke für $c(y) - c(x)$.

Lemma 22.25. (Mahdian und Pál [2003]) Sei $\epsilon > 0$. Sei x eine zulässige Lösung für eine gegebene Instanz und $t \in \mathcal{F}$. Dann gibt es einen Algorithmus mit $O(|V|^3 \log |V| \epsilon^{-1})$ -Laufzeit, der ein $\delta \in \mathbb{R}_+$ mit $c^x(t, \delta) \leq -\epsilon c(x)$ findet oder entscheidet, dass es kein $\delta \in \mathbb{R}_+$ mit $c^x(t, \delta) \leq -2\epsilon c(x)$ gibt.

Beweis: Wir können annehmen, dass $c(x) > 0$. Sei $C := \{v\epsilon c(x) : v \in \mathbb{Z}_+, v \leq \lceil \frac{1}{\epsilon} \rceil\}$. Sei δ_γ für jedes $\gamma \in C$ das größte $\delta \in \mathbb{R}_+$ mit $f_t(\sum_{j \in \mathcal{D}} x_{tj} + \delta) - f_t(\sum_{j \in \mathcal{D}} x_{tj}) \leq \gamma$. Wir werden $c^x(t, \delta_\gamma)$ für alle $\gamma \in C$ berechnen.

Angenommen, es gibt ein $\delta \in \mathbb{R}_+$ mit $c^x(t, \delta) \leq -2\epsilon c(x)$. Betrachte

$$\gamma := \epsilon c(x) \left[\frac{1}{\epsilon c(x)} \left(f_t \left(\sum_{j \in \mathcal{D}} x_{tj} + \delta \right) - f_t \left(\sum_{j \in \mathcal{D}} x_{tj} \right) \right) \right] \in C.$$

Beachte, dass $\delta_\gamma \geq \delta$. Somit gilt $c^x(t, \delta_\gamma) < c^x(t, \delta) + \epsilon c(x) \leq -\epsilon c(x)$.

Die Laufzeit wird durch die Lösung von $|C|$ Problemen vom Typ (22.15) dominiert. Damit folgt die Laufzeit nach Proposition 22.21. \square

Gibt es keine ausreichend gewinnbringende ADD-Operation, so können die Servicekosten beschränkt werden. Das folgende Resultat stammt im Wesentlichen von Pál, Tardos und Wexler [2001]:

Lemma 22.26. Sei $\epsilon > 0$. Seien x, x^* zulässige Lösungen für eine gegebene Instanz und sei $c^x(t, \delta) \geq -\frac{\epsilon}{|\mathcal{F}|} c(x)$ für alle $t \in \mathcal{F}$ und $\delta \in \mathbb{R}_+$. Dann gilt $c_S(x) \leq c_F(x^*) + c_S(x^*) + \epsilon c(x)$.

Beweis: Betrachte den (vollständigen bipartiten) Digraphen $G = (\mathcal{D} \cup \mathcal{F}, (\mathcal{D} \times \mathcal{F}) \cup (\mathcal{F} \times \mathcal{D}))$ mit den Kantengewichten $c((j, i)) := c_{ij}$ und $c((i, j)) := -c_{ij}$ für $i \in \mathcal{F}$ und $j \in \mathcal{D}$. Sei $b(i) := \sum_{j \in \mathcal{D}} (x_{ij} - x_{ij}^*)$ für $i \in \mathcal{F}$, $S := \{i \in \mathcal{F} : b(i) > 0\}$ und $T := \{i \in \mathcal{F} : b(i) < 0\}$.

Sei $g : E(G) \rightarrow \mathbb{R}_+$ der b -Fluss mit $g(i, j) := \max\{0, x_{ij} - x_{ij}^*\}$ und $g(j, i) := \max\{0, x_{ij}^* - x_{ij}\}$ für $i \in \mathcal{F}, j \in \mathcal{D}$.

Schreibe g als Summe von b_t -Flüssen g_t für $t \in T$, wobei $b_t(t) = b(t)$, $b_t(v) = 0$ für $v \in T \setminus \{t\}$ und $0 \leq b_t(v) \leq b(v)$ für $v \in V(G) \setminus T$. (Dies kann mit Standardmethoden der Flussdekomposition bewerkstelligt werden.)

Für jedes $t \in T$ definiert g_t eine zulässige Neuzuordnung der Kunden zu t , d. h. eine neue Lösung x^t gegeben durch $x_{ij}^t := x_{ij} + g_t(j, i) - g_t(i, j)$ für $i \in \mathcal{F}, j \in \mathcal{D}$. Es gilt $c_S(x^t) = c_S(x) + \sum_{e \in E(G)} c(e)g_t(e)$, also folgt

$$c^x(t, -b(t)) \leq \sum_{e \in E(G)} c(e)g_t(e) + f_t \left(\sum_{j \in \mathcal{D}} x_{tj}^* \right) - f_t \left(\sum_{j \in \mathcal{D}} x_{tj} \right).$$

Ist die linke Seite mindestens gleich $-\frac{\epsilon}{|\mathcal{F}|} c(x)$ für jedes $t \in T$, so bekommen wir mittels Summierung:

$$\begin{aligned}
-\epsilon c(x) &\leq \sum_{e \in E(G)} c(e)g(e) + \sum_{t \in T} f_t \left(\sum_{j \in \mathcal{D}} x_{tj}^* \right) \\
&\leq \sum_{e \in E(G)} c(e)g(e) + c_F(x^*) \\
&= c_S(x^*) - c_S(x) + c_F(x^*).
\end{aligned}$$

□

Nun wenden wir uns der zweiten Operation zu. Sei x eine zulässige Lösung einer gegebenen Instanz des UNIVERSELLEN STANDORTPROBLEMS. Sei A eine Arboreszenz mit $V(A) \subseteq \mathcal{F}$ und $\delta \in \Delta_A^x := \{\delta \in \mathbb{R}^{V(A)} : \sum_{j \in \mathcal{D}} x_{ij} + \delta_i \geq 0 \text{ für alle } i \in V(A), \sum_{i \in V(A)} \delta_i = 0\}$.

Die zweite Operation ist $\text{PIVOT}(A, \delta)$ und besteht aus dem Ersetzen von x durch eine Lösung x' mit $\sum_{j \in \mathcal{D}} x'_{ij} = \sum_{j \in \mathcal{D}} x_{ij} + \delta_i$ für $i \in V(A)$, $\sum_{j \in \mathcal{D}} x'_{ij} = \sum_{j \in \mathcal{D}} x_{ij}$ für $i \in \mathcal{F} \setminus V(A)$ und $c(x') \leq c(x) + c^x(A, \delta)$, wobei $c^x(A, \delta) := \sum_{i \in V(A)} c_{A,i}^x(\delta)$ und

$$c_{A,i}^x(\delta) := f_i \left(\sum_{j \in \mathcal{D}} x_{ij} + \delta_i \right) - f_i \left(\sum_{j \in \mathcal{D}} x_{ij} \right) + \left| \sum_{l \in A_i^+} \delta_l \right| c_{ip(i)}$$

für $i \in V(A)$. Hier bezeichnet A_i^+ die Menge der von i aus in A erreichbaren Knoten, und $p(i)$ ist der Vorgänger von i in A (bzw. ist beliebig, wenn i die Wurzel ist). Ein solches x' kann leicht konstruiert werden, indem man Nachfrage entlang den Kanten in A in umgekehrter topologischer Reihenfolge bewegt. Beachte, dass die Orientierung von A für $c^x(A, \delta)$ unbedeutend ist und nur der Vereinfachung der Notation dient.

Die Operation wird verwendet, wenn ihre *geschätzten Kosten* $c^x(A, \delta)$ ausreichend negativ sind. Damit wird gewährleistet, dass der resultierende auf lokaler Suche basierende Algorithmus nach einer polynomiellen Anzahl von Verbesserungsschritten terminiert. Wir nennen $\sum_{i \in V(A)} \left| \sum_{l \in A_i^+} \delta_l \right| c_{ip(i)}$ die *geschätzten Routing-Kosten* von $\text{PIVOT}(A, \delta)$.

Wir zeigen nun, wie wir eine verbessерnde PIVOT-Operation finden, sofern wir nicht bereits an einem approximativen lokalen Optimum angelangt sind:

Lemma 22.27. (Vygen [2007]) *Sei $\epsilon > 0$. Sei A eine Arboreszenz mit $V(A) \subseteq \mathcal{F}$ und x eine zulässige Lösung. Dann gibt es einen Algorithmus mit $O(|\mathcal{F}|^4 \epsilon^{-3})$ -Laufzeit, der ein $\delta \in \Delta_A^x$ mit $c^x(A, \delta) \leq -\epsilon c(x)$ findet oder entscheidet, dass es kein $\delta \in \Delta_A^x$ mit $c^x(A, \delta) \leq -2\epsilon c(x)$ gibt.*

Beweis: Nummeriere $V(A) = \{1, \dots, n\}$ in umgekehrter topologischer Reihenfolge, d.h. es gilt $i > j$ für alle $(i, j) \in E(A)$. Setze $B(k) := \{i < k : (p(k), i) \in E(A)\}$ für $k \in V(A)$ mit $(p(k), k) \in E(A)$; dies ist die Menge der jüngeren Geschwister von k . Setze $B(k) := \emptyset$, falls k die Wurzel von A ist. Setze ferner $I_k := \bigcup_{l \in B(k) \cup \{k\}} A_l^+, b(k) := \max(\{0\} \cup B(k))$ und $s(k) := \max(\{0\} \cup (A_k^+ \setminus \{k\}))$.

Setze $C := \{v \frac{\epsilon}{n} c(x) : v \in \mathbb{Z}, -\lceil \frac{n}{\epsilon} \rceil - n \leq v \leq \lceil \frac{n}{\epsilon} \rceil + n\}$. Wir berechnen die folgendermaßen definierte Tabelle $(T_A^x(k, \gamma))_{k \in \{0, \dots, n\}, \gamma \in C}$: Es ist $T_A^x(0, 0) := 0$, $T_A^x(0, \gamma) := \emptyset$ für alle $\gamma \in C \setminus \{0\}$, und für $k = 1, \dots, n$ sei $T_A^x(k, \gamma)$ eine optimale Lösung $\delta \in \mathbb{R}^{I_k}$ von

$$\max \left\{ \sum_{i \in I_k} \delta_i : \gamma' \in C, T_A^x(b(k), \gamma') \neq \emptyset, \delta_i = (T_A^x(b(k), \gamma'))_i \text{ für } i \in \bigcup_{l \in B(k)} A_l^+, \right.$$

$$\gamma'' \in C, T_A^x(s(k), \gamma'') \neq \emptyset, \delta_i = (T_A^x(s(k), \gamma''))_i \text{ für } i \in A_k^+ \setminus \{k\},$$

$$\left. \sum_{j \in \mathcal{D}} x_{kj} + \delta_k \geq 0, \gamma' + \gamma'' + c_{A,k}^x(\delta) \leq \gamma \right\},$$

falls die Menge, über der das Maximum genommen wird, nicht leer ist, und $T_A^x(k, \gamma) := \emptyset$ sonst.

Grob ausgedrückt ist $-\sum_{i \in I_k} (T_A^x(k, \gamma))_i$ der minimale Überschuss, den wir beim Vorgänger $p(k)$ von k bei der Nachfrageverschiebung von jedem Knoten in I_k zum jeweiligen Vorgänger oder auch umgekehrt bekommen, bei gerundeten geschätzten Gesamtkosten von höchstens γ .

Beachte, dass $T_A^x(k, 0) \neq \emptyset$ für $k = 0, \dots, n$. Damit können wir das Minimum $\gamma \in C$ so wählen, dass $T_A^x(n, \gamma) \neq \emptyset$ und $\sum_{i=1}^n (T_A^x(n, \gamma))_i \geq 0$. Dann wählen wir $\delta \in \Delta_A^x$ mit $\delta_i = (T_A^x(n, \gamma))_i$ oder $0 \leq \delta_i \leq (T_A^x(n, \gamma))_i$ für alle $i = 1, \dots, n$ und $|\sum_{l \in A_i^+} \delta_l| \leq |\sum_{l \in A_i^+} (T_A^x(n, \gamma))_l|$ für alle $i = 1, \dots, n$. Um dies zu erreichen, setzen wir $\delta := T_A^x(n, \gamma)$ und verringern δ_i schrittweise, wobei i maximal bezüglich der folgenden Eigenschaft ist: Es ist $\delta_i > 0$ und $\sum_{l \in A_k^+} \delta_l > 0$ für alle Knoten k auf dem Weg von n nach i in A . Beachte, dass die Eigenschaft $c^x(A, \delta) \leq \gamma$ erhalten bleibt. Es bleibt zu zeigen, dass γ genügend klein ist.

Angenommen, es gibt eine Operation PIVOT(A, δ) mit $c^x(A, \delta) \leq -2\epsilon c(x)$. Da $c_{A,i}^x(\delta) \geq -f_i(\sum_{j \in \mathcal{D}} x_{ij}) \geq -c(x)$ für alle $i \in V(A)$, folgt hieraus auch, dass $c_{A,i}^x(\delta) < c_F(x) \leq c(x)$. Somit gilt $\gamma_i := \lceil c_{A,i}^x(\delta) \frac{n}{\epsilon c(x)} \rceil \frac{\epsilon c(x)}{n} \in C$ für $i = 1, \dots, n$ und $\sum_{i \in I} \gamma_i \in C$ für alle $I \subseteq \{1, \dots, n\}$. Dann folgt mittels eines leichten Induktionsbeweises, dass $\sum_{i \in I_k} (T_A^x(k, \sum_{l \in I_k} \gamma_l))_i \geq \sum_{i \in I_k} \delta_i$ für $k = 1, \dots, n$. Demnach erhalten wir eine PIVOT-Operation mit geschätzten Kosten von höchstens $\sum_{i=1}^n \gamma_i < c^x(A, \delta) + \epsilon c(x) \leq -\epsilon c(x)$.

Die Laufzeit kann wie folgt geschätzt werden. Wir müssen $n|C|$ Tabellenelemente berechnen, und für jedes Element $T_A^x(k, \gamma)$ probieren wir alle Werte von γ' und $\gamma'' \in C$ durch. Damit erhalten wir Werte δ_i für $i \in I_k \setminus \{k\}$, und der Hauptschritt ist die Berechnung des maximalen δ_k mit $\gamma' + \gamma'' + c_{A,k}^x(\delta) \leq \gamma$. Dies kann direkt mit dem Orakel, welches wir für die Funktionen $f_i, i \in \mathcal{F}$, vorausgesetzt haben, erledigt werden. Die abschließende Berechnung von δ aus den $T_A^x(n, \gamma), \gamma \in C$, ist leicht in linearer Zeit zu bewerkstelligen. Somit haben wir eine $O(n|C|^3) = O(|\mathcal{F}|^4 \epsilon^{-3})$ -Gesamtlaufzeit. \square

Nun betrachten wir die Operation PIVOT(A, δ) für besondere Arboreszenzen, nämlich Sterne und Kometen. Eine Arboreszenz A heißt **Stern mit Mittelpunkt**

v , falls $A = (\mathcal{F}, \{(v, w) : w \in \mathcal{F} \setminus \{v\}\})$ ist, und **Komet mit Mittelpunkt v und Schweif** (t, s) , falls v, t, s paarweise verschiedene Elemente aus \mathcal{F} sind und $A = (\mathcal{F}, \{(t, s)\} \cup \{(v, w) : w \in \mathcal{F} \setminus \{v, s\}\})$ ist. Beachte, dass es insgesamt weniger als $|\mathcal{F}|^3$ Sterne und Kometen gibt.

Wir werden nun zeigen, dass ein (approximatives) lokales Optimum niedrige Bereitstellungskosten hat.

Lemma 22.28. Seien x, x^* zulässige Lösungen für eine gegebene Instanz. Sei $c^x(A, \delta) \geq -\frac{\epsilon}{|\mathcal{F}|}c(x)$ für alle Sterne und Kometen A und $\delta \in \Delta_A^x$. Dann gilt $c_F(x) \leq 4c_F(x^*) + 2c_S(x^*) + 2c_S(x) + \epsilon c(x)$.

Beweis: Wir bedienen uns der Notation von Lemma 22.26 und betrachten die folgende Instanz des HITCHCOCK-PROBLEMS:

$$\begin{aligned} \min \quad & \sum_{s \in S, t \in T} c_{st} y(s, t) \\ \text{bzgl.} \quad & \sum_{t \in T} y(s, t) = b(s) \quad (s \in S) \\ & \sum_{s \in S} y(s, t) = -b(t) \quad (t \in T) \\ & y(s, t) \geq 0 \quad (s \in S, t \in T) \end{aligned} \tag{22.16}$$

Nach Proposition 9.22 gibt es eine optimale Lösung $y : S \times T \rightarrow \mathbb{R}_+$ des LP (22.16), so dass $F := (S \cup T, \{(s, t) : y(s, t) > 0\})$ ein Wald ist.

Da $(b_t(s))_{s \in S, t \in T}$ eine zulässige Lösung von (22.16) ist, haben wir

$$\begin{aligned} \sum_{s \in S, t \in T} c_{st} y(s, t) & \leq \sum_{s \in S, t \in T} c_{st} b_t(s) \\ & = \sum_{s \in S, t \in T} c_{st} (g_t(\delta^+(s)) - g_t(\delta^-(s))) \\ & \leq \sum_{e \in E(G)} |c(e)| g(e) \\ & \leq c_S(x^*) + c_S(x). \end{aligned} \tag{22.17}$$

Nun werden wir höchstens $|\mathcal{F}|$ PIVOT-Operationen definieren. Wir sagen: Eine Operation PIVOT(A, δ) schließt $s \in S$ (bezüglich x und x^*), falls $\sum_{j \in \mathcal{D}} x_{sj} > \sum_{j \in \mathcal{D}} x_{sj} + \delta_s = \sum_{j \in \mathcal{D}} x_{sj}^*$. Ferner sagen wir: Die obige Operation öffnet $t \in T$, falls $\sum_{j \in \mathcal{D}} x_{tj} < \sum_{j \in \mathcal{D}} x_{tj} + \delta_t \leq \sum_{j \in \mathcal{D}} x_{tj}^*$. Beziehlich der Gesamtheit der noch zu definierenden Operationen gilt: Jedes $s \in S$ wird genau einmal geschlossen und jedes $t \in T$ wird höchstens viermal geöffnet. Ferner werden die gesamten geschätzten Routing-Kosten höchstens gleich $2 \sum_{s \in S, t \in T} c_{st} y(s, t)$ sein. Demnach werden die geschätzten Gesamtkosten der Operationen höchstens $4c_F(x^*) + 2c_S(x^*) + 2c_S(x) - c_F(x)$ betragen. Damit wird das Lemma bewiesen sein.

Zur Definition der Operationen orientieren wir F als Branching B mit der Eigenschaft, dass jede Zusammenhangskomponente ein Element aus T als Wurzel

hat. Wir setzen $y(e) := y(s, t)$, falls $e \in E(B)$ die Endknoten $s \in S$ und $t \in T$ hat. Ein Knoten $v \in V(B)$ heißt *schwach*, falls $y(\delta_B^+(v)) > y(\delta_B^-(v))$, sonst *stark*. Mit $\Gamma_s^+(v)$, $\Gamma_w^+(v)$ und $\Gamma^+(v)$ ($s=\text{strong}$, $w=\text{weak}$) bezeichnen wir die Mengen der starken Kinder, der schwachen Kinder bzw. aller Kinder von $v \in V(B)$ in B .

Sei $t \in T$ und $\Gamma_w^+(t) = \{w_1, \dots, w_k\}$ die Menge der schwachen Kinder von t , die so geordnet sind, dass $r(w_1) \leq \dots \leq r(w_k)$, wobei $r(w_i) := \max\{0, y(w_i, t) - \sum_{t' \in \Gamma_w^+(w_i)} y(w_i, t')\}$. Sei ferner $\Gamma_s^+(t) = \{s_1, \dots, s_l\}$ so geordnet, dass $y(s_1, t) \geq \dots \geq y(s_l, t)$.

Zunächst nehmen wir an, dass $k > 0$. Für jedes $i = 1, \dots, k-1$ betrachten wir eine PIVOT-Operation mit dem in w_i zentrierten Stern und den folgenden Routings:

- höchstens $2y(w_i, t')$ Nachfrageeinheiten von w_i zu jedem schwachen Kind t' von w_i ,
- genau $y(w_i, t')$ Nachfrageeinheiten von w_i zu jedem starken Kind t' von w_i und
- genau $r(w_i)$ Nachfrageeinheiten von w_i nach $\Gamma_s^+(w_{i+1})$,

und auch noch der Schließung von w_i und der Öffnung einer Teilmenge von $\Gamma^+(w_i) \cup \Gamma_s^+(w_{i+1})$. Die geschätzten Routing-Kosten betragen höchstens

$$\begin{aligned} & \sum_{t' \in \Gamma_w^+(w_i)} c_{w_i t'} 2y(w_i, t') + \sum_{t' \in \Gamma_s^+(w_i)} c_{w_i t'} y(w_i, t') + c_{tw_i} r(w_i) \\ & + c_{tw_{i+1}} r(w_{i+1}) + \sum_{t' \in \Gamma_s^+(w_{i+1})} c_{w_{i+1} t'} y(w_{i+1}, t'), \end{aligned}$$

$$\text{da } r(w_i) \leq r(w_{i+1}) \leq \sum_{t' \in \Gamma_s^+(w_{i+1})} y(w_{i+1}, t').$$

Um weitere PIVOT-Operationen bezüglich t zu definieren, unterscheiden wir drei Fälle.

Fall 1: Es ist t stark oder $l = 0$. Betrachte:

- (1) eine PIVOT-Operation mit dem in w_k zentrierten Stern und den folgenden Routings:
 - genau $y(w_k, t')$ Nachfrageeinheiten von w_k zu jedem Kind t' von w_k und
 - genau $y(w_k, t)$ Nachfrageeinheiten von w_k nach t ,
 und der Schließung von w_k und Öffnung von t und der Kinder von w_k , und
- (2) eine PIVOT-Operation mit dem in t zentrierten Stern und dem folgenden Routing:
 - höchstens $2y(s, t)$ Nachfrageeinheiten von jedem starken Kind s von t nach t ,
 und der Schließung der starken Kinder von t und Öffnung von t .

(Im Falle $l = 0$ kann die zweite PIVOT-Operation weggelassen werden.)

Fall 2: Es ist t schwach, $l \geq 1$ und $y(w_k, t) + y(s_1, t) \geq \sum_{i=2}^l y(s_i, t)$. Betrachte:

- (1) eine PIVOT-Operation mit dem in w_k zentrierten Stern und den folgenden Routings:

- genau $y(w_k, t')$ Nachfrageeinheiten von w_k zu jedem Kind t' von w_k und
 - genau $y(w_k, t)$ Nachfrageeinheiten von w_k nach t ,
- und der Schließung von w_k , der Öffnung der Kinder von w_k und der Öffnung von t ,
- (2) eine PIVOT-Operation mit dem in s_1 zentrierten Stern und den folgenden Routings:
- genau $y(s_1, t')$ Nachfrageeinheiten von s_1 zu jedem Kind t' von s_1 und
 - genau $y(s_1, t)$ Nachfrageeinheiten von s_1 nach t ,
- und der Schließung von s_1 , der Öffnung der Kinder von s_1 und der Öffnung von t , und
- (3) eine PIVOT-Operation mit dem in t zentrierten Stern und dem folgenden Routing:
- höchstens $2y(s_i, t)$ Nachfrageeinheiten von s_i nach t für $i = 2, \dots, l$,
und der Schließung von s_2, \dots, s_l und Öffnung von t .

Fall 3: Es ist t schwach, $l \geq 1$ und $y(w_k, t) + y(s_1, t) < \sum_{i=2}^l y(s_i, t)$. Betrachte:

- (1) eine PIVOT-Operation mit dem Komet mit Mittelpunkt w_k und Schweif (t, s_1) und den folgenden Routings:
- genau $y(w_k, t')$ Nachfrageeinheiten von w_k zu jedem Kind t' von w_k ,
 - genau $y(w_k, t)$ Nachfrageeinheiten von w_k nach t und
 - höchstens $2y(s_1, t)$ Nachfrageeinheiten von s_1 nach t ,
- und der Schließung von w_k und s_1 und Öffnung von t und der Kinder von w_k ,
- (2) eine PIVOT-Operation mit dem in t zentrierten Stern und dem folgenden Routing:
- höchstens $2y(s_i, t)$ Nachfrageeinheiten von s_i nach t für jedes ungerade Element i von $\{2, \dots, l\}$,
und der Schließung der ungeraden Elemente von $\{s_2, \dots, s_l\}$ und Öffnung von t , und
- (3) eine PIVOT-Operation mit dem in t zentrierten Stern und dem folgenden Routing:
- höchstens $2y(s_i, t)$ Nachfrageeinheiten von s_i nach t für jedes gerade Element i von $\{2, \dots, l\}$,
und der Schließung der geraden Elemente von $\{s_2, \dots, s_l\}$ und Öffnung von t .

Ist $k = 0$, so betrachten wir dieselben PIVOT-Operationen mit den folgenden Abweichungen: In den Fällen 1 und 2 (wo $y(w_0, t) := 0$) wird (1) weggelassen und in Fall 3 wird (1) ersetzt durch die PIVOT-Operation mit dem in t zentrierten Stern und dem Routing von höchstens $2y(s_1, t)$ Nachfrageeinheiten von s_1 nach t , der Schließung von s_1 und der Öffnung von t .

Wir sammeln all diese PIVOT-Operationen für alle $t \in T$ zusammen. Insgesamt haben wir dann jedes $s \in S$ genau einmal geschlossen und jedes $t \in T$ höchstens viermal geöffnet, mit gesamten geschätzten Routing-Kosten von höchstens $2 \sum_{\{s,t\} \in E(F)} c_{st} y(s, t)$, die nach (22.17) höchstens gleich $2c_S(x^*) + 2c_S(x)$ sind.

Sind die geschätzten Kosten einer jeden Operation größer oder gleich $-\frac{\epsilon}{|\mathcal{F}|}c(x)$, so folgt $-\epsilon c(x) \leq -c_F(x) + 4c_F(x^*) + 2c_S(x^*) + 2c_S(x)$, wie gewünscht. \square

Aus den vorangehenden Resultaten folgern wir:

Satz 22.29. *Sei $0 < \epsilon \leq 1$. Seien x, x^* zulässige Lösungen für eine gegebene Instanz und sei $c^x(t, \delta) > -\frac{\epsilon}{8|\mathcal{F}|}c(x)$ für $t \in \mathcal{F}$, $\delta \in \mathbb{R}_+$ und $c^x(A, \delta) > -\frac{\epsilon}{8|\mathcal{F}|}c(x)$ für alle Sterne und Kometen A und $\delta \in \Delta_A^x$. Dann gilt $c(x) \leq (1 + \epsilon)(7c_F(x^*) + 5c_S(x^*))$.*

Beweis: Nach Lemma 22.26 gilt $c_S(x) \leq c_F(x^*) + c_S(x^*) + \frac{\epsilon}{8}c(x)$, und nach Lemma 22.28 gilt $c_F(x) \leq 4c_F(x^*) + 2c_S(x^*) + 2c_S(x) + \frac{\epsilon}{8}c(x)$. Damit haben wir $c(x) = c_F(x) + c_S(x) \leq 7c_F(x^*) + 5c_S(x^*) + \frac{\epsilon}{2}c(x)$, woraus $c(x) \leq (1 + \epsilon)(7c_F(x^*) + 5c_S(x^*))$ folgt. \square

Schließlich wenden wir noch eine Standard-Skalierungsmethode an, um das Hauptresultat dieses Abschnitts zu erhalten:

Satz 22.30. (Vygen [2007]) *Für jedes $\epsilon > 0$ gibt es einen polynomiellen $(\frac{\sqrt{41}+7}{2} + \epsilon)$ -Approximationsalgorithmus für das UNIVERSELLE STANDORTPROBLEM.*

Beweis: Wir können annehmen, dass $\epsilon \leq \frac{1}{3}$. Sei $\beta := \frac{\sqrt{41}-5}{2} \approx 0,7016$. Setze $f'_i(z) := \beta f_i(z)$ für alle $z \in \mathbb{R}_+$ und $i \in \mathcal{F}$ und betrachte diese veränderte Instanz.

Sei x irgendeine anfängliche zulässige Lösung. Wende die Algorithmen von Lemma 22.25 und Lemma 22.27 mit $\frac{\epsilon}{16|\mathcal{F}|}$ statt ϵ an. Entweder finden diese Algorithmen eine ADD- oder eine PIVOT-Operation, welche die Kosten der aktuellen Lösung x um mindestens $\frac{\epsilon}{16|\mathcal{F}|}c(x)$ verringert, oder sie bestätigen, dass die Voraussetzungen von Satz 22.29 erfüllt sind.

Ist x die resultierende Lösung, bezeichnen c'_F und c_F die Bereitstellungskosten der veränderten bzw. ursprünglichen Instanz und ist x^* irgendeine zulässige Lösung, so gilt $c_F(x) + c_S(x) =$

$$\frac{1}{\beta}c'_F(x) + c_S(x) \leq \frac{1}{\beta}(6c'_F(x^*) + 4c_S(x^*) + \frac{3\epsilon}{8}c(x)) + c'_F(x^*) + c_S(x^*) + \frac{\epsilon}{8}c(x) \leq (6 + \beta)c_F(x^*) + (1 + \frac{4}{\beta})c_S(x^*) + \frac{3\epsilon}{4}c(x) = (6 + \beta)(c_F(x^*) + c_S(x^*)) + \frac{3\epsilon}{4}c(x).$$

Damit folgt $c(x) \leq (1 + \epsilon)(6 + \beta)c(x^*)$.

Jede Iteration verringert die Kosten um mindestens den Faktor $\frac{1}{1 - \frac{\epsilon}{16|\mathcal{F}|}}$, also werden die Kosten nach $\frac{1}{-\log(1 - \frac{\epsilon}{16|\mathcal{F}|})} < \frac{16|\mathcal{F}|}{\epsilon}$ Iterationen um mindestens den Faktor 2 verringert (Beachte, dass $\log x < x - 1$ für $0 < x < 1$). Dies bedeutet eine schwach polynomielle Laufzeit. \square

Da $\frac{\sqrt{41}+7}{2} < 6,702$, haben wir insbesondere einen 6,702-Approximationsalgorithmus. Dies ist die momentan beste bekannte Approximationsgüte.

Aufgaben

- Man zeige, dass es für das k -MEDIAN-PROBLEM (ohne metrische Kosten vorzusetzen) keinen Approximationsalgorithmus mit konstanter Gütegarantie gibt, sofern $P \neq NP$.
- Man betrachte eine Instanz des UNBESCHRÄNKEN STANDORTPROBLEMS und beweise, dass $c_S : 2^{\mathcal{F}} \rightarrow \mathbb{R}_+ \cup \{\infty\}$ supermodular ist, wobei $c_S(X) := \sum_{j \in \mathcal{D}} \min_{i \in X} c_{ij}$.
- Man betrachte eine andere Formulierung des UNBESCHRÄNKEN STANDORTPROBLEMS als ganzzahliges LP, mit einer 0/1-Variable z_S für jedes Paar $S \in \mathcal{F} \times 2^{\mathcal{D}}$:

$$\begin{aligned} \min \quad & \sum_{S=(i,D) \in \mathcal{F} \times 2^{\mathcal{D}}} \left(f_i + \sum_{j \in D} c_{ij} \right) z_S \\ \text{bzgl.} \quad & \sum_{S=(i,D) \in \mathcal{F} \times 2^{\mathcal{D}}: j \in D} z_S \geq 1 \quad (j \in \mathcal{D}) \\ z_S \in & \{0, 1\} \quad (S \in \mathcal{F} \times 2^{\mathcal{D}}). \end{aligned}$$

Man betrachte die natürliche LP-Relaxierung und das duale LP. Man zeige, wie man diese in polynomieller Zeit lösen kann (trotz ihrer exponentiellen Größe). Man zeige, dass der optimale Zielfunktionswert gleich dem von (22.2) und von (22.3) ist.

- Man betrachte die LP-Relaxierung eines einfachen Spezialfalls des METRISCHEN BESCHRÄNKEN STANDORTPROBLEMS, in welchem jeder Standort bis zu u Kunden ($u \in \mathbb{N}$) versorgen kann: Dieses LP erhält man, indem man (22.2) durch Hinzufügen der Nebenbedingungen $y_i \leq 1$ und $\sum_{j \in \mathcal{D}} x_{ij} \leq uy_i$ für $i \in \mathcal{F}$ erweitert.

Man zeige, dass diese Klasse von LPs eine unbeschränkte Ganzzahligkeitslücke hat, d. h. das Verhältnis der Kosten einer optimalen ganzzahligen Lösung zum optimalen Zielfunktionswert des LP kann beliebig groß sein.

(Shmoys, Tardos und Aardal [1997])

- Man betrachte das UNBESCHRÄNKTE STANDORTPROBLEM mit der Eigenschaft, dass jedem Kunden $j \in \mathcal{D}$ eine Nachfrage $d_j > 0$ entspricht und die Servicekosten pro Nachfrageeinheit metrisch sind, d. h. $\frac{c_{ij}}{d_j} + \frac{c_{i'j}}{d_j} + \frac{c_{i'j'}}{d_{j'}} \geq \frac{c_{ij'}}{d_{j'}}$ für $i, i' \in \mathcal{F}$ und $j, j' \in \mathcal{D}$. Man ändere die Approximationsalgorithmen für den Fall, dass alle Nachfragen gleich 1 sind und zeige, dass man dieselben Approximationsgüten im obigen allgemeineren Fall bekommen kann.
- Man zeige, dass (22.7) tatsächlich äquivalent als LP formuliert werden kann.
- Man betrachte das LP (22.7) mit $\gamma_F = 1$ und zeige, dass das Supremum der Optima für alle $d \in \mathbb{N}$ gleich 2 ist.
(Jain et al. [2003])
- Man betrachte eine Instanz des METRISCHEN UNBESCHRÄNKEN STANDORTPROBLEMS. Hier sei das Ziel, eine Menge $X \subseteq \mathcal{F}$ mit $\sum_{i \in X} f_i +$

$\sum_{j \in \mathcal{D}} \min_{i \in X} c_{ij}^2$ minimal zu finden. Man beschreibe einen k -Approximationsalgorithmus für dieses Problem und versuche, eine Approximationsgüte unter 3 zu erreichen.

9. Man kombiniere die Sätze 22.3 und 22.10 um zu zeigen, dass der JAIN-VAZIRANI-ALGORITHMUS zusammen mit Skalierung und Greedy-Augmentierung eine Approximationsgüte von 1,853 hat.

- * 10. Das MAX- k -COVER-PROBLEM wird wie folgt definiert. Für ein gegebenes Mengensystem (U, \mathcal{F}) und eine natürliche Zahl k finde man eine Teilmenge $S \subseteq \mathcal{F}$ mit $|S| = k$ und $|\bigcup S|$ maximal. Man beweise, dass der natürliche Greedy-Algorithmus (man wählt in jedem Schritt eine Menge, die so viele neue Elemente wie möglich überdeckt) ein $(\frac{e}{e-1})$ -Approximationsalgorithmus für das MAX- k -COVER-PROBLEM ist.
- 11. Man zeige, dass es einen 2-Approximationsalgorithmus für das METRISCHE SCHWACH-BESCHRÄNKTE STANDORTPROBLEM gibt.
Hinweis: Man kombiniere den Beweis von Satz 22.22 mit der Analyse des DUAL-FITTING-ALGORITHMUS; hier können die Ungleichungen (22.6) verschärft werden.
(Mahdian, Ye und Zhang [2006])
- 12. Man kombiniere lokale Suche (Satz 22.20) mit Kostendiskretisierung (Lemma 22.15), Skalierung und Greedy-Augmentierung (Satz 22.10), um einen 2,375-Approximationsalgorithmus für das METRISCHE UNBESCHRÄNKTE STANDORTPROBLEM zu erhalten.
- 13. Man betrachte den Spezialfall des UNIVERSELLEN STANDORTPROBLEMS, wenn die Kostenfunktionen f_i für alle $i \in \mathcal{F}$ linear sind, und beschreibe einen 3-Approximationsalgorithmus für diesen Fall.
- 14. Seien $\alpha_0, \alpha_1, \dots, \alpha_r \in \mathbb{R}_+$ mit $\alpha_1 = \max_{i=1}^r \alpha_i$ und $S := \sum_{i=0}^r \alpha_i$. Man zeige, dass es eine Partition $\{2, \dots, r\} = I_0 \cup I_1$ mit $\alpha_k + \sum_{i \in I_k} 2\alpha_i \leq S$ für $k = 0, 1$ gibt.
Hinweis: Man sortiere die Liste und wähle jedes zweite Element.
- * 15. Man betrachte einen auf lokaler Suche basierenden Algorithmus für das METRISCHE BESCHRÄNKTE STANDORTPROBLEM, der zusätzlich zu dem Algorithmus in Abschnitt 22.8 eine weitere Operation hat, nämlich eine PIVOT-Operation auf Wäldern, die die disjunkte Vereinigung zweier Sterne sind. Es kann bewiesen werden, dass diese Operation in diesem Spezialfall in polynomieller Zeit implementiert werden kann. Man zeige, dass man mit dieser zusätzlichen Operation eine Approximationsgüte von 5,83 erreichen kann.
Hinweis: Man ändere den Beweis von Lemma 22.28 mittels dieser neuen Operation und verwende Aufgabe 14.
(Zhang, Chen und Ye [2005])
- 16. Man betrachte den als SOFT-CAPACITATED STANDORTPROBLEM bekannten Spezialfall, in welchem für jedes i die Kapazität u_i des Standortes i gleich 1 oder 2 ist. Man zeige, dass dieses Problem in polynomieller Zeit optimal gelöst werden kann.

Literatur

Allgemeine Literatur:

- Cornuéjols, G., Nemhauser, G.L., und Wolsey, L.A. [1990]: The uncapacitated facility location problem. In: Discrete Location Theory (P.B. Mirchandani, R.L. Francis, Hrsg.), Wiley, New York 1990, pp. 119–171
- Shmoys, D.B. [2000]: Approximation algorithms for facility location problems. Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization; LNCS 1913 (K. Jansen, S. Khuller, Hrsg.), Springer, Berlin 2000, pp. 27–33
- Vygen, J. [2005]: Approximation algorithms for facility location problems (lecture notes). Report No. 05950-OR, Research Institute for Discrete Mathematics, University of Bonn, 2005

Zitierte Literatur:

- An, H.-C., Singh, M., und Svensson, O. [2017]: LP-based algorithms for capacitated facility location. SIAM Journal on Computing 46 (2017), 272–306
- Archer, A., Rajagopalan, R., und Shmoys, D.B. [2003]: Lagrangian relaxation for the k -median problem: new insights and continuity properties. In: Algorithms – Proceedings of the 11th European Symposium on Algorithms (ESA); LNCS 2832 (G. di Battista, U. Zwick, Hrsg.), Springer, Berlin 2003, pp. 31–42
- Arora, S., Raghavan, P., und Rao, S. [1998]: Approximation schemes for Euclidean k -medians and related problems. Proceedings of the 30th Annual ACM Symposium on Theory of Computing (1998), 106–113
- Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., und Pandit, V. [2004]: Local search heuristics for k -median and facility location problems. SIAM Journal on Computing 33 (2004), 544–562
- Balinski, M.L. [1965]: Integer programming: methods, uses, computation. Management Science 12 (1965), 253–313
- Balinski, M.L., und Wolfe, P. [1963]: On Benders decomposition and a plant location problem. Working paper ARO-27. Mathematica, Princeton 1963
- Byrka, J., und Aardal, K. [2007]: The approximation gap for the metric facility location problem is not yet closed. Operations Research Letters 35 (2007), 379–384
- Byrka, J., und Aardal, K. [2010]: An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. SIAM Journal on Computing 39 (2010), 2212–2231
- Charikar, M., und Guha, S. [2005]: Improved combinatorial algorithms for the facility location and k -median problems. SIAM Journal on Computing 34 (2005), 803–824
- Charikar, M., Guha, S., Tardos, É., und Shmoys, D.B. [2002]: A constant-factor approximation algorithm for the k -median problem. Journal of Computer and System Sciences 65 (2002), 129–149
- Chudak, F.A., und Shmoys, D.B. [2003]: Improved approximation algorithms for the uncapacitated facility location problem. SIAM Journal on Computing 33 (2003), 1–25
- Feige, U. [1998]: A threshold of $\ln n$ for the approximating set cover. Journal of the ACM 45 (1998), 634–652
- Guha, S., und Khuller, S. [1999]: Greedy strikes back: improved facility location algorithms. Journal of Algorithms 31 (1999), 228–248

- Hochbaum, D.S. [1982]: Heuristics for the fixed cost median problem. *Mathematical Programming* 22 (1982), 148–162
- Jain, K., Mahdian, M., Markakis, E., Saberi, A., und Vazirani, V.V. [2003]: Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM* 50 (2003), 795–824
- Jain, K., und Vazirani, V.V. [2001]: Approximation algorithms for metric facility location and k -median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM* 48 (2001), 274–296
- Kolliopoulos, S.G., und Rao, S. [2007]: A nearly linear-time approximation scheme for the Euclidean k -median problem. *SIAM Journal on Computing* 37 (2007), 757–782
- Korupolu, M., Plaxton, C., und Rajaraman, R. [2000]: Analysis of a local search heuristic for facility location problems. *Journal of Algorithms* 37 (2000), 146–188
- Kuehn, A.A., und Hamburger, M.J. [1963]: A heuristic program for locating warehouses. *Management Science* 9 (1963), 643–666
- Levi, R., Shmoys, D.B., und Swamy, C. [2012]: LP-based approximation algorithms for capacitated facility location. *Mathematical Programming A* 131 (2012), 365–379
- Li, S. [2013]: A 1.488-approximation algorithm for the uncapacitated facility location problem. *Information and Computation* 222 (2013), 45–58
- Mahdian, M., und Pál, M. [2003]: Universal facility location. In: *Algorithms – Proceedings of the 11th European Symposium on Algorithms (ESA); LNCS 2832* (G. di Battista, U. Zwick, Hrsg.), Springer, Berlin 2003, pp. 409–421
- Mahdian, M., Ye, Y., und Zhang, J. [2006]: Approximation algorithms for metric facility location problems. *SIAM Journal on Computing* 36 (2006), 411–432
- Manne, A.S. [1964]: Plant location under economies-of-scale-decentralization and computation. *Management Science* 11 (1964), 213–235
- Pál, M., Tardos, É., und Wexler, T. [2001]: Facility location with nonuniform hard capacities. *Proceedings of the 42nd Annual IEEE Symposium on the Foundations of Computer Science* (2001), 329–338
- Shmoys, D.B., Tardos, É., und Aardal, K. [1997]: Approximation algorithms for facility location problems. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (1997), 265–274
- Stollsteimer, J.F. [1963]: A working model for plant numbers and locations. *Journal of Farm Economics* 45 (1963), 631–645
- Sviridenko, M. [2002]: An improved approximation algorithm for the metric uncapacitated facility location problem. In: *Integer Programming and Combinatorial Optimization; Proceedings of the 9th International IPCO Conference; LNCS 2337* (W.J. Cook, A. Schulz, Hrsg.), Springer, Berlin 2002, pp. 240–257
- Vygen, J. [2007]: From stars to comets: improved local search for universal facility location. *Operations Research Letters* 35 (2007), 427–433
- Zhang, J., Chen, B., und Ye, Y. [2004]: A multi-exchange local search algorithm for the capacitated facility location problem. *Mathematics of Operations Research* 30 (2005), 389–403
- Zhang, P. [2007]: A new approximation algorithm for the k -facility location problem. *Theoretical Computer Science* 384 (2007), 126–135

Symbolverzeichnis

\mathbb{N}	Menge der natürlichen Zahlen $\{1, 2, 3, \dots\}$	
$\mathbb{Z} (\mathbb{Z}_+)$	Menge der (nichtnegativen) ganzen Zahlen	
$\mathbb{Q} (\mathbb{Q}_+)$	Menge der (nichtnegativen) rationalen Zahlen	
$\mathbb{R} (\mathbb{R}_+)$	Menge der (nichtnegativen) reellen Zahlen	
\subset	echte Teilmenge	
\subseteq	Teilmenge oder dieselbe Menge	
\cup	disjunkte Vereinigung	
$X \triangle Y$	symmetrische Differenz der Mengen X und Y	
$\ x\ _2$	euklidische Norm eines Vektors x	
$\ x\ _\infty$	∞ -Norm eines Vektors x	
$x \bmod y$	die eindeutig bestimmte Zahl z mit $0 \leq z < y$ und $\frac{x-z}{y} \in \mathbb{Z}$	
x^T, A^T	transponierter Vektor, transponierte Matrix	
$[x]$	die kleinste ganze Zahl größer oder gleich x	
$\lfloor x \rfloor$	die größte ganze Zahl kleiner oder gleich x	
$f = O(g)$	O -Notation	4
$f = \Theta(g)$	Θ -Notation	4
$\text{size}(x)$	Kodierungslänge von x ; Länge des binären Strings x	6, 82, 410
$\log x$	Logarithmus von x zur Basis 2	8
$V(G)$	Knotenmenge des Graphen G	15
$E(G)$	Kantenmenge des Graphen G	15
$G[X]$	der von $X \subseteq V(G)$ induzierte Teilgraph von G	16
$G - v$	der von $V(G) \setminus \{v\}$ induzierte Teilgraph von G	16
$G - e$	der aus G durch Entfernen der Kante e hervorgehende Graph	16
$G + e$	der aus G durch Hinzufügung der Kante e hervorgehende Graph	16
$G + H$	Summe der Graphen G und H	16
G/X	der aus G durch Kontraktion der Kantenmenge X hervorgehende Graph	17
$E(X, Y)$	Menge der Kanten mit einem Endknoten in $X \setminus Y$ und dem anderen in $Y \setminus X$	17
$E^+(X, Y)$	Menge der gerichteten Kanten von $X \setminus Y$ nach $Y \setminus X$	17
$\delta(X), \delta(v)$	$E(X, V(G) \setminus X)$, $E(\{v\}, V(G) \setminus \{v\})$	17
$\Gamma(X), \Gamma(v)$	Menge der Nachbarn der Knotenmenge X bzw. des Knotens v	17

$\delta^+(X), \delta^+(v)$	Menge der in der Knotenmenge X bzw. im Knoten v beginnenden Kanten	17
$\delta^-(X), \delta^-(v)$	Menge der in der Knotenmenge X bzw. im Knoten v ankommenden Kanten	17
2^S	Potenzmenge von S	18
K_n	vollständiger Graph mit n Knoten	18
$P_{[x,y]}$	x - y -Teilweg von P	19
$\text{dist}(v, w)$	Länge des kürzesten x - y -Wege	19
$c(F)$	$\sum_{e \in F} c(e)$ (mit $c : E \rightarrow \mathbb{R}$ und $F \subseteq E$)	19
$K_{n,m}$	vollständiger bipartiter Graph mit n und m Knoten	38
$cr(J, l)$	Anzahl der Kreuzungen der Linie l durch das Polygon J	40, 631
G^*	planares Dual von G	47
e^*	eine Kante von G^* ; duale Kante von e	47
$x^\top y, xy$	Skalarprodukt der Vektoren x und y	57
$x \leq y$	für Vektoren x und y gilt die Ungleichung komponentenweise	57
$\text{rank}(A)$	Rang der Matrix A	59
$\dim X$	Dimension einer nichtleeren Teilmenge $X \subseteq \mathbb{R}^n$	59
I	Einheitsmatrix	62
e_j	j -ter Einheitsvektor	62
A_J	Untermatrix von A bestehend aus den Zeilen mit Index in J	63
b_J	Teilvektor von b bestehend aus den Komponenten mit Index in J	63
$\mathbb{1}$	ein aus lauter Einsen bestehender Vektor	66
A^J	Untermatrix von A bestehend aus den Spalten mit Index in J	67
$\text{conv}(X)$	konvexe Hülle aller Vektoren in X	74
$\det A$	Determinante der Matrix A	82
$\text{sgn}(\pi)$	Signum der Permutation π	83
$E(A, x)$	Ellipsoid	91
$B(x, r)$	euklidische Kugel mit Mittelpunkt x und Radius r	91
$\text{volume}(X)$	Volumen der nichtleeren Menge $X \subseteq \mathbb{R}^n$	91
$\ A\ $	Norm der Matrix A	93
X°	polare Menge von X	104
P_I	ganzzahlige Hülle des Polyeders P	110
$\Xi(A)$	der größte aller Beträge der Unterdeterminanten der Matrix A	112
$P', P^{(i)}$	erste und i -te Gomory-Chvátal-Stützung von P	127
$LR(\lambda)$	Lagrange-Relaxierung	132
$\delta(X_1, \dots, X_p)$	Multischnitt	157
$c_\pi((x, y))$	reduzierte Kosten der Kante (x, y) bezüglich π	173
(\bar{G}, \bar{c})	metrischer Abschluss von (G, c)	174
$\text{ex}_f(v)$	Differenz zwischen ankommendem und abgehendem Fluss im Knoten v	185

$\text{value}(f)$	Wert eines s - t -Flusses f	185
\overleftarrow{G}	der aus G durch Hinzufügung der umgekehrt orientierten Kanten hervorgehende Graph	187
\overleftarrow{e}	gerichtete Kante e mit der umgekehrten Orientierung	187
$u_f(e)$	Residualkapazität der Kante e bezüglich des Flusses f	187
G_f	Residualgraph bezüglich des Flusses f	187
G_f^L	Level-Graph von G_f	195
λ_{st}	minimale Kapazität eines s und t trennenden Schnittes (lokaler Kantenzusammenhang)	204
$\lambda(G)$	minimale Kapazität eines Schnittes in G (Kantenzusammenhang)	212
$v(G)$	maximale Kardinalität eines Matchings in G	260
$\tau(G)$	minimale Kardinalität einer Knotenüberdeckung in G	260
$T_G(x)$	Tutte-Matrix von G , abhängig vom Vektor x	262
$q_G(X)$	Anzahl der ungeraden Zusammenhangskomponenten in $G - X$	265
$\alpha(G)$	maximale Kardinalität einer stabilen Menge in G	284
$\zeta(G)$	minimale Kardinalität einer Kantenüberdeckung in G	284
$r(X)$	Rang einer Menge X in einem Unabhängigkeitssystem	346
$\sigma(X)$	Abschluss einer Menge X in einem Unabhängigkeitssystem	346
$\mathcal{M}(G)$	Kreismatroid eines ungerichteten Graphen G	348
$\rho(X)$	unterer Rang einer Menge X in einem Unabhängigkeitssystem	349
$q(E, \mathcal{F})$	Rangquotient eines Unabhängigkeitssystems (E, \mathcal{F})	349
$C(X, e)$	(für $X \in \mathcal{F}$) der eindeutig bestimmte Kreis in $X \cup \{e\}$, falls $X \cup \{e\} \notin \mathcal{F}$, sonst \emptyset	354
(E, \mathcal{F}^*)	Dual des Unabhängigkeitssystems (E, \mathcal{F})	354
$P(f)$	Polymatroid für eine submodulare Funktion f	386
\sqcup	Freistelle („blank“)	410
$\{0, 1\}^*$	Menge aller binären Strings	410
P	Klasse der polynomiell lösbarer Entscheidungsprobleme	418
NP	Klasse der Entscheidungsprobleme mit Zertifikaten für Ja-Instanzen	419
\bar{x}	Negation des Literals x	422
$coNP$	Klasse der Komplemente von Problemen in NP	433
$\text{OPT}(x)$	Wert einer optimalen Lösung für die Instanz x	436
$A(x)$	Wert des Outputs des Algorithmus A für ein Optimierungsproblem mit Input x	436
$\text{largest}(x)$	größte in der Instanz x auftretende ganze Zahl	438
$H(n)$	$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$	451
$\chi(G)$	chromatische Zahl von G	465
$\omega(G)$	maximale Kardinalität einer Clique in G	465
$\text{Exp}(X)$	Erwartungswert der Zufallsvariable X	474
$\text{Prob}(X)$	Wahrscheinlichkeit des Ereignisses X	474
$\text{SUM}(I)$	Summe über alle Elemente in I	517

$NF(I)$	Output des NEXT-FIT-ALGORITHMUS für die Instanz I	520
$FF(I)$	Output des FIRST-FIT-ALGORITHMUS für die Instanz I	520
$FFD(I)$	Output des FIRST-FIT-DECREASING-ALGORITHMUS für I	522
$G_i^{(a,b)}$	verschobenes Gitter	630
$Q(n)$	konvexe Hülle der Inzidenzvektoren der Touren in K_n	643
$HK(K_n, c)$	Held-Karp-Schranke für die TSP-Instanz (K_n, c)	650
$c_F(X), c_F(x)$	Bereitstellungskosten einer Lösung	665, 686
$c_S(X), c_S(x)$	Servicekosten einer Lösung	665, 686

Personenverzeichnis

- Aardal, K., 405, 665–667, 675, 697, 699, 700
- Aarts, E., 642, 659, 661
- Ackermann, W., 145, 148
- Adleman, L.M., 435, 446
- Agrawal, A., 592, 619
- Agrawal, M., 435, 446
- Aho, A.V., 8, 13, 446
- Ahuja, R.K., 164, 170, 182, 183, 199, 221, 222, 255
- Ajtai, M., 484, 493
- Albrecht, C., 182, 183
- Alekseev, A.S., 344
- Alizadeh, F., 460, 493
- Alspach, B., 379
- Alt, H., 262, 289
- An, H.-C., 689, 699
- Anderson, I., 267, 287, 289
- Anstee, R.P., 338, 343
- Anstreicher, K.M., 132, 138
- Aoshima, K., 37, 54
- Appel, K., 468, 493
- Applegate, D., 641, 652, 654, 658, 659
- Archer, A., 677, 699
- Arkin, E.M., 254, 256
- Armstrong, R.D., 249, 256
- Arora, S., 446, 460, 479–481, 493, 494, 552, 568, 575, 619, 629, 632, 634, 635, 659, 665, 699
- Arya, V., 680, 682, 684, 699
- Asano, T., 478, 493, 494
- Aumann, Y., 552, 568
- Ausiello, G., 446, 479, 493
- Avi-Itzhak, B., 164
- Avidor, A., 478, 494
- Avis, D., 65, 78
- Babenko, M.A., 340, 343
- Bachem, A., 322, 343, 406, 407
- Baker, B.S., 522, 534
- Balakrishnan, V.K., 164
- Balas, E., 512, 514
- Balinski, M.L., 318, 320, 663, 666, 699
- Ball, M.O., 164, 288, 314, 319, 320, 343, 618, 659
- Bansal, N., 531, 534
- Bar-Hillel, Y., 13
- Bar-Yehuda, R., 454, 455, 490, 494
- Barahona, F., 340–343
- Barak, B., 446
- Bazaraa, M.S., 173, 183
- Becker, A., 490, 494
- Becker, M., 560, 568
- Becvar, J., 515
- Beier, R., 511, 512, 515
- Bellare, M., 478, 479, 494
- Bellman, R.E., 168, 171, 173, 175, 183, 223, 237, 256, 503, 515, 655, 659
- Benders, J.F., 699
- Berge, C., 24, 54, 262, 267, 274, 275, 277, 285, 289, 295, 318, 320, 465, 466, 494
- Berkovitch, I., 478, 494
- Berman, L., 578, 579, 621
- Berman, P., 488, 494, 579, 582, 619, 628, 659
- Bern, M., 575, 619
- Bernoulli, J., 564
- Bertsimas, D.J., 78, 138, 573, 604, 619, 620
- Bienstock, D., 547, 568
- Birkhoff, G., 296, 318, 321
- Bixby, R.E., 378, 405, 659

- Björklund, A., 578, 619
 Björner, A., 405
 Bland, R.G., 64, 65, 78, 91, 108
 Blum, M., 501, 502, 515
 Blum, N., 289
 Bock, F.C., 152, 164
 Boesch, F., 566, 568
 Bollobás, B., 54
 Bondy, J.A., 54
 Borchers, A., 581, 619
 Borgwardt, K.-H., 65, 78
 Borradale, G., 215, 222, 575, 619
 Borůvka, O., 141, 164, 166
 Bovet, D.P., 446
 Boyd, E.A., 131, 138
 Boyd, S.C., 648, 659
 Brègman, L.M., 287, 289
 Brooks, R.L., 465, 494
 Bruns, W., 134, 138
 Buchbinder, N., 399, 400, 405, 406
 Buchin, K., 659
 Budach, L., 290
 Burkard, R.E., 296, 321, 625, 659
 Busacker, R.G., 236, 256
 Byrka, J., 579, 586, 587, 591, 619, 675, 699
 Cai, L., 661
 Camion, P., 50, 51, 54
 Caprara, A., 338, 343, 532, 534
 Carathéodory, C., 77, 78, 134, 138, 140, 534
 Carr, R.D., 112, 138, 513, 515, 648, 659
 Cayley, A., 143, 160, 165
 Chakraborty, T., 610, 619
 Chalasani, P., 656, 660
 Chan, T.M., 506, 515
 Chandra, B., 637, 660
 Chang, R.C., 655, 660
 Chao, K.-M., 164
 Charikar, M., 565, 568, 569, 673, 679, 686, 699
 Chazelle, B., 148, 165
 Chekuri, C., 566, 568
 Chen, B., 689, 698, 700
 Chen, J., 475, 494
 Chen, S.-W., 661
 Cheng, X., 618, 620
 Cherian, J., 610
 Cheriton, D., 148, 165
 Cherian, J., 203, 222, 619
 Cherkassky, B.V., 173, 183, 198, 222
 Cheung, H.Y., 210, 222
 Cheung, K.K.H., 221, 222
 Chlebík, M., 489, 494, 579, 619
 Chlebíková, J., 489, 494, 579, 619
 Cholesky, A.-L., 106, 457, 460
 Choukhmane, E., 578, 619
 Christofides, N., 616, 625, 641, 651, 656, 660
 Chu, Y., 152, 165
 Chudak, F.A., 547, 569, 667, 699
 Chudnovsky, M., 466, 494
 Church, A., 412
 Chuzhoy, J., 610, 619
 Chvátal, V., 65, 78, 119, 127, 131, 138–140, 320, 343, 450, 466, 494, 647, 659, 660
 Clarke, M.R.B., 508, 515
 Clementi, A.E.F., 489, 494
 Cobham, A., 7, 13
 Coffman, E.G., 533
 Collatz, L., 379
 Conforti, M., 162, 165
 Cook, S.A., 422, 423, 426, 446
 Cook, W.J., 113, 114, 121, 131, 134, 138, 221, 255, 256, 313, 321, 343, 378, 447, 465, 493, 495, 568, 641, 659, 700
 Cormen, T.H., 13, 164, 182, 221
 Cornuéjols, G., 131, 138, 162, 165, 494, 699
 Correa, J.R., 534
 Cramer, G., 83, 112, 117, 122, 125
 Crescenzi, P., 446, 493
 Crowder, H., 658, 660
 Cunningham, W.H., 138, 221, 222, 248, 255, 256, 313, 314, 316, 321, 343, 369, 378, 379, 392, 405, 406, 648, 659
 Dadush, D., 128, 138
 Dahlhaus, E., 221, 222
 Dantzig, G.B., 63, 64, 78, 79, 123, 127, 138, 140, 189, 222, 248, 256, 500, 503, 515, 642, 659, 660
 de Berg, M., 184, 636, 659
 de Wolf, R., 660

- Dehne, F., 447, 535
Deineko, V.G., 625, 659
Delaunay, B., 161, 166
Dell'Amico, M., 296, 321
Demers, A., 535
Derigs, U., 314, 319, 320
Dessouky, M.I., 215, 224
Dey, S.S., 128, 138
Deza, M.M., 140, 491, 494
di Battista, G., 699, 700
Dial, R.B., 180, 183
Diestel, R., 54, 156, 165
Dijkstra, E.W., 31, 145, 165, 169, 170,
174, 179–181, 183, 238, 244, 254,
543, 545, 550, 577, 579, 614
Dilworth, R.P., 284, 289
Dinic, E.A. = Dinitz, E.A., 194–196, 198,
217, 221, 222
Dinur, I., 454, 479, 494
Dirac, G.A., 49, 54
Dixon, B., 148, 165
Doig, A.G., 652, 661
Dósa, G., 521, 522, 534
Dreyfus, S.E., 182, 576, 577, 614, 619
Du, D.-Z., 580, 581, 618–621
Duan, R., 313, 321
Dulmage, A.L., 285, 290
- Edmonds, J., 7, 13, 27, 54, 89, 106, 108,
118, 119, 138, 139, 151, 152, 154–
157, 162, 163, 165, 166, 172, 183,
193–195, 222, 238, 239, 256, 260,
261, 264, 274, 277, 280–283, 288,
289, 293, 296–299, 302, 313–315,
321, 324, 325, 330, 334, 338, 339,
341, 343, 361–366, 368, 369, 371,
372, 377–379, 386–388, 402, 404,
406, 434, 446, 660
Egerváry, E., 318, 321
Egoryčev, G.P., 287, 289
Eisemann, K., 524, 534
Eisenbrand, F., 131, 136, 139, 532, 534
Eleutério, V., 547, 569
Elias, P., 189, 222
Englert, M., 637, 660
Erdős, P., 288, 289, 343, 492, 495
Erickson, R.E., 614, 619
Erlebach, T., 494
- Euler, L., 15, 35–37, 41, 42, 47, 48, 52,
54, 328, 356, 632
Even, S., 217, 222, 454, 455, 494, 552,
569
- Faenza, Y., 320, 321
Faigle, U., 378
Falikman, D.I., 287, 289
Farkas, G., 73, 74, 79, 97, 117
Feder, T., 262, 289
Feige, U., 398, 404–406, 452, 478, 479,
481, 491, 495, 565, 569, 676, 699
Feinstein, A., 189, 222
Feldman, M., 400, 405, 406
Feng, Q., 580, 619
Fernández-Baca, D., 485, 486, 495
Fernandez de la Vega, W., 524–526, 530–
532, 534
Fiorini, S., 649, 660
Fischetti, M., 338, 343, 515
Fleischer, L.K., 251, 256, 392, 404, 406,
515, 547, 569, 610, 619
Floyd, R.W., 174, 175, 181, 183, 515
Fonlupt, J., 134, 138
Ford, L.R., 171, 173, 175, 183, 188, 189,
193, 213–215, 221, 222, 229–231,
237, 250, 252, 256, 261, 543, 561,
569, 615
Fortune, S., 161, 165, 553, 569
Fourier, J.B.J., 77, 79
Francis, R.L., 699
Frank, A., 99, 108, 159, 164, 165, 211,
219, 221, 222, 288, 289, 343, 365,
371, 373, 378, 379, 389, 402, 405,
406, 556, 560, 567–569, 656, 660
Fredman, M.L., 146, 148, 165, 170, 183
Fremuth-Paeger, C., 283, 289
Fridman, A.A., 222
Friesen, D.K., 475, 494
Frieze, A.M., 508, 515, 655, 660
Frobenius, G., 261, 267, 289
Fuchs, B., 578, 620
Fujishige, S., 159, 165, 194, 198, 199,
217, 223, 392, 398, 404–406
Fujito, T., 488, 494
Fulkerson, D.R., 119, 127, 138, 139, 154,
165, 188, 189, 193, 213–215, 221,
222, 229–231, 237, 250, 252, 256,
261, 284, 289, 357, 377, 379, 466,

- 495, 543, 561, 569, 615, 642, 659, 660
- Fürer, M., 416, 447, 469, 495
- Gabow, H.N., 148, 152, 164, 165, 211, 220, 223, 313, 321, 332, 343, 369, 378, 379, 595, 596, 599, 615, 616, 620
- Gabriel, R., 662
- Gács, P., 97, 108
- Galbiati, G., 655, 660
- Gale, D., 70, 79, 227, 256
- Galil, Z., 148, 165, 198, 223
- Gallai, T., 189, 223, 283, 284, 288–290, 298, 302
- Gallo, G., 182
- Gambosi, G., 446, 493
- Garcia-Diaz, A., 221
- Garey, M.R., 12, 13, 438, 446, 447, 453, 456, 493, 495, 507, 514, 515, 518, 521, 522, 533–535, 575, 581, 620, 628, 660
- Garg, N., 545, 569, 699
- Gärtner, B., 78
- Gauß, C.F., 87
- Gavril, F., 453, 454
- Gavrilova, M., 447, 535
- Geelen, J.F., 264, 290
- Geiger, D., 490, 494
- Gens, G.V., 505, 515
- Geoffrion, A.M., 133, 139
- Gerards, A.M.H., 138, 288, 320, 338, 343
- Ghouila-Houri, A., 124, 139, 388
- Gilbert, E.N., 578, 620
- Giles, R., 27, 54, 118–120, 139, 404, 406
- Gilmore, P.C., 528, 534
- Goemans, M.X., 106, 112, 137, 139, 456, 460, 461, 475–478, 491, 494, 495, 532, 534, 573, 592, 593, 595, 596, 599, 600, 604, 615–618, 620, 621
- Goffin, J.L., 132, 139
- Goldberg, A.V., 172, 173, 181, 183, 199, 203, 221, 223, 234, 255, 256, 283, 290, 338, 343, 620
- Goldfarb, D., 91, 108
- Goldreich, O., 446, 479, 494
- Goldwasser, S., 495
- Gomory, R.E., 119, 127, 128, 131, 138, 139, 204–206, 210, 212, 218, 219, 223, 320, 336, 337, 342, 343, 528, 534, 594–596, 613, 615
- Gondran, M., 164, 183, 221, 255, 378
- Gonzalez, T., 623, 662
- Gowen, P.J., 236, 256
- Goyal, N., 611, 613, 620
- Graham, R.L., 12, 13, 54, 221, 256, 289, 320, 378, 405, 533–535, 575, 620, 628, 660
- Grandoni, F., 613, 619, 620
- Graver, J.E., 114, 139
- Graves, R.L., 139
- Graves, S.C., 535
- Grigoriadis, M.D., 547, 569
- Gröpl, C., 585, 620
- Grötschel, M., 54, 87, 91, 93, 96, 99–101, 103, 106–108, 221, 254, 256, 289, 320–322, 343, 378, 390, 392, 405–407, 468, 495, 528–530, 533, 556, 618, 643, 647–649, 660
- Guan, M., 328, 343
- Gubeladze, J., 138
- Guha, S., 664, 673, 675, 676, 686, 699
- Gupta, A., 613, 618, 620
- Guruswami, V., 556, 569
- Gusfield, D., 210, 223
- Gutin, G., 659
- Guy, R., 165, 343, 379, 406
- Haase, C., 445, 447
- Hadlock, F., 54, 340, 343
- Hajiaghayi, M.T., 568
- Hajnal, A., 569
- Haken, W., 468, 493
- Hall, M., 223, 256
- Hall, P., 260, 261, 285, 290
- Halldórsson, M.M., 488, 495
- Halmos, P.R., 260, 290
- Hamburger, M.J., 663, 700
- Hammer, P.L., 54, 139, 321, 379, 406, 496, 534
- Han, X., 534
- Han, Y., 12, 13, 174, 183
- Hanan, M., 575, 620
- Hanani, H., 165, 343, 379, 406
- Hansen, M.D., 565, 569
- Hao, J., 211, 223
- Harrelson, C., 181, 183
- Harvey, D., 416, 447

- Harvey, W., 131, 139
Hassin, R., 253, 256
Håstad, J., 461, 479, 481, 491, 495, 569
Hausmann, D., 349, 360, 377, 379
Hazan, E., 552, 568
Heawood, P.J., 468, 495
Held, M., 138, 649–653, 655, 660, 662
Held, S., 465, 495
Hell, P., 379
Helsgaun, K., 641, 660
Henk, M., 138
Henzinger, M.R. = Rauch, M., 148, 165, 170, 183, 211, 220, 223
Hetzl, A., 576, 620
Hierholzer, C., 35, 54
Hirai, H., 562, 569
Hitchcock, F.L., 222, 228, 229, 256
Hoberg, R., 533, 534
Hochbaum, D.S., 136, 139, 455, 489, 493, 495, 532–534, 568, 618, 664, 700
Hoey, D., 161, 166
Hoffman, A.J., 61, 79, 118, 122–124, 139, 214, 223, 252, 256, 296
Hoyer, I., 462, 495
Hoogeveen, J.A., 656, 660
Hopcroft, J.E., 8, 13, 47, 54, 262, 283, 286, 290, 416, 446, 447, 553, 569
Hoppe, B., 251, 257
Horowitz, E., 446, 493, 533, 534
Hougardy, S., 13, 175, 183, 479, 495, 577, 579, 620, 658, 660
Hsu, W.L., 489, 495
Hu, T.C., 204–206, 210, 212, 215, 218, 219, 223, 336, 337, 342, 567, 569, 594–596, 613, 615
Hurkens, C.A.J., 624, 660
Husfeldt, T., 619
Hwang, F.K., 580, 618, 620
Hwang, R.Z., 655, 660

Ibaraki, T., 211, 212, 220–222, 224, 398, 406, 407, 541, 554, 569, 570
Ibarra, O.H., 504, 505, 515
Iri, M., 37, 54, 236, 257
Itai, A., 552, 569
Iudin, D.B., 91, 108
Iwama, K., 493
Iwata, S., 264, 290, 392, 396, 403–407

Iyengar, G., 547, 568
Jain, K., 604, 605, 607, 609, 610, 617–619, 621, 665, 667–670, 672, 677, 687, 697, 698, 700
Jansen, B.M.P., 659
Jansen, K., 531, 535, 699
Jarník, V., 145, 166
Jelinek, F., 205, 223
Jenkyns, T.A., 349, 360, 379
Jensen, P.M., 403, 407
Jewell, W.S., 236, 257
Jin, Z., 249, 256
John, F., 91
Johnson, D.B., 170, 173, 184
Johnson, D.S., 12, 13, 222, 438, 446, 447, 450, 452, 453, 456, 474, 475, 477, 493–495, 507, 514, 515, 518, 521, 522, 533–535, 575, 581, 620, 628, 652, 657, 660, 661
Johnson, E.L., 54, 139, 321, 325, 330, 334, 339, 341, 343, 379, 406, 496, 534
Johnson, S., 127, 138, 642, 659, 660
Jothi, R., 615, 621
Jünger, M., 138, 569, 654, 659, 661
Jungnickel, D., 221, 256, 283, 289, 659

Kahn, A.B., 33, 54
Kaibel, V., 319, 322, 569, 620, 657, 661
Kale, S., 460, 493, 552, 568
Kamidoi, Y., 221, 223
Kann, V., 446, 493
Kannan, R., 131, 138, 140
Karzanov, A.V., 236
Karakostas, G., 547, 569
Karel, C., 661
Karger, D.R., 148, 166, 211, 220, 223
Karloff, H., 568, 637, 660
Karmarkar, N., 98, 108, 527–531, 533, 535
Karp, R.M., 99, 108, 138, 150, 152, 166, 172, 176, 178, 183, 184, 193–195, 222, 238, 239, 256, 261, 262, 283, 286, 290, 422, 426, 427, 431–433, 446, 447, 503, 515, 527–531, 533, 535, 556, 569, 574, 621, 628, 649–653, 655, 660–662
Karpinski, M., 579, 621, 628, 659, 661

- Karzanov, A.V., 33, 55, 194, 196, 217, 221–223, 257, 262, 283, 286, 290, 334, 338, 340, 343, 344, 562, 569
- Kaski, P., 619
- Kawarabayashi, K., 563, 569, 570
- Kayal, N., 435, 446
- Kellerer, H., 506, 514, 515
- Kelner, J.A., 65, 79
- Kenyon, C. = Mathieu, C., 534, 575, 619
- Kerivin, H., 618
- Kern, W., 620
- Kernighan, B.W., 637–641, 649, 656, 657, 659–661
- Khachiyan, L.G., 91, 97, 98, 108, 419, 530, 547, 569
- Khan, A., 531, 534
- Khandekar, R., 699
- Khanna, S., 465, 495, 566, 568, 610, 619
- Khintchine, A., 86, 108
- Khot, S., 454, 496
- Khuller, S., 178, 182, 184, 615, 621, 664, 675, 676, 699
- Kiefer, S., 445, 447
- Kim, C.E., 504, 505, 515
- King, V., 148, 166
- Klee, V., 65, 79
- Klein, M., 231, 257
- Klein, P.N., 148, 166, 183, 215, 222, 575, 592, 593, 619, 621, 636, 661
- Kleinberg, J., 566, 570, 620
- Klimov, V.S., 344
- Klinz, B., 251, 257
- Knuth, D.E., 13, 33, 55, 161, 166, 460, 496
- Kobayashi, Y., 403, 407, 563, 569
- Koch, J., 468, 493
- Koivisto, M., 619
- Kolliopoulos, S.G., 665, 700
- Kolmogorov, V., 313, 321
- Könemann, J., 545, 569, 617, 618, 621
- König, D., 30, 38, 55, 135, 260, 261, 284, 290, 318, 463, 496
- Koopmans, T.C., 78, 79
- Korst, J., 642, 661
- Korte, B., 54, 139, 145, 166, 183, 221, 255, 321, 322, 343, 349, 360, 362, 377, 379, 385, 403, 405–407, 496, 506, 507, 515, 534, 568, 576, 621, 660
- Kortsarz, G., 610, 621
- Korupolu, M., 680, 689, 700
- Kou, L.T., 578, 579, 621
- Krauthgamer, R., 610, 621
- Krentel, M.W., 657, 661
- Krogdahl, S., 378
- Kruskal, J.B., 61, 79, 122–124, 139, 144, 153, 154, 161, 166, 296, 360, 363, 384
- Kuehn, A.A., 663, 700
- Kuhn, H.W., 70, 77, 79, 139, 222, 261, 290, 294, 319, 321
- Kuich, W., 224
- Kumar, A., 620
- Kumar, M.P., 197, 224
- Kuratowski, K., 43, 44, 46, 52, 53, 55
- Ladner, R.E., 435, 447
- Lagergren, J., 485, 486, 495
- Lam, T.-W., 661
- Lampis, M., 628, 661
- Land, A.H., 652, 661
- Langley, R.W., 173, 183
- Lasserre, J.B., 116, 139
- Lau, L.C., 210, 222, 469, 496, 618
- Laurent, M., 491, 494
- Lawler, E.L., 183, 256, 288, 313, 319, 320, 372, 378, 506, 515, 533–535, 659
- Lecerf, G., 416, 447
- Lee, J.R., 565, 569, 610, 621, 649, 661
- Lee, R.C.T., 655, 660
- Lee, Y.T., 98, 108, 199, 224, 245, 257, 396, 407
- Legendre, A.M., 41, 55
- Lehman, A., 357, 379
- Leighton, T., 549–552, 565, 570
- Leiserson, C.E., 13, 164, 182, 221
- Lenstra, H.W., 525, 535
- Lenstra, J.K., 534, 535, 642, 659
- Letchford, A.N., 336, 337, 344
- Leung, K.M., 210, 222
- Leung, V.J., 515
- Levi, R., 689, 700
- Levin, A., 136, 139
- Levine, M.S., 211, 223
- Levner, E.V., 505, 515

- Lewis, H.R., 416, 447
Lewis, P.M., 624, 661
Li, R., 534
Li, S., 675, 700
Li, Y., 131, 138
Lieberherr, K., 492, 496
Liebling, T., 138
Lin, S., 637–641, 649, 656, 657, 659–661
Linhares Sales, C., 496
Linial, N., 465, 495, 552, 570
Lipton, R.J., 313, 321
Little, J.D.C., 652, 661
Liu, T., 152, 165
Liu, X., 494
Lomonosov, M.V., 221, 222, 562, 570
London, E., 552, 570
Louveaux, Q., 138
Lovász, L., 54, 87, 91, 93, 96, 97, 99–
101, 103, 106–108, 159, 165, 166,
221, 254–256, 264, 268, 269, 277,
288–290, 318, 320, 321, 343, 378,
385, 390, 392, 403, 405–407, 447,
450, 461, 466, 468, 493, 495, 496,
528–530, 533, 556, 568–570, 621
Löwner, K., 91
Lucchesi, C.L., 555, 556, 567, 570
Lueker, G.S., 524–526, 530–532, 534
Lund, C., 494
Lyusternik, L.A., 223

Madduri, K., 184
Mader, W., 156, 212, 216, 224
Mądry, A., 262, 290
Maffioli, F., 655, 660
Magnani, T.L., 162, 164, 182, 221, 255,
288, 320, 343, 618, 659
Mahajan, S., 461, 478, 496
Mahdian, M., 672, 675, 686, 688–690,
698, 700
Maheshwari, S.N., 197, 203, 222, 224
Mahjoub, A.R., 618
Malhotra, V.M., 197, 224
Mangaserian, O., 515
Manne, A.S., 663, 700
Manokaran, R., 569
Manu, K.S., 164, 165
Marchetti-Spaccamela, A., 446, 493
Markakis, E., 700
Markowsky, G., 578, 579, 621

Marsh, A.B., 313, 314, 316, 321, 324,
338, 344
Martello, S., 296, 321, 514
Martin, A., 138, 576, 621
Massar, S., 660
Mathieu, C. = Kenyon, C., 534, 575, 619
Matoušek, J., 78
Matsui, T., 245, 257
Matsumoto, K., 558, 570
Matsuyama, A., 578, 621
Matula, D.W., 545, 571
Mayeda, W., 205, 223
McCormick, S.T., 236, 257, 405
McGeoch, L.A., 652, 660
Megiddo, N., 178, 184
Mehlhorn, K., 13, 183, 203, 222, 289,
313, 321, 560, 568, 579, 621
Meinardus, G., 379
Melkonian, V., 610, 621
Mendelsohn, N.S., 285, 290
Menger, K., 186, 190–192, 215, 219, 220,
224, 260, 341, 538, 542, 543, 554,
613
Meyer, R.R., 110, 111, 139, 515
Meyer, U., 184
Meyerson, A., 699
Micali, S., 283, 290
Michiels, W., 642, 661
Middendorf, M., 557, 570
Mihail, M., 621
Miklós, D., 343
Milková, E., 141, 166
Miller, C.E., 657, 661
Miller, D.J., 379
Miller, R.E., 447, 515, 569, 621
Minkowski, H., 62, 73, 74, 79
Minoux, M., 164, 183, 221, 255, 378
Minty, G.J., 22, 23, 55, 65, 79
Mirchandani, P.B., 699
Mirrokni, V.S., 398, 404–406
Mitchell, J., 629, 661
Mölle, D., 620
Monge, G., 294, 321
Monma, C.L., 164, 288, 320, 343, 360,
362, 379, 614, 618, 619, 659
Moore, E.F., 30, 55, 171, 173, 175, 184,
237, 578
Motwani, R., 262, 289, 494, 656, 660

- Motzkin, T.S., 77, 79
 Mozes, S., 172, 184
 Mucha, M., 264, 290
 Müller, D., 547, 570
 Mulmuley, K., 264, 290
 Munagala, K., 699
 Munkres, J., 294, 321
 Murty, K.G., 661
 Murty, U.S.R., 54
 Naddef, D., 138, 320, 321, 654, 661
 Nagamochi, H., 211, 212, 220–224, 398, 406, 407, 541, 570
 Nagarajan, V., 609, 621
 Näher, S., 321
 Namaad, A., 198, 223
 Naor, J., 406
 Nash-Williams, C.S.J.A., 156, 157, 159, 166, 369, 377, 379, 554, 566, 570
 Naves, G., 560, 568, 570
 Nemhauser, G.L., 54, 135, 138, 139, 164, 288, 320, 343, 405, 406, 489, 495, 509–512, 515, 618, 659, 699
 Nemirovskii, A.S., 91, 108
 Nešetřil, J., 141, 145, 166
 Nešetřilová, H., 141, 166
 Nicholls, W., 161, 166
 Nierhoff, T., 620
 Nishimura, K., 220, 224
 Nishizeki, T., 558, 568, 570
 Nolles, W., 655, 662
 Okamura, H., 558, 566, 570
 Olver, N., 611, 613, 620, 621
 Orden, A., 64, 79, 229, 257
 Ore, O., 252, 257
 Oriolo, G., 320, 321, 620
 Orlin, J.B., 164, 170, 181–184, 199, 211, 221–224, 240–242, 244, 245, 249, 254, 255, 257, 407, 440, 447
 Oxley, J.G., 378
 Padberg, M.W., 78, 99, 107, 108, 336–338, 344, 643, 647–649, 658, 660
 Pál, M., 686, 689, 690, 700
 Pallottino, S., 182
 Pandit, V., 699
 Pap, G., 403, 407
 Papadimitriou, C.H., 99, 108, 222, 289, 320, 416, 429, 444–447, 452, 461, 472, 483–486, 488, 493, 496, 514, 519, 535, 626, 628, 642, 649, 657–659, 661
 Pardalos, P.M., 184
 Pashkovish, K., 621
 Paul, M., 289
 Persiano, G., 494
 Petersen, J., 262, 287, 290, 605, 617
 Pettie, S., 148, 166, 171, 174, 184, 313, 321
 Pfeiffer, F., 557, 570
 Pferschy, U., 506, 514, 515
 Phillips, C.A., 515
 Phillips, D.T., 221
 Phillips, S., 215, 224
 Picard, J.-C., 214, 224
 Pisinger, D., 504, 514, 515
 Plassmann, P., 575, 619
 Plaxton, C., 680, 689, 700
 Plotkin, S.A., 242, 243, 249, 257, 531, 535, 620
 Plummer, M.D., 269, 277, 288, 343
 Pokutta, S., 136, 139, 660
 Poljak, S., 554, 569
 Pollak, H.O., 578, 620
 Polyak, B.T., 132, 139
 Pomerance, C., 435, 446
 Prädel, L., 531, 535
 Pratt, V., 435, 447, 515
 Prim, R.C., 145, 146, 148, 161, 166, 169, 179, 384, 579
 Prömel, H.J., 221, 255, 479, 495, 568, 576, 579, 618, 620, 621
 Protasi, M., 446, 493
 Prüfer, H., 160, 166
 Pulleyblank, W.R., 120, 138–140, 221, 255, 289, 320, 321, 325, 343, 344, 378, 648, 649, 659, 660
 Punnen, A.P., 440, 447, 659
 Queyranne, M., 213, 214, 224, 396, 398, 406, 407, 531, 535
 Rabani, Y., 552, 568
 Rabin, M.O., 264, 290
 Rabinovich, Y., 552, 570
 Räcke, H., 565, 570

- Radhakrishnan, J., 488, 495
Radke, K., 547, 570
Rado, R., 361, 363, 364, 377, 379, 380
Radzik, T., 178, 184
Raghavachari, B., 178, 182, 184, 469, 495, 615, 621
Raghavan, P., 475, 496, 564, 570, 665, 699
Raghavendra, P., 569, 649, 661
Rajagopalan, R., 677, 699
Rajaraman, R., 680, 689, 700
Ramachandran, V., 148, 166, 171, 184
Ramaiyer, V., 579, 582, 619
Raman, R., 170, 184
Ramesh, H., 461, 478, 496
Rao, A., 656, 660
Rao, M.R., 99, 108, 336–338, 344
Rao, S., 183, 199, 211, 220, 223, 549–552, 565, 568, 570, 635, 661, 665, 699, 700
Rastogi, R., 620
Rauch, M. = Henzinger, M.R., 148, 165, 170, 183, 220, 223
Ravi, R., 592, 593, 609, 618, 619, 621
Raz, R., 452, 496
Recsiki, A., 378
Rédei, L., 51, 55
Reed, B.A., 496, 563, 569
Regev, O., 454, 496
Reinelt, G., 138, 336, 337, 344, 659
Richards, D.S., 618
Richter, S., 620
Rinaldi, G., 138, 659
Rinnooy Kan, A.H.G., 534, 535, 659
Ripphausen-Lipa, H., 568
Rivest, R.L., 13, 164, 182, 221, 515
Rizzi, R., 284, 290, 342, 344, 398, 407
Robbins, H.E., 51, 55, 566, 568
Robertson, N., 52, 53, 55, 468, 469, 494, 496, 562, 563, 567, 570
Robins, G., 579, 583, 586, 621
Robinson, S.M., 515
Röck, H., 404, 406
Röglin, H., 511, 515, 637, 660
Rohe, A., 313, 321, 641, 659
Rose, D.J., 219, 224
Rosenberg, I.G., 140
Rosenkrantz, D.J., 624, 661
Rosenstiehl, P., 379
Rossmanith, P., 620
Rothberg, E.E., 652, 660
Rothschild, B., 562, 570
Rothvoß, T., 136, 140, 342, 344, 532–534, 619, 620, 649, 662
Rubinstein, J.H., 618, 621
Ruhe, G., 221, 256
Rumely, R.S., 435, 446
Rustin, R., 165
Saberi, A., 700
Sack, J.-R., 447, 535
Safra, S., 452, 454, 465, 479, 480, 494–496
Sahni, S., 446, 493, 504, 515, 533, 534, 623, 662
Saito, N., 558, 570
Sanders, D.P., 496
Sanders, P., 13, 471, 496
Sanità, L., 136, 140, 619
Sankowski, P., 264, 290
Sauer, N., 165, 343, 379, 406
Saxena, N., 435, 446
Schäfer, G., 313, 321
Scheffler, P., 568, 570
Scheifele, R., 245, 257
Schietke, J., 183
Schmied, R., 628, 661
Schönhage, A., 416, 447
Schönheim, J., 165, 343, 379, 406
Schrader, R., 385, 405, 506, 507, 515
Schrijver, A., 74, 78, 87, 91, 93, 96, 99–101, 103, 106–108, 121, 127, 129–131, 134, 138, 140, 164, 183, 221, 222, 255, 257, 287, 289, 291, 318, 322, 339, 343, 378, 390, 392–395, 404–407, 468, 495, 528–530, 533, 556, 568, 621, 659
Schulz, A.S., 136, 139, 254, 257, 440, 447, 700
Schwartz, R., 406
Schwarz, U.M., 531, 535
Schwärzler, W., 560, 570
Sebő, A., 134, 140, 333, 344, 558, 563, 568, 571, 613
Sedeño-Noda, A., 181, 184
Seiden, S.S., 523, 535
Sewell, E.C., 465, 495

- Seymour, P.D., 52, 53, 55, 127, 140, 222, 333, 344, 376, 380, 493, 494, 496, 557, 558, 562, 563, 566–568, 570, 571
- Sgall, J., 521, 534
- Shahrokhi, F., 545, 571
- Shamir, A., 552, 569
- Shamos, M.I., 161, 166
- Shannon, C.E., 189, 222
- Shenoy, N., 161, 166
- Shepherd, F.B., 139, 611, 613, 620
- Sherman, J., 552, 571
- Shiloach, Y., 198, 216, 224
- Shioura, A., 199, 224
- Shisha, O., 79
- Shmonin, G., 532, 534
- Shmoys, D.B., 489, 493, 495, 531–535, 545, 552, 568, 620, 651, 659, 662, 665–667, 677, 689, 697, 699, 700
- Shor, N.Z., 91, 108
- Sidford, A., 98, 108, 199, 224, 245, 257, 396, 407
- Silvanus, J., 577, 620
- Silverberg, E.B., 254, 256
- Simchi-Levi, D., 522, 535
- Singh, M., 469, 496, 609, 618, 621, 689, 699
- Skutella, M., 137–139, 251, 256, 620
- Slavík, P., 452, 496
- Sleator, D.D., 198, 199, 224
- Smith, J.M., 618, 621
- Smith, W.D., 635, 661
- Sós, V.T., 165, 343, 407, 569
- Specker, E., 492, 496
- Spencer, T., 148, 165
- Sperner, E., 285, 291
- Spielman, D.A., 65, 79
- Spirakis, P., 257
- Stauffer, G., 320, 321
- Stearns, R.E., 624, 661
- Steger, A., 479, 495, 576, 618, 621
- Steiglitz, K., 289, 320, 429, 446, 452, 496, 514, 642, 658, 659, 661
- Stein, C., 13, 164, 182, 220, 221, 223
- Steinitz, E., 74, 79, 107, 108
- Steurer, D., 471, 496, 649, 661
- Stirling, J., 3, 13
- Stockmeyer, L., 438, 447, 456, 462, 495, 496
- Stoer, M., 211, 212, 224, 618
- Stollsteimer, J.F., 663, 700
- Strassen, V., 416, 447
- Su, X.Y., 216, 224
- Subramani, K., 184
- Subramanian, S., 183
- Sudan, M., 478, 479, 494
- Svensson, O., 655, 662, 689, 699
- Sviridenko, M., 534, 664, 667, 676, 700
- Swamy, C., 621, 689, 700
- Swamy, M.N.S., 222
- Sweeny, D.W., 661
- Sylvester, J.J., 143, 166
- Szegedy, B., 288, 291
- Szegedy, C., 288, 291
- Szegedy, M., 494, 495
- Szigeti, Z., 288, 291
- Szőnyi, T., 343
- Takada, H., 493
- Takahashi, M., 578, 621
- Takaoka, T., 174, 183
- Tang, L., 221, 222
- Tardos, É., 99, 108, 138, 221, 234, 242, 243, 249, 251, 255, 257, 378, 379, 531, 535, 567, 569, 610, 620, 621, 665–667, 689, 690, 697, 699, 700
- Tarjan, R.E., 33, 47, 54, 55, 145, 146, 148, 161, 164–166, 170, 183, 198, 199, 203, 215, 217, 221–224, 234, 255, 256, 289, 313, 321, 515
- Tarnawski, J., 655, 662
- Tarry, G., 30
- Taub, A.H., 79
- Teng, S.-H., 65, 79
- Teo, C., 573, 604, 619
- Thatcher, J.W., 447, 515, 569, 621
- Theis, D.O., 336, 337, 344
- Thomas, R., 494, 496
- Thomassen, C., 39, 43, 44, 55
- Thompson, C.D., 475, 496, 564, 570
- Thorup, M., 170, 184, 221, 224
- Thulasiraman, K., 222
- Tindell, R., 566, 568
- Tiwary, H.R., 660
- Todd, M.J., 91, 108
- Tolstoī, A.N., 231, 257

- Tomizawa, N., 238, 257
- Tóth, C.D., 447, 535
- Toth, P., 514
- Tovey, C., 637, 660
- Traub, V., 656, 662
- Trémaux, C.P., 30
- Trevisan, L., 489, 492, 494, 496
- Triesch, E., 655, 660, 662
- Tsitsiklis, J.N., 78
- Tucker, A.W., 70, 79, 139, 222, 657, 661
- Tunçel, L., 203, 224
- Turing, A.M., 409–413, 415–417, 419, 422, 423, 441–443, 447
- Tutte, W.T., 41, 43, 55, 156, 157, 166, 262–267, 277, 287, 291, 295, 318, 320, 327, 339, 344, 570
- Tuza, Z., 534
- Ullman, J.D., 8, 13, 416, 446, 447, 535
- Ullmann, Z., 509–512, 515
- Vaidya, P.M., 105, 108
- van der Hoeven, J., 416, 447
- van der Waerden, B.L., 287, 289
- van Emde Boas, P., 416, 446
- van Leeuwen, J., 446
- van Vliet, A., 523, 535
- Varadarajan, K.R., 313, 322
- Varadarajan, S., 615, 621
- Vaughan, H.E., 260, 290
- Vazirani, U.V., 264, 290, 552, 568
- Vazirani, V.V., 220, 224, 264, 283, 290, 291, 493, 618, 621, 667, 668, 670, 677, 687, 698, 700
- Végh, L., 655, 662
- Veinott, A.F., 123, 140, 614, 619
- Vempala, S., 112, 138
- Vetta, A., 610, 619
- Vielma, J.P., 128, 138
- Vishkin, U., 615, 621
- Vizing, V.G., 463, 464, 470, 496
- Vöcking, B., 511, 512, 515, 637, 660
- von Neumann, J., 70, 79, 296, 321
- von Randow, R., 378
- Vondrák, J., 398, 404–406
- Voronoi, G., 161
- Vušković, K., 494
- Vygen, J., 13, 183, 217, 225, 245, 256, 257, 395, 407, 447, 501, 515, 547,
- 554, 557, 565, 568, 570, 571, 577, 578, 620, 621, 628, 655, 656, 659, 660, 662, 664, 676, 689, 691, 696, 699, 700
- Wagner, D., 321, 560, 568, 571
- Wagner, F., 211, 212, 224
- Wagner, H.M., 229, 257
- Wagner, K., 43, 46, 55
- Wagner, R.A., 576, 577, 614, 619
- Walter, M., 319, 322
- Wang, D., 211, 223
- Wang, X., 620
- Warme, D.M., 579, 621
- Warshall, S., 174, 175, 181, 183, 184
- Weber, G.M., 314, 322
- Wegener, I., 446
- Weihe, K., 215, 225, 560, 568, 571
- Weismantel, R., 138, 254, 257, 405
- Welsh, D.J.A., 378
- Weltge, S., 657, 661
- Werners, B., 662
- Wetterling, W., 379
- Wexler, T., 689, 690, 700
- Weyl, H., 73, 74, 79
- Whinston, A., 562, 570
- White, N., 378, 405
- Whitney, H., 34, 49, 54, 55, 191, 225, 355, 356, 380
- Wigderson, A., 491, 496
- Willard, D.E., 148, 165, 170, 183
- Williamson, D.P., 456, 460, 461, 475–478, 491, 493–495, 515, 593, 595, 596, 599–601, 603, 610, 616–621, 651, 662
- Williamson, M., 184
- Wilson, R.J., 54
- Winter, P., 579, 618, 621
- Woeginger, G.J., 251, 257, 507, 516, 624, 625, 654, 659, 660, 662
- Wolfe, P., 64, 79, 139, 663, 699
- Wollan, P., 563, 570
- Wolsey, L.A., 114, 132, 135, 138, 140, 162, 164, 651, 662, 699
- Wong, S.C., 396, 407
- Wu, B.Y., 164
- Wulff-Nilsen, C., 172, 184
- Wyllie, J., 553, 569

Xu, Y., 369, 379

Yamashita, Y., 493

Yannakakis, M., 220, 222, 224, 342, 344, 445, 447, 461, 472, 475, 483, 484, 486, 488, 493, 496, 497, 626, 628, 657, 661

Yao, A.C., 534

Ye, Y., 672, 675, 688, 689, 698, 700

Yener, B., 620

Yoshida, N., 221, 223

Young, N., 178, 182, 184, 545, 571

Younger, D.H., 555, 556, 567, 570

Yue, M., 522, 535

Zachariasen, M., 579, 621

Zambelli, G., 162, 165

Zelikovsky, A.Z., 579, 582, 583, 586, 621, 622

Zemel, E., 512, 514

Zemlin, R.A., 657, 661

Zenklausen, R., 620

Zhang, J., 672, 675, 688, 689, 698, 700

Zhang, P., 680, 700

Zhang, Y., 580, 619

Zheng, H., 475, 494

Zhou, H., 161, 166

Zhou, X., 568, 570

Ziegler, G.M., 254, 257, 405

Zimmermann, U., 404, 406

Zipkin, P.H., 535

Zuckerman, D., 465, 481, 497

Zwick, U., 174, 184, 478, 494, 699, 700

Stichwortverzeichnis

- ϵ -Dominanz, 506
 ϵ -DOMINANZ-PROBLEM, 506
 γ -Expander, 484
0-1-String, 12, 410
1-Baum, 649, 650
2-Matching-Ungleichung, 646, 654, 657
2-Polymatroid, 403
2-fach kantenzusammenhängender Graph, 51
2-fach zusammenhängender Graph, 34
2-opt, 636
2-opt Tour, 637
2SAT, 426, 443
3-DIMENSIONALES MATCHING (3DM), 430
3-MATROID-INTERSEKTION, 444
3-OCCURRENCE-MAX-SAT-PROBLEM, 484, 486
3-OCCURRENCE-SAT, 443
3-OPT-ALGORITHMUS, 640
3-Schnitt, 157, 220
3-dimensionales Polytop, 107
3-fach zusammenhängender Graph, 43
3-fach zusammenhängender planarer Graph, 107
3-opt, 636, 637
3-opt Tour, 640
3DM, 431
3SAT, 425, 426
abhängige Menge, 345
Abschluss, 346, 352
Abschlussoperator, 382
Abschluss-Orakel, 360
absoluter Approximationsalgorithmus, 449, 464, 469, 470, 504
abstraktes Dual, 49, 54, 356
Ackermann-Funktion, 145, 148
ADD, 689
Adjazenzliste, 29, 418
Adjazenzmatrix, 29
affin unabhängig, 76
aktive Knotenmenge (Netzwerk-Design), 596, 616
aktiver Knoten (Netzwerkflüsse), 196
algebraische Zahlen, 416
Algorithmus, 6, 409, 412
exakter, 436
Algorithmus für ein Entscheidungsproblem, 418
ALGORITHMUS FÜR STARKE ZUSAMMENHANGSKOMPONENTEN, 31–34, 597
allgemeiner Blütenwald, 277, 300, 310
ALLGEMEINES SPARSEST-CUT-PROBLEM, 548
Alphabet, 410
alternierende Ohrenzerlegung, 268–271
alternierender Spaziergang, 637
geschlossener, 638, 640
guter, 638, 640
guter geschlossener, 638
alternierender Wald, 276
alternierender Weg, 262, 301, 366
Angebot, 227
Angebotskante, 192, 537
Anti-Arboreszenz, 32
Antiblocker, 492
Antikette, 284
Antimatroid, 382, 402
Approximationsalgorithmus
 absoluter, 449, 464, 469, 470, 504

- asymptotischer k -, 470
 k -, 449
- Approximationsgüte, 449
- Approximationsschema, 471, 472, 481, 483, 484, 508, 575, 629, 635
 - asymptotisches, 471, 472, 526
 - voll-polynomielles, 471, 505–507, 544, 545
 - voll-polynomielles asymptotisches, 471, 527, 530
- Approximationszahl, 471
 - asymptotische, 471
- äquivalente boolesche Formeln, 445
- äquivalente Probleme, 142
- Arboreszenz, 21, 28, 154, 157, 159, 164, 383, *siehe* MINIMUM-WEIGHT-ARBOR-ESSENCE-PROBLEM, *siehe* MINIMUM-WEIGHT-ROOTED-ARBORESCENCE-PROBLEM
- Arboreszenz-Polytop, 163
- ARORAS ALGORITHMUS, 634, 635
- Artikulationsknoten, 20
- ASYMMETRISCHES TSP, 623, 655
- asymptotische Approximationsgüte, 470
- asymptotische Approximationszahl, 471
- asymptotischer k -Approximationsalgorithmus, 470
- asymptotisches Approximationsschema, 471, 472, 526
- aufspannender Baum, 21, 50, 141, 157, 356, 445, 649, *siehe* MINIMUM-SPANNING-TREE-PROBLEM
- aufspannender Teilgraph, 16
- augmentieren (Netzwerkflüsse), 187
- augmentierender Kreis, 231, 232
- augmentierender Weg, 187–189, 193, 262, 274, 285, 302
- Ausgangsgrad, 17
- äußere Blüte, 277
- äußerer Knoten, 276, 277
- äußeres Gebiet, 40, 53, 558, 577, 614
- AUSWAHLPROBLEM, 501, 502
- azyklischer Digraph, 23, 33, 54
- b -Faktor, 323
- b -Fluss, 227
- b -Fluss zugehörig zu einer Baumstruktur, 246
- b -Matching, 323, 324, *siehe* MAXIMUM-WEIGHT- b -MATCHING-PROBLEM
- b -Matching-Polytop, 324, 325, 337
- Backtracking, 3
- Balance, 227
- balancierter Schnitt, 564
- BAR-YEHUDA-EVEN-ALGORITHMUS, 454
- Barriere, 265
- Basis, 68, 345, 350, 376
- Basislösung, 59, 508, 605, 607
- Basis-Obermengen-Orakel, 359
- Basispolyeder, 392
- Baum, 20, 28, 383
- Baumdarstellung, 26, 126, 300, 602, 607
- Baumlösung, 245, 254
- Baumpolytop, 153, 154, 162
- Baumrepräsentation, 221
- Baumstruktur, 246
 - stark zulässige, 246
 - zulässige, 246
- BAUMWEG, 302, 310
- Baumweite, 52, 567
- Baumzerlegung, 52, 563
- Bellmanns Optimalitätsprinzip, 168
- benachbarte Kanten, 16
- benachbarte Knoten, 15
- berechenbar in polynomieller Zeit, 411
- berechenbare Funktion, 411, 412
- berechenbare Relation, 411
- Berechnungsproblem, 411
- Bereitstellungskosten, 665
- Berge-Tutte-Formel, 267, 277, 295, 318, 320
- beschränkte Standortprobleme, 686
- beschränktes LP, 111
- beschränktes Polyeder, 97
- BESCHRÄNKTES STANDORTPROBLEM METRISCHES, 687, 689, 697, 698
- BEST-IN-GREEDY-ALGORITHMUS, 359–364, 376, 384, 450
- BFS, 30, 31, 33, 169, 372, 579
- BFS-Baum, 30
- Bildchensammler-Problem, 591
- BIN-PACKING-PROBLEM, 517, 518, 520–526, 529, 530
- binäre Darstellung, 6
- binäre Suche, 186, 390, 440
- binärer Clutter, 376

- binärer String, 410, 418
 bipartiter Graph, 38, 49
Bipartition, 37, 38, 52
Bipartitions-Ungleichung, 648
Birkhoff-von-Neumann-Satz, 296, 318
Bisektion, 565
Bit, 410
Blands Pivotregel, 64
Blatt, 20, 21
Block, 34, 51
 blockierender Clutter, 356, 357, 376
 blockierender Fluss, 195, 198, 217
 blockierendes Polyeder, 376
Blüte, 275, 277, 300, 301, 310
Blüte außerhalb des Waldes, 300
Blütenbasis, 275
Blütenlemma, 275, 278
Blütenwald, 277–279, 300, 310, 401
BLÜTENWEG, 301
Bogen, 16
BOHRPUNKTPROBLEM, 1
BOOLESCHE ÄQUIVALENZ, 445
 boolesche Formel, 422, 445, 446
 boolesche Variable, 422
Bottleneck-Funktion, 376, 385, 401
Bottleneck-Kante, 188
Bottleneck-Matching-Problem, 319
BRANCH-AND-BOUND, 652–654
Branch-and-Bound-Baum, 654
Branch-and-Cut, 654
Branching, 21, 149, 150, 164, 445, *siehe MAXIMUM-WEIGHT-BRANCHING-PROBLEM*
Branching-Greedoid, 383
Branching-Polytop, 163
brauchbare Funktion, 593, 594
BREADTH-FIRST-SEARCH, *siehe* BFS
Brücke, 20, 49
Brückenlemma, 589

CHINESISCHES POSTBOTEN-PROBLEM, 328, 444
GERICHTETES, 253
UNGERICHTETES, 328, 339
Cholesky-Faktorisierung, 106
chordaler Graph, 219, 491
CHRISTOFIDES' ALGORITHMUS, 616, 625, 651, 656
chromatische Zahl, 462, 463

chromatischer Index, 462
Church'sche These, 412
Chvátal-Rang, 131
Clique, 18, *siehe* MAXIMUM-CLIQUE-PROBLEM, *siehe* MAXIMUM-WEIGHT-CLIQUE-PROBLEM
CLIQUE, 427, 444
Cliquen-Polytop, 492
Cliquenbaum-Ungleichung, 648, 654, 658
Clutter, 356, 357, 376
CONCURRENT-FLOW-PROBLEM, 544, 548, 549, 564
coNP, 433
coNP-vollständig, 433, 434
critical path method (CPM), 215
Cut-Cancelling-Algorithmus, 253
Cutting-Stock-Problem, 517

Dekompositionssatz für Polyeder, 77
Delaunay-Triangulierung, 161, 640
DEPTH-FIRST-SEARCH, *siehe* DFS
Derandomisierung, 474, 635
Determinante, 87, 88
DFS, 30, 32, 33, 286
DFS-Baum, 30
DFS-Wald, 32
dichter Graph, 29
Digraph, 15
DIJKSTRAS ALGORITHMUS, 31, 169, 170, 174, 179, 180, 238, 543, 614
Dimension, 59
DINIC' ALGORITHMUS, 195, 198, 217
DISJUNKTE-WEGE-PROBLEM, 192, *siehe KANTENDISJUNKTE-WEGE-PROBLEM*, *siehe* KNOTENDISJUNKTE-WEGE-PROBLEM
Distanz, 19
Distanzkriterium, 539–541, 556, 564
Distanzmarkierung, 200
divide and conquer, 10, 313, 552
Dominanzrelation, 506
DOMINATING-SET, 444
DOPPELBAUM-ALGORITHMUS, 625, 656
doppelt-stochastische Matrix, 296
doppelt-stochastische Matrix, 287, 296
Drehkreuz, 611
Dreiecksungleichung, 181, 578, 614, 624, 655

- DREYFUS-WAGNER-ALGORITHMUS, 577, 614
- Dual
abstraktes, 49, 54, 356
planares, 47, 49, 339, 355
- DUAL-FITTING-ALGORITHMUS, 670, 673, 675
- duales LP, 70
- duales Unabhängigkeitssystem, 354
- Dualitätssatz, 70, 73
- dünner Graph, 29
- Durchmesser, 550
- durchschnittliche Laufzeit, 7, 65
- dynamic tree, 198, 199
- Dynamische Optimierung, 168, 176, 503, 577, 655
- dynamischer Fluss, 249
- echte Ohrenzerlegung, 34
- Ecke, 16
- Ecke eines Polyeders, 59, 61, 75, 77
- EDMONDS' BRANCHING-ALGORITHMUS, 151, 152
- EDMONDS' KARDINALITÄTS-MATCHING-ALGORITHMUS, 277, 280–283, 298
- EDMONDS' MATROID-INTERSEKTIONSALGORITHMUS, 366, 368, 369
- EDMONDS-KARP-ALGORITHMUS, 193–195, 261
- Edmonds-Rado-Satz, 361, 363, 364
- effizienter Algorithmus, 7
- einfache Jordankurve, 38
- einfacher Graph, 15
- EINFACHER MAXIMIERUNGSALGORITHMUS FÜR SUBMODULARE FUNKTIONEN, 399, 405
- einfaches b -Matching, 323
- Eingangsgrad, 17
- EINGESCHRÄNKTES HAMILTON-KREIS-PROBLEM, 641
- einseitiger Fehler, 420
- Einweg-Schnitt-Inzidenz-Matrix, 126, 127, 135
- elementarer Schritt, 6, 415, 416
- Ellipsoid, 91, 107
- ELLIPSOIDMETHODE, 81, 91–93, 98, 100, 390, 468, 525, 646
- Endknoten einer Kante, 16
- Endknoten eines Weges, 19
- endlich erzeugter Kegel, 62, 63, 73
- Endpunkte einer einfachen Jordankurve, 38
- Entfernung, 439
- entscheidbar in polynomieller Zeit, 411
- entscheidbare Sprache, 411
- Entscheidungsproblem, 417, 418
- Enumeration, 2, 652
- erbliche Grapheneigenschaft, 53
- erfüllbar, 422
- erfüllende Kantenmenge (Netzwerk-Design), 596
- erfüllte Klausel, 422
- erlaubte Kante (Netzwerkflüsse), 200
- erreichbar, 19
- erreichbares Mengensystem, 381, 382
- erweiterte Formulierung, 162, 342, 649, 657
- euklidische Kugel, 91
- euklidische Norm, 93
- EUKLIDISCHER ALGORITHMUS, 84, 85, 90
- EUKLIDISCHES STEINERBAUM-PROBLEM, 575, 656
- EUKLIDISCHES TSP, 628, 629, 632, 634, 635, 655
- EULERS ALGORITHMUS, 36, 37
- Eulersche Formel, 41, 42, 356
- euclischer Digraph, 35, 561
- euclischer Graph, 35, 36, 49, 328, 561, 624
- euclischer Spaziergang, 35, 328, 578, 624, 632
- exakter Algorithmus, 436
- EXAKTER KNAPSACK-ALGORITHMUS, 503–505, 529
- Expandergraph, 484, 564
- exponierter Knoten, 259
- Extrempunkt, 74, 77
- f -augmentierender Kreis, 231
- f -augmentierender Weg, 187, 189
- Facette, 60, 76
- facettenbestimmende Ungleichung, 60, 648
- faktorkritischer Graph, 266, 268–271
- Farbe, 462
- Farkas' Lemma, 73
- fast perfektes Matching, 266, 268

- fast-erfüllende Kantenmenge (Netzwerk-Design), 596
- Feedback-Kantenmenge, 357, 556
- Feedback-Knotenmenge, *siehe* MINIMUM-WEIGHT-FEEDBACK-VERTEX-SET-PROBLEM
- Feedback-Zahl, 556, 565
- FERNANDEZ-DE-LA-VEGA-LUEKER-ALGORITHMUS, 530, 531
- FF, *siehe* FIRST-FIT-ALGORITHMUS
- FFD, *siehe* FIRST-FIT-DECREASING-ALGORITHMUS
- Fibonacci-Heap, 146, 148, 152, 161, 170, 211
- Fibonacci-Zahl, 106
- FIRST-FIT-ALGORITHMUS, 521, 531
- FIRST-FIT-DECREASING-ALGORITHMUS, 521, 522, 531
- fixed-parameter tractable, 441
- FLOYD-WARSHALL-ALGORITHMUS, 174, 175, 181
- Fluss, 185, *siehe* MAXIMUM-FLOW-OVERTIME-PROBLEM, *siehe* MAXIMUM-FLOW-PROBLEM, *siehe* MINIMUM-COST-FLOW-PROBLEM
 b -, 227
blockierender, 195, 198, 217
 s - t -, 185, 189
- Flussdekompositionssatz, 189
- Flusserhaltungsregel, 185
- FORD-FULKERSON-ALGORITHMUS, 188, 189, 193, 213, 261, 615
- Fourier-Motzkin-Elimination, 77
- FPAS, FPTAS, *siehe* voll-polynomielles Approximationsschema
- FUJISHIGES ALGORITHMUS, 198, 199, 217
- Fundamentalkreis, 24, 50, 247
- Fundamentalschnitt, 24, 25, 336, 594
- Fünf-Farben-Satz, 468
- Funktion berechnen, 7
- Funktion polynomiell berechenbar, 7
- Gallai-Edmonds-Dekomposition, 283, 298, 302
- Gallai-Edmonds-Struktursatz, 283
- ganzzahlige Hülle, 110
- GANZZAHLIGE LINEARE UNGLEICHUNGEN, 418, 419, 433
- GANZZAHLIGE OPTIMIERUNG, 109, 111, 418, 439, 440, 525
- ganzzahliges Polyeder, 118, 119, 122, 445
- Ganzzahligkeitslücke, 112, 652, 697
- Ganzzahligkeitslucke, 592
- GAUSS-ELIMINATION, 64, 87–90, 97, 106
- Gebiet eines eingebetteten Graphen, 39–41
- GEBROCHENES b -MATCHING-PROBLEM, 253
- GEBROCHENES KNAPSACK-PROBLEM, 500, 502
- gebrochenes Matching-Polytop, 296
- gebrochenes Perfektes-Matching-Polytop, 296, 318, 320
- gegenläufige Kante, 187
- geglättete Analyse, 511
- gehörend zu der Ohrenzerlegung, 269, 310
- gemischt-ganzzahlige Optimierung, 131, 134
- gemischt-ganzzahliges Programm, 109
- gemischter Graph, 561, 566
- Gerichtete-Komponenten-LP, 586
- gerichteter Graph, *siehe* Digraph
- GERICHTETER HAMILTONSCHER WEG, 444
- gerichteter Schnitt, 22, 218, 555
- GERICHTETES CHINESISCHES POSTBOTEN-PROBLEM, 253
- GERICHTETES KANTENDISJUNKTE-WEGE-PROBLEM, 192, 216, 552–554, 556, 557, 565, 566
- GERICHTETES KNOTENDISJUNKTE-WEGE-PROBLEM, 192, 568
- GERICHTETES MAXIMUM-WEIGHT-CUT-PROBLEM, 491
- GERICHTETES MINIMUM-MEAN-CYCLE-PROBLEM, 176, 177, 182
- GERICHTETES MULTICOMMODITY-FLOW-PROBLEM, 537, 538
- GERICHTETES STEINERBAUM-PROBLEM, 614
- geschachtelte Familie, 25
- geschlossene Jordankurve, 38

- geschlossener alternierender Spaziergang, 638, 640
 geschlossener Spaziergang, 19
 Gewicht, 19, 437
 GEWICHTETER MATCHING-ALGORITHMUS, 304, 310, 312–314, 319, 328, 332, 338
 GEWICHTETER MATROID-INTERSEKTIONSALGORITHMUS, 371, 373, 378, 402
 gewichteter Median, 500, 501
 GEWICHTETER MEDIAN-ALGORITHMUS, 501
 GEWICHTETES MATROID-INTERSEKTIONSPROBLEM, 371, 373
 GEWICHTETES MEDIAN-PROBLEM, 500
 Gewinn eines alternierenden Spaziergangs, 638
 Gipfel (Netzwerk-Simplexalgorithmus), 247
 Gittergraph, 575
 GOEMANS-WILLIAMSON-ALGORITHMUS, 476, 477
 GOEMANS-WILLIAMSON-MAX-CUT-ALGORITHMUS, 460, 461
 Gomory-Chvátal-Stützung, 127, 320
 GOMORY-HU-ALGORITHMUS, 206, 210, 615
 Gomory-Hu-Baum, 205, 206, 210, 336, 594–596, 613, 615
 Gomory Schnitzebenenmethode, 128
 Grad, 17
 Graph, 9, 15
 - abstrakter dualer, 49, 54, 356
 - einfacher, 15
 - gemischter, 561, 566
 - gerichteter, *siehe* Digraph
 - planarer, 39, 46, 47, 54, 355, 356
 - planarer dualer, 47, 49, 339, 355, 555, 557
 - ungerichteter, 15- GRAPH-SCANNING-ALGORITHMUS, 28, 29
 graphisches Matroid, 348, 356, 362, 375
 Greedoid, 381–385, 402
 Greedy-Algorithmus, 144, 359, 384, 450, 488, 490, 513, 518, 532, 565, 665, 698
 GREEDY-ALGORITHMUS FÜR GREEDOIDE, 384, 401
- GREEDY-ALGORITHMUS FÜR SET-COVER, 450
 GREEDY-ALGORITHMUS FÜR VERTEX-COVER, 452
 Greedy-Augmentierung, 673, 674, 698
 GREEDY-FÄRBUNGSALGORITHMUS, 465, 491
 Griff, 648
 größter gemeinsamer Teiler, 84
 GRÖTSCHEL-LOVÁSZ-SCHRIJVER-ALGORITHMUS, 101, 103, 528, 529, 556
 gute Charakterisierung, 267, 434
 guter Algorithmus, 7
 guter alternierender Spaziergang, 638, 640
 guter geschlossener alternierender Spaziergang, 638
 Halb-Ellipsoid, 91, 94
 halbganzzahlige Lösung, 254, 319, 320, 490, 605
 Hall-Bedingung, 260
 HALTEPROBLEM, 441
 Hamilton-Kreis, 19, 359, 376
 HAMILTON-KREIS, 418, 419, 427
 hamiltonscher Graph, 19, 49
 HAMILTONSCHER WEG, 444
 - GERICHTETER, 444
- hamiltonscher Weg, 19
 harmonische Zahl, 591
 Heap, 146
 Heap-Ordnung, 147
 Heiratssatz, 261
 Held-Karp-Schranke, 650–653
 hermitesche Normalform, 117
 Heuristik, 436
 Hilbertbasis, 112, 134
 HITCHCOCK-PROBLEM, 228, 229, 687, 693
 HOPCROFT-KARP-ALGORITHMUS, 262, 286
 Hypergraph, 25
 induzierter Teilgraph, 16
 inklusionsminimal, 18
 innere Blüte, 277
 innere Knoten eines Weges, 19
 Innere-Punkte-Algorithmus, 81, 98, 460
 innerer Knoten, 276, 277

- Inputgröße, 6
- Instanz, 418, 436
- intern disjunkte Wege, 191
- Intervall-Packungsproblem, 136, 254
- Intervalgraph, 491
- Intervalmatrix, 136
- Inverse einer Matrix, 88
- inzident, 16
- Inzidenzmatrix, 28, 125
- Inzidenzvektor, 75
- isolierter Knoten, 17
- isomorphe Graphen, 16
- ITERATIVER RANDOMISIERTER RUNDUNGSALGORITHMUS, 587, 591
- Ja-Instanz, 418
- JAIN-VAZIRANI-ALGORITHMUS, 668
- JAINS ALGORITHMUS, 609, 610
- JOB-ZUORDNUNGSPROBLEM, 2, 9, 133, 185, 227, 259
- JOHNSONS ALGORITHMUS, 474, 475, 477
- Jordanscher Kurvensatz, 39
- k*-Approximationsalgorithmus, 449
 - asymptotischer, 470
- k*-beschränkter Steinerbaum, 579
- k*-CENTER-PROBLEM, 489
- k*-fach kantenzusammenhängender Graph, 33, 191
 - stark, 216, 566
- k*-fach kantenzusammenhängender Teilgraph, 592, *siehe* MINIMUM-WEIGHT-*k*-EDGE-CONNECTED-SUBGRAPH-PROBLEM
- k*-fach zusammenhängender Graph, 33, 191
- k*-MEDIAN-PROBLEM, 676
 - METRISCHES, 680, 682
- k*-opt Tour, 636, 657
- k*-OPT-ALGORITHMUS, 636, 637
- k*-regulärer Graph, 17
- K*-SCHWERSTE TEILMENGE, 445
- k*-STANDORTPROBLEM, 676
 - METRISCHES, 676, 677, 680
- Kürzeste-Wege-Baum, 179
- $K_{3,3}$, 42, 46
- K_5 , 42, 46
- Kaktus-Repräsentation, 221
- Kamm-Ungleichung, 646, 648, 654
- Kante, 15
- kantenchromatische Zahl, 462, 463
- kantendisjunkt, 16
- KANTENDISJUNKTE-WEGE-PROBLEM, 539–543
- GERICHTETES, 192, 216, 552–554, 556, 557, 565, 566
- UNGERICHTETES, 192, 557, 558, 560, 562, 566–568
- Kantenfärbung, 462, 463
- KANTENFÄRBUNGS-PROBLEM, 462–464, 471
- Kantenfolge, 19, 172, 176
- Kantengraph, 19, 320
- Kantenüberdeckung, 18, 284, 288, *siehe* MINIMUM-WEIGHT-EDGE-COVER-PROBLEM
- Kantenzusammenhang, 34, 204, 211, 219
- Kapazität, 185, 187
- KAPAZITÄTS-SKALIERUNGS-ALGORITHMUS, 239
- KARDINALITÄTS-MATCHING-PROBLEM, 259, 261, 264, 283, 286
- KARMARKAR-KARP-ALGORITHMUS, 527–531, 533
- Karp-Reduktion, 422
- Kegel, 62
 - endlich erzeugter, 62, 63, 73
 - polyedrischer, 62, 63, 73, 112, 134
- Kern-Reduktion, 441
- Kette, 284
- KETTENBRUCH-ERWEITERUNG, 85, 86, 106
- Kind, 51
- Kind eines Knotens, 21
- klauenfrei, 320
- Klausel, 422
- KNAPSACK-ALGORITHMUS
 - EXAKTER, 503–505, 529
- KNAPSACK-APPROXIMATIONSSCHEMA, 505
- Knapsack-Cover-Ungleichungen, 513
- KNAPSACK-PROBLEM, 346, 499, 502–505, 510, 513, 514, 528, 529
 - GEBROCHENES, 499, 500, 502
- Knoten, 15, 276, 277
- knotendisjunkt, 16

- KNOTENDISJUNKTE-WEGE-PROBLEM, 567
 GERICHTETES, 192, 568
 UNGERICHTETES, 192, 562, 567, 568
- Knotenfärbung, 462, 465, 468
- KNOTENFÄRBUNGS-PROBLEM, 462, 465, 468
- Knotenüberdeckung, 18, 260, 385, *siehe*
 MINIMUM-VERTEX-COVER-PROBLEM,
siehe MINIMUM-WEIGHT-VERTEX-
 COVER-PROBLEM
- Knotenzusammenhang, 34, 213, 220
- Komet, 693
- komplementärer Schlupf, 71, 72, 603, 667
- komplementäres Entscheidungsproblem, 433
- Komplementgraph, 18
- Konfigurations-LP, 524
- Königsberg, 35
- konjunktive Normalform, 422
- Konnektor, 573
- konservative Gewichte, 167, 173, 328
- Kontraktion, 17
- konvexe Funktion, 403
- konvexe Hülle, 74, 75
- konvexe Menge, 74, 99
- Konvexitätskombination, 74
- Kosten, 19, 437
 reduzierte, 173
- KOU-MARKOWSKY-BERMAN-ALGORITHMUS, 578, 579
- Kozyklenbasis, 24
- Kozyklenraum, 24
- Kreis, 19, 48, 345, 353
 negativer, 173, 175
 ungerader, 38, 52
 ungerichteter, 19, 22, 24
- Kreisbasis, 24, 41, 52
- Kreismatroid, 348, 355, 360
- Kreisraum, 24
- kreuzend-submodulare Funktion, 403
- kreuzungsfreie Familie, 25–27, 126
- KRUSKALS ALGORITHMUS, 144, 153, 154, 161, 360, 363, 384
- Kunde (Standortproblem), 663
- KÜRZESTE-WEGE-PROBLEM, 167, 169, 170, 329, 331, 346, 544
- KÜRZESTE-WEGE-PROBLEM FÜR ALLE PAARE, 173–175, 331
- kürzester Weg, 19, 30, 167
- KÜRZESTER WEG, 444
- Kürzester-Wege-Baum, 172
- L-Reduktion, 483, 575, 626
- L-reduzierbar, 483
- ℓ_1 -Abstand, 1
- ℓ_∞ -Abstand, 1
- Lagrange-Dual, 132, 133, 137, 650
- Lagrange-Multiplikator, 132, 650
- Lagrange-Relaxierung, 132, 134, 136, 137, 514, 649, 650, 676
- laminare Familie, 25–27, 126, 300, 310, 606
- landmark, 181
- Länge (eines Strings), 410
- Länge (eines Weges oder Kreises), 19
- Las-Vegas-Algorithmus, 148, 420
- Laufzeit von Graphenalgorithmen, 29
- leerer Graph, 18
- leerer String, 410
- leichte Steinertour, 631, 632
- Leiterplatte, 1
- Level-Graph, 195
- lexikographisch sortieren, 13
- lexikographische Ordnung, 3, 13
- lexikographische Regel, 64
- LIN-KERNIGHAN-ALGORITHMUS, 639–641, 656
- linear reduzierbar, 142
- LINEARE OPTIMIERUNG, 57, 61, 63, 81, 97–99, 418, 435
- LINEARE UNGLEICHUNGEN, 418, 419, 434
- linearer Algorithmus, 4, 6
- linearer Graphenalgoritmus, 29
- lineares Gleichungssystem, 87
- lineares Programm, *siehe* LP
- lineares Ungleichungssystem, 72, 77, 97
- Linie, 16
- Literal, 422
- LOGIK-MINIMIERUNG, 446
- lokale Suche, 636, 642, 657, 680, 686
- lokaler Kantenzusammenhang, 204, 218
- lokales Optimum, 657
- Lösung eines Optimierungsproblems
 optimale, 436

- zulässige, 59, 436
- Lovász Theta-Funktion, 468
- Löwner-John-Ellipsoid, 91
- LP, 9, 57
 - duales, 70
 - primales, 70
- LP-Dualitätssatz, *siehe* Dualitätssatz
- LP-Relaxierung, 111, 133, 297, 454, 475, 490, 524, 527, 549, 593, 604, 642, 649, 653, 697
- M*-alternierende Ohrenzerlegung, 268–271
- M*-alternierender Weg, 262
- M*-augmentierender Weg, 262, 285
- m*-DIMENSIONALES KNAPSACK-PROBLEM, 507, 508, 514
- MA-Reihenfolge, 211, 219
- MANHATTAN-STEINERBAUM-PROBLEM, 575, 579, 580
- Matching, 10, 18, 260, 315, *siehe* KARDINALITÄTS-MATCHING-PROBLEM, *siehe* MAXIMUM-WEIGHT-MATCHING-PROBLEM
 - b*-, 323, 324
 - perfektes, 259, 314
- Matching-Polytop, 315
- Matrixnorm, 93
- Matroid, 347, 350–353, 361
- Matroid-Intersektion, *siehe* GEWICHTETES MATROID-INTERSEKTION-PROBLEM
- MATROID-INTERSEKTION-PROBLEM, 365, 369, 371
- MATROID-PARITÄTS-PROBLEM, 403
- MATROID-PARTITIONS-PROBLEM, 369, 371
- Matroid-Polytop, 154, 362, 386, 388
- MAX-2SAT, 437, 493
- MAX-3SAT, 478, 481, 484
- MAX-CUT, 455, *siehe* MAXIMUM-WEIGHT-CUT-PROBLEM
- Max-Flow-Min-Cut-Eigenschaft, 357, 376
- Max-Flow-Min-Cut-Quotient, 549, 552, 564
- Max-Flow-Min-Cut-Theorem, 189, 341, 358, 597
- MAX-*k*-COVER-PROBLEM, 698
- MAX-SAT, 473–478, *siehe* MAXIMUM-SATISFIABILITY
- MAX-SAT-PROBLEM
 - 3-OCCURRENCE-, 484, 486
- maximale Blüte, 300
- maximaler Teilgraph bez. einer Eigenschaft, 16
- MAXIMIERUNGSPROBLEM FÜR UNABHÄNGIGKEITSSYSTEME, 346, 359–361, 363, 506
- MAXIMIERUNGSPROBLEM SUBMODULÄRER FUNKTIONEN, 399
- MAXIMUM-CLIQUE-PROBLEM, 480, 481, 492
- MAXIMUM-CUT-PROBLEM, 455, 456, 490, 492
- MAXIMUM-FLOW-OVER-TIME-PROBLEM, 250
- MAXIMUM-FLOW-PROBLEM, 185, 186, 188, 193–195, 198–200, 203, 204, 556
- MAXIMUM-MATCHING-PROBLEM, 282
- MAXIMUM-MULTICOMMODITY-FLOW-PROBLEM, 544
- MAXIMUM-STABLE-SET-PROBLEM, 480, 481, 488
- MAXIMUM-WEIGHT-*b*-MATCHING-PROBLEM, 324, 325, 338, 342
- MAXIMUM-WEIGHT-BRANCHING-PROBLEM, 149, 151, 347
- MAXIMUM-WEIGHT-CLIQUE-PROBLEM, 468
- MAXIMUM-WEIGHT-CUT-PROBLEM, 339, 455, 456, 490
 - GERICHTETES, 491
- MAXIMUM-WEIGHT-FOREST-PROBLEM, 142, 347
- MAXIMUM-WEIGHT-MATCHING-PROBLEM, 293, 319, 347
- MAXIMUM-WEIGHT-STABLE-SET-PROBLEM, 346, 468
- MAXSNP, 484
- MAXSNP-schwer, 484, 486, 488, 493, 575, 614, 626, 628
- Median, *siehe* GEWICHTETER-MEDIAN-PROBLEM
 - gewichteter, 500, 501
- Mehrgüterfluss-Relaxierung, 541, 563
- Mengensystem, 25
- MERGE-SORT-ALGORITHMUS, 10–12

- Methode der bedingten Wahrscheinlichkeiten, 474
- metrischer Abschluss, 174, 181, 578
- METRISCHES BESCHRÄNKTES STANDORTPROBLEM, 687, 689, 697, 698
- METRISCHES BIPARTITES TSP, 656
- METRISCHES k -MEDIAN-PROBLEM, 680, 682
- METRISCHES k -STANDORTPROBLEM, 676, 677, 680
- METRISCHES SCHWACH-BESCHRÄNKTES STANDORTPROBLEM, 687, 688, 698
- METRISCHES TSP, 624–626, 628, 642, 651
- METRISCHES UNBESCHRÄNKTES STANDORTPROBLEM, 664, 667, 668, 673, 684, 687
- minimale Seitenfläche, 61
- minimaler s - t -Schnitt, 187
- minimaler Teilgraph bez. einer Eigenschaft, 16
- MINIMIERUNGSPROBLEM FÜR UNABHÄNGIGKEITSSYSTEME, 346, 357, 360, 362, 363
- MINIMIERUNGSPROBLEM SUBMODULÄRER FUNKTIONEN, 390–393, 396
- MINIMUM-CAPACITY-CUT-PROBLEM, 204, 212, 443
- MINIMUM-CAPACITY- T -CUT-PROBLEM, 336, 342
- MINIMUM-COST-FLOW-PROBLEM, 228, 229, 231, 232, 234, 237, 239, 240, 242, 244, 245, 248, 252–255, 404
- MINIMUM-MEAN-CYCLE-ALGORITHMUS, 177, 178, 182
- MINIMUM-MEAN-CYCLE-CANCELLING-ALGORITHMUS, 232, 234, 236
- MINIMUM-MEAN-CYCLE-PROBLEM UNGERICHTETES, 340, 341
- MINIMUM-SET-COVER-PROBLEM, 450, 462
- MINIMUM-SPANNING-TREE-PROBLEM, 142–145, 148, 152, 161, 347, 578, 625, 650
- MINIMUM-VERTEX-COVER-PROBLEM, 452–454, 480, 488–490, 493, 626
- MINIMUM-WEIGHT-ARBORESCENCE-PROBLEM, 149
- MINIMUM-WEIGHT-EDGE-COVER-PROBLEM, 319, 452
- MINIMUM-WEIGHT-FEEDBACK-VERTEX-SET-PROBLEM, 490
- MINIMUM-WEIGHT- k -EDGE-CONNECTED-SUBGRAPH-PROBLEM, 615
- MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM, 293, 294, 296, 297, 304, 313, 329, 330
- MINIMUM-WEIGHT-ROOTED-ARBORESCENCE-PROBLEM, 149, 154, 155, 161, 216
- MINIMUM-WEIGHT-SET-COVER-PROBLEM, 450, 454
- MINIMUM-WEIGHT- T -JOIN-PROBLEM, 328–331, 334, 335, 339, 616
- MINIMUM-WEIGHT-VERTEX-COVER-PROBLEM, 450, 455, 490
- Minor, 42, 53
- modulare Funktion, 18, 19, 345, 385, 389
- monotone Mengenfunktion, 386
- Monte-Carlo-Algorithmus, 220, 420
- MOORE-BELLMAN-FORD-ALGORITHMUS, 171, 173, 237
- MULTICOMMODITY-FLOW-APPROXIMATIONSSCHEMA, 545
- MULTICOMMODITY-FLOW-PROBLEM, 538–540, 543, 563
- GERICHTETES, 537, 538
- MAXIMUM-, 544
- UNGERICHTETES, 528, 538, 567
- UNIFORMES, 552
- MULTIDIMENSIONALES KNAPSACK-PROBLEM, 507
- Multigraph, 16
- MULTIPROCESSOR-SCHEDULING-PROBLEM, 532, 533
- Multischnitt, 157, 162
- nach rechts oben unbeschränktes Polymer, 376
- Nachbar, 15, 17
- Nachfolger, 51
- Nachfrage, 227
- Nachfragekante, 192, 537
- Nearest-Neighbour-Heuristik, 624
- negativer Kreis, 173, 175
- Nein-Instanz, 418

- NEMHAUSER-ULLMANN-ALGORITHMUS, 510–512
- Netzwerk, 185
- NETZWERK-SIMPLEXALGORITHMUS, 245, 248
- Netzwerkmatrix, 127, 135
- NEXT-FIT-ALGORITHMUS, 520
- NF, *siehe* NEXT-FIT-ALGORITHMUS
- nichtdeterministischer Algorithmus, 420
- nichtnegative Gewichte, 437
- nichtsaturierender Push, 202
- NP, 419, 420, 443, 479
- NP-äquivalent, 436
- NP-leicht, 436
- NP-Optimierungsproblem, 436
- NP-schwer, 436
 - stark, 439, 507
- NP-vollständig, 422, 423, 425
 - stark, 439
- O-Notation, 4
- oblivious routing, 611
- Ohr, 34
- Ohrenzerlegung, 34, 51
 - alternierende, 268–271
 - echte, 34
 - gehörend zu der, 269, 310
 - M-alternierende, 268–271
 - ungerade, 268, 288
- OHRENZERLEGUNGSALGORITHMUS, 269, 270
- Ω-notation, 4
- Online-Algorithmus, 523
- optimale Basislösung, 97, 98
- optimale Lösung eines LP, 57
- optimale Lösung eines Optimierungsproblems, 436
- OPTIMALES LINEAR-ARRANGEMENT-PROBLEM, 107, 565
- Optimierung mit mehreren Zielfunktionen, 509
- Optimierungsproblem, 436
- Orakel, 100, 346, 359, 377, 390, 393, 399, 594, 598
- Orakel-Algorithmus, 99, 421
- Orakel-Turingmaschine, 417
- Orientierung, 16, 561, 566
- ORLINS ALGORITHMUS, 241, 242, 244
- P, 418
- Padberg-Rao-Satz, 337
- parallele Kanten, 15
- Pareto-optimal, 509
- partiell geordnete Menge, 284
- PARTITION, 433, 439
- partitionierbar, 369, 377
- Patching-Lemma, 631, 633
- PCP, 479
- PCP($\log n, 1$), 478, 479
- PCP-Satz, 479
- perfekter Graph, 465, 466, 491
- perfektes b -Matching, 323, 327
- perfektes einfaches 2-Matching, 323, 339, 464, 654
- perfektes Matching, 259, 261, 263, 266, 267, 314, *siehe* MINIMUM-WEIGHT-PERFECT-MATCHING-PROBLEM
- Perfektes-Matching-Polytop, 314, 318, 320, 336
- Permanente einer Matrix, 286
- Permutation, 1, 3, 83
- Permutationsmatrix, 287, 296
- Petersen-Graph, 605
- PIVOT, 691
- Pivotregel, 64
- planare Einbettung, 39, 47, 53
- planarer Graph, 39, 46, 47, 54, 355, 356
- planares Dual, 47, 49, 339, 355
- platonischer Graph, 52
- platonischer Körper, 52
- PLS, 657
- polare Menge, 104, 107
- Polyeder, 9, 59
 - beschränktes, 97
 - ganzzahliges, 118, 119, 122, 445
 - nach rechts oben unbeschränktes, 376
 - rationales, 59
 - volldimensionales, 59, 97
- polyedrische Kombinatorik, 75
- polyedrischer Kegel, 62, 63, 73, 112, 134
- Polygon, 38, 39, 628
- polygonaler Streckenzug, 38, 39
- Polymatroid, 386, 390, 391, 402
- POLYMATROID-GREEDY-ALGORITHMUS, 386, 390, 391, 402
- POLYMATROID-MATCHING-PROBLEM, 403
- Polymatroidschnitt-Satz, 387

- polynomiell äquivalente Orakel, 359, 377
 polynomiell äquivalente Probleme, 436
 polynomiell berechenbare Funktion, 7
 polynomielle Reduktion, 421, 422, 437
 polynomielle Transformation, 422
 polynomielle Turingmaschine, 411, 416
 polynomieller Algorithmus, 6, 7, 412,
 416
 schwach, 6
 streu, 6, 99
 Polytop, 59, 74, 75
 Portal (EUKLIDISCHES TSP), 631
 positiv semidefinite Matrix, 106
 Potenzial zugehörig zu einer Baumstruktur, 246
 Potenzmenge, 18
 Präfluss
 s - t -, 196, 217
 s - t -maximaler, 217
 Praprozessierung, 441
 PRIM, 435
 primal-dualer Algorithmus, 297, 371, 544,
 599, 667
 PRIMAL-DUALER ALGORITHMUS FÜR
 NETZWERK-DESIGN, 598, 599, 604,
 616
 primales LP, 70
 PRIMS ALGORITHMUS, 145, 146, 148,
 161, 179, 384, 579
 Prioritätswarteschlange, 146
 prize-collecting Steiner forest problem,
 617
 probabilistische Methode, 473
 program evaluation and review technique
 (PERT), 215
 pseudopolynomieller Algorithmus, 438,
 439, 503, 504, 507, 518, 531
 PTAS, *siehe* Approximationsschema
 PUNKT-ZU-PUNKT-VERBINDUNGSPROBLEM,
 617
 Push
 nichtsaturierender, 202
 saturierender, 202
 PUSH, 200, 202
 PUSH-RELABEL-ALGORITHMUS, 200–
 203, 218, 254
 QUADRATISCHES ZUORDNUNGSPROBLEM,
 531
 Quelle, 185, 227
 r -Schnitt, 22
 radix sorting, 13
 RAM-Maschine, 416, 442
 randomisierter Algorithmus, 148, 220,
 264, 420, 473, 475
 Rang einer Matrix, 59, 87, 88
 Rangfunktion, 346, 351
 untere, 349
 Rang-Orakel, 360
 Rangquotient, 349, 375
 rationales Polyeder, 59
 realisierbare Nachfragekante, 541
 realisierender Weg, 192
 reduzierte Kosten, 173
 Region (EUKLIDISCHES TSP), 630
 regulärer Ausdruck, 8
 rekursiver Algorithmus, 10
 RELABEL, 200
 relative Approximationsgüte, 449
 Relaxierung
 Lagrange-, 132, 134, 136, 137, 514,
 649, 650, 676
 LP-, 111, 133, 297, 454, 475, 490, 524,
 527, 549, 593, 604, 642, 649, 653,
 697
 Mehrgüterfluss-, 541, 563
 Repräsentantensystem, 285
 repräsentierbares Matroid, 348, 375
 Residualgraph, 187
 Residualkapazität, 187
 Restriktion eines Problems, 439
 revidierte Simplexmethode, 70
 ROBINS-ZELIKOVSKY-ALGORITHMUS, 583,
 586
 robustes Network-Design, 611
 S-L Bäume, 178
 s - t -Fluss, 185, 189
 s - t -Präfluss, 196, 217
 maximaler, 217
 s - t -Schnitt, 22, 187, 189, 214, 597
 minimaler, 187
 s - t -Weg, 360
 s - t -zeitabhängiger Fluss, 249
 Sammler-Steinerwald-Problem, 617
 SATISFIABILITY, 423
 saturierender Push, 202

- Satz über ganzzahlige Flüsse, 189
 Satz von Berge, 262, 274, 275, 285
 Satz von Carathéodory, 77
 Satz von Cayley, 143, 160
 Satz von Dilworth, 284
 Satz von Hall, 260, 261
 Satz von Hoffman und Kruskal, 122, 124, 296
 Satz von Khachiyan, 97, 98
 Satz von König, 135, 260, 284, 318
 Satz von Kuratowski, 43, 44, 46, 52
 Satz von Lucchesi und Younger, 555, 556
 Satz von Menger, 190–192, 215, 219, 260, 341, 542, 613
 Satz von Okamura und Seymour, 558, 566
 Satz von Padberg und Rao, 336, 337
 Satz von Tutte, 266, 267, 287
 Satz von Vizing, 463, 464, 470
 Scheduling-Problem, 533
 schiefsymmetrisch, 263
 Schleife, 16, 47, 49
 Schlüssel, 146
 schnelle Matrizenmultiplikation, 174
 Schnellstes-Transshipment-Problem, 251
 Schnitt, 22, 48, *siehe* MAXIMUM-WEIGHT-CUT-PROBLEM, *siehe* MINIMUM-CAPACITY-CUT-PROBLEM
 gerichteter, 22, 218, 555
 r -, 22
 $s-t$ -, 22, 187, 189, 214, 597
 minimaler, 187
 T -, 332, 342
 ungerichteter, 22, 24
 Schnitt von Matroiden, 364
 Schnitt von Unabhängigkeitssystemen, 364
 Schnitt-Inzidenz-Matrix
 Einweg-, 126, 127, 135
 Zweiwege-, 126, 127, 135
 Schnitt-Semimetrik, 491
 Schnittebenenmethode, 127, 131, 654
 Schnittkriterium, 540, 541, 554, 557, 558, 562–564, 566, 567
 SCHRIJVERS ALGORITHMUS, 393, 395, 404
 Schrumpfung, 17
 schwach polynomieller Algorithmus, 6
 schwach supermodulare Funktion, 593, 605, 607
 schwacher Dualitätssatz, 64
 SCHWACHES OPTIMIERUNGSPROBLEM, 100, 101, 103, 528
 schwaches Separations-Orakel, 100
 SCHWACHES SEPARATIONS-PROBLEM, 100, 101, 528, 529
 Schwellenrundung, 617
 seichte leichte Bäume, 180
 Seitenfläche eines Polyeders, 59, 60
 semidefinite Optimierung, 81, 457, 460, 552
 Senke, 185, 227
 SEPARATIONS-PROBLEM, 99, 103, 105, 336, 337, 390, 391, 528, 543, 544, 556, 594, 595, 610, 646, 648
 Separator, 313
 serienparalleler Graph, 52
 Servicekosten, 665
 SET-PACKING-PROBLEM, 492
 shallow-light trees, 180
 SHMOYS-TARDOS-AARDAL-ALGORITHMUS, 666
 Signum einer Permutation, 83
 SIMPLEXALGORITHMUS, 63–66, 71, 81, 87, 98, 245, 525, 543, 642
 Simplextableau, 68
 simpliziale Reihenfolge, 219
 singleton, 17
 Skalarprodukt, 57
 Skalierungstechnik, 199, 239, 254
 smoothed analysis, 65, 511
 SOFT-CAPACITATED STANDORTPROBLEM, 698
 sortieren, 10, 12, 13
 lexikographisch, 13
 Spaltenerzeugung, 70
 Spaltenerzeugungsmethode, 528, 544
 SPARSEST-CUT-PROBLEM, 548, 552
 ALLGEMEINES, 548
 Spaziergang, 19
 geschlossener, 19
 Sperner's Lemma, 285
 spezieller Blütenwald, 277–279, 401
 spitzes Polyeder, 61
 Sprache, 410, 417

- stabile Menge, 18, 284, *siehe* MAXIMUM-STABLE-SET-PROBLEM, *siehe* MAXIMUM-WEIGHT-STABLE-SET-PROBLEM
- Stabile-Mengen-Polytop, 466
- STABLE-SET, 426, 444
- Stadt, 439
- Standardeinbettung, 47
- Standort (Standortproblem), 663
- Standortproblem, 136, 663, *siehe* (METRISCHES) UNBESCHRÄNKTES STANDORTPROBLEM, *siehe* METRISCHES k -STANDORTPROBLEM, *siehe* METRISCHES (SCHWACH) BESCHRÄNKTES STANDORTPROBLEM, *siehe* SOFT-CAPACITATED STANDORTPROBLEM, *siehe* UNIVERSELLES STANDORTPROBLEM
- Standortprobleme
- beschränkte, 686
 - stark k -fach kantenzusammenhängender Graph, 216, 566
 - stark NP -schwer, 439, 507
 - stark NP -vollständig, 439
 - stark zulässige Baumstruktur, 246
 - stark zusammenhängender Digraph, 23, 51, 54, 556, 565, 566
 - starke Zusammenhangskomponente, 23, 31–33
- Steiner-Quotient, 580
- STEINERWALD-PROBLEM, 610
- Steinerbaum, 573, 578, *siehe* STEINER-BAUM-PROBLEM
- STEINERBAUM-PROBLEM, 347, 574–579, 583, 586, 587, 591, 614
- EUKLIDISCHES, 575
 - MANHATTAN-, 575, 579, 580
- Steinerpunkt, 573, 614
- Steinertour, 631
- leichte, 631, 632
- STEINERWALD-PROBLEM, 592, 595, 617
- Stern, 20, 692
- Stirlingformel, 3
- straffe Kante (gewichtetes Matching), 298, 310
- straffe Kantenmenge (Netzwerk-Design), 605
- streng polynomieller Algorithmus, 6, 99
- String, 410
- Strong-Perfect-Graph-Satz, 466
- Stufe (EUKLIDISCHES TSP), 630
- stützende Hyperebene, 59
- Subgradienten-Verfahren, 132, 650
- submodulare Funktion, 18, 213, 350, 385, 387, 389–391, 403
- Maximierung, 404
- submodularer Fluss, 403, 404
- SUBMODULARER-FLUSS-PROBLEM, 403, 404
- SUBSET-SUM, 432, 438
- Subtour-Ungleichung, 645, 646, 654
- Subtouren-Polytop, 645, 651
- SUKZESSIVE-KÜRZESTE-WEGE-ALGORITHMUS, 237, 238
- SUKZESSIVE-KÜRZESTE-WEGE-ALGORITHMUS, 237, 294
- Summe von Matroiden, 369
- supermodulare Funktion, 18, 389
- schwach, 593, 605, 607
- SURVIVABLE-NETWORK-DESIGN-PROBLEM, 573, 592, 599, 604, 610, 615, 616, 618
- symmetrische submodulare Funktion, 396, 398
- T*-Join, 328, 329, 332, 341, 358, *siehe* MINIMUM-WEIGHT-*T*-JOIN-PROBLEM
- T*-Join-Polyeder, 335
- T*-Schnitt, 332, 342, *siehe* MINIMUM-CAPACITY-*T*-CUT-PROBLEM
- Tagebau, 218
- Taille, 42, 331
- TDI, 119–121, 123, 124, 135, 154, 318, 387, 404, 492, 567
- teilerfremd, 82
- Teilgraph, 16
- aufspannender, 16
 - induzierter, 16
 - k -fach kantenzusammenhängender, 592
 - maximal bez. einer Eigenschaft, 16
 - minimal bez. einer Eigenschaft, 16
- Teilgraph-Grad-Polytop, 339
- Terminal (DISJUNKTE-WEGE-PROBLEM), 192
- Terminal (Mehrgüterflüsse), 537
- Terminal (Steinerbaum), 573
- Testmenge, 114
- Theta-Funktion, 468

- Θ-Notation, 4
 topologische Ordnung, 23, 33, 597
 totally dual integral, *siehe* TDI
 Tour, 645
 Transportproblem, 228
 Transshipment-Problem, 228
 Transversale, 375, 377
 Traveling-Salesman-Polytop, 643, 649
TRAVELING-SALESMAN-PROBLEM (TSP),
 346, 439, 623, 636, 639, 641, 642,
 650, 653
 trennende Hyperebene, 99
 trennende Kantenmenge, 22
 Trennungs-Orakel, 99, 104
TSP, *siehe* TRAVELING-SALESMAN-PROBLEM
TSP-FACETTEN, 649
 Turing-Reduktion, 422
 Turingmaschine, 409, 410, 412, 413, 415
 Turnier, 51
 Tutte-Bedingung, 265–267
 Tutte-Matrix, 262
 Tutte-Menge, 267
 überdeckter Knoten, 259
 Überdeckung, 450, *siehe* MINIMUM-SET-COVER-PROBLEM, *siehe* MINIMUM-WEIGHT-SET-COVER-PROBLEM
 Überschuss, 185
 unabhängige Menge, 345
 Unabhängigkeits-Orakel, 359, 373
 Unabhängigkeitssystem, 345, *siehe* MAXIMIERUNGSPROBLEM FÜR UNABHÄNGIGKEITSSYSTEME
 duales, 354
 unbeschränktes Gebiet, 40
 unbeschränktes LP, 57, 72, 73
UNBESCHRÄNKTES STANDORTPROBLEM,
 137, 665, 666, 668, 670, 676
METRISCHES, 664, 667, 668, 673,
 684, 687
 unentscheidbares Problem, 441
 Ungarische Methode, 294, 295, 319
 ungerade Kreisüberdeckung, 339
 ungerade Ohrenzerlegung, 268, 288
 ungerade Überdeckung, 37
 ungerade Verbindung, 37
 ungerader Kreis, 38, 52
 ungerichteter Graph, 15
 ungerichteter Kreis, 19, 22, 24
 ungerichteter Schnitt, 22, 24
 ungerichteter Weg, 19
UNGERICHTETES CHINESISCHES POSTBOTEN-PROBLEM, 328, 339
UNGERICHTETES KANTENDISJUNKTE-WEGE-PROBLEM, 192, 557, 558, 560, 562, 566–568
UNGERICHTETES KNOTENDISJUNKTE-WEGE-PROBLEM, 192, 562, 567, 568
UNGERICHTETES MINIMUM-MEAN-CYCLE-PROBLEM, 340, 341
UNGERICHTETES MULTICOMMODITY-FLOW-PROBLEM, 528, 538, 567
 uniformes Matroid, 348, 375
UNIFORMES MULTICOMMODITY-FLOW-PROBLEM, 552
 unimodulare Matrix, 116, 117, 134
 unimodulare Transformation, 116, 134
UNIVERSELLES STANDORTPROBLEM, 687, 696
 Unterdeterminante, 112
 untere Rangfunktion, 349
 Unterteilung, 52, 53
 unzulässiges LP, 57, 72, 73
 unzusammenhängend, 20
 UPDATE, 304
 Vektormatroid, 348
 verbotener Minor, 53
 Vereinigung von Matroiden, 369
 verletzte Knotenmenge (Netzwerk-Design), 595
 Verlust eines Steinerbaumes, 581
 verschobenes Gitter (EUKLIDISCHES TSP), 630
VERTEX-COVER, 427, 441
 Vierfarbenproblem, 468
 Vierfarbensatz, 468, 469
 VLSI-Design, 76, 182, 575
 voll-polynomielles Approximationsschema, 471, 505–507, 544, 545
 voll-polynomielles asymptotisches Approximationsschema, 471, 527, 530
 volldimensionales Polyeder, 59, 97
 volle Komponente eines Steinerbaumes, 579
 voller Steinerbaum, 579

- vollständig-unimodulare Matrix, 122–127, 189
- vollständige duale Ganzzahligkeit, *siehe* TDI
- vollständiger Graph, 18
- Vorfahre, 51
- Vorgänger, 21, 51
- Voronoi-Diagramm, 161
- VPN-PROBLEM, 611, 613
- Wachstumsrate, 4
- Wahrheitsbelegung, 422
- Wald, 20, 159, *siehe* MAXIMUM-WEIGHT-FOREST-PROBLEM
- Wald-Polytop, 162
- Weak-Perfect-Graph-Satz, 466
- Weg, 19
 - ungerichteter, 19
- WEG-ENUMERATIONS-ALGORITHMUS, 3, 5
- Wert eines s - t -Flusses, 185
- wohlgerundete Instanz des EUKLIDISCHEN TSP, 629
- Worst-Case-Laufzeit, 7
- WORST-OUT-GREEDY-ALGORITHMUS, 160, 359, 360, 362, 363
- Wort, 410
- Wurzel, 21, 51, 276
- Zeit-Kosten-Tradeoff-Problem, 215
- zeitabhängiger Fluss, 249
 - s - t -, 249
- zeiterweitertes Netzwerk, 255
- Zertifikat, 419, 434
- Zertifikat-Prüfalgorithmus, 419, 478, 479
- Zeugenbaum, 588
- Zinke, 648
- Zirkulation, 185, 230
- Zirkulationssatz von Hoffman, 214
- zufälliges Runden, 475, 564
- zugrunde liegender ungerichteter Graph, 16
- zulässige Baumstruktur, 246
- zulässige Lösung eines LP, 57
- zulässige Lösung eines Optimierungsproblems, 59, 436
- zulässiges Potenzial, 173, 231
- ZUORDNUNGSPROBLEM, 294, 295, 319
 - QUADRATISCHEs, 531
- zusammenhängende Knotenmenge, 20
- zusammenhängender Digraph, 20
- zusammenhängender ungerichteter Graph, 20
- zusammenhängendes Gebiet, 38, 39
- Zusammenhangsbedingungen, 592
- Zusammenhangskomponente, 20, 29, 145
 - starke, 23, 31–33
- 2-Band-Turingmaschine, 412, 413, 415
- ZWEITER HAMILTON-KREIS, 656, 657
- Zweiwege-Schnitt-Inzidenz-Matrix, 126, 127, 135
- Zykluszeit, 182



Willkommen zu den Springer Alerts

Jetzt
anmelden!

- Unser Neuerscheinungs-Service für Sie:
aktuell *** kostenlos *** passgenau *** flexibel

Springer veröffentlicht mehr als 5.500 wissenschaftliche Bücher jährlich in gedruckter Form. Mehr als 2.200 englischsprachige Zeitschriften und mehr als 120.000 eBooks und Referenzwerke sind auf unserer Online Plattform SpringerLink verfügbar. Seit seiner Gründung 1842 arbeitet Springer weltweit mit den hervorragendsten und anerkanntesten Wissenschaftlern zusammen, eine Partnerschaft, die auf Offenheit und gegenseitigem Vertrauen beruht.

Die SpringerAlerts sind der beste Weg, um über Neuentwicklungen im eigenen Fachgebiet auf dem Laufenden zu sein. Sie sind der/die Erste, der/die über neu erschienene Bücher informiert ist oder das Inhaltsverzeichnis des neuesten Zeitschriftenheftes erhält. Unser Service ist kostenlos, schnell und vor allem flexibel. Passen Sie die SpringerAlerts genau an Ihre Interessen und Ihren Bedarf an, um nur diejenigen Information zu erhalten, die Sie wirklich benötigen.

Mehr Infos unter: springer.com/alert