Combinatorial Optimization 1

Lecture notes, University of Technology Graz

Lukas Prokop

March 11, 2015

Contents

1 Course organization **4**

2 Introduction **6**

2.1 A generic combinatorial optimization problem 6

2.2 Possible common cost models 6

3 Problem 1: Drill machine problem **6**

4 Problem 2: Scheduling problem **7**

5 Partial enumeration **7**

5.1 Working principle 8

5.2 Algorithm efficieny 8

6 Analysis of algorithms **9**

7 Spanning trees and arborescences **10**

7.1 Minimal spanning tree problem (MST) 10

7.2 Maximum weight forest problem (MWF) 10

7.3 Equivalence of problems 10

7.4 MST and MWF are equivalent 10

7.4.1 Reduce MWF to MST 11

7.4.2 Reduce MST to MWF 11

7.4.3 $a \Rightarrow b$ 12
7.4.4 $b \Rightarrow c$ 12
7.4.5 $c \Rightarrow d$ 13
7.4.6 $d \Rightarrow a$ 13
7.5 Kruskal's algorithm 13
7.6 Prim's algorithm 16

8 Number of spanning trees **17**
8.1 Minimum Weight Arborescence Problem (MWA) 18
8.2 Minimum Weighted Rooted Arborescence Problem (MWRA) . . 18
8.3 Maximum Weighted Branching Problem (MWB) 18
8.4 Equivalence of MWA, MWRA and MWB 19
8.5 Edmonds' branching algorithm 20

9 Shortest path problems in graphs **22**
9.1 Single source shortest path problems (SSSP) 22
9.2 Dijkstra's algorithm for SSSP 23
9.2.1 Analysis 24
9.3 Moore-Bellman-Ford algorithm 25
9.3.1 Analysis 25

10 Potential **27**

11 All Pairs Shortest Paths Problem **28**
11.1 All pairs shortest paths problems (APSP) 28
11.2 Floyd-Warshall algorithm 29

12 Cycles with minimal mean edge weight **30**
12.1 Minimal mean-cycle problem (MMC) 30
12.2 Algorithm for minimal mean cycle problem 32

13 Network flows **34**
13.1 Definition 34
13.2 Maximum flow problem (MF) 35
13.3 Example: Job assignment problem 35

13.4 Maximum-flow problem (cont.) 36
13.5 Algorithm by Ford & Fulkerson 41
13.5.1 Analysis 42
13.6 Edmonds and Karp algorithm 44
13.6.1 Runtime analysis 48
13.7 Blocking flows and Dinitz's algorithm (1970) 49
13.8 Goldberg & Tarjan: Push-Relabel algorithm 52
13.9 Minimum-capacity cut problem 59
13.10 Gomory-Hu algorithm 61
13.11 Minimal capacity of a cut in an undirected graph / MA-order . . 64

14 Flows with minimum costs **69**
14.1 Minimum cost flow problem (MCFP/MKFP) 69
14.2 The transportation problem 71
14.3 An optimality criterion 72
14.4 Minimum-mean cycle cancelling algorithm 76
14.4.1 Analysis of MMCC 78

15 Successive shortest path algorithm **82**
15.0.2 Initial flow for successively shortest path algorithm 84
15.1 Successively shortest path algorithm 84

16 Time-dependent dynamic flow **92**
16.1 Max-Flow-over-time problem (MFoTP) 93

17 Matchings **95**
17.1 Definitions and optimality criterion 95
17.2 Matchings in bipartite graphs 98
17.3 Theorem of Tutte 101

18 Blossom algorithm **106**
18.1 Using Edmonds Blossom Algorithm to determine a matching with maximal cardinality 111

19 Weighted matching problems and complete unimodular matrices **113**

19.1 Weighted matching problems 113
19.2 The MinWPMP in bipartite graphs 114
19.3 Definition of the assignment problem as (MI)LP 116
19.3.1 Examples 124

20 Matroids **129**
20.1 Maximization problem for IDS 130
20.2 Minimization problem for IDS 130

21 Examples **130**
21.1 Max-Weight Stable Set Problem 130
21.2 Travelling Salesman Problem 130
21.3 Shortest-Path Problem 131
21.4 Knapsack Problem 131
21.5 Minimum Spanning Tree Problem 131
21.6 Maximum Weighted Forest Problem 131
21.7 Maximum Weight Matching Problem 132
21.8 Additional matroid axioms 137
21.9 Duality of matroids 141
21.10The greedy algorithm 142
21.11IDS for TSP 143

1 Course organization

This lecture took place on 2nd of Oct 2014.

Practicals every 2 weeks 2 hours. Homepage at opt.math.tu-graz.ac.at.
Partial exams:

- 28th of Nov 2014
- 30th of January 2014

Contents:

- graph theory
- linear optimization

- algorithms for linear optimization
- spanning trees and tree views
- integer programming
- shortest path problem
- network flows
- matching problems
- matroids

The lecture and therefore these lecture notes are heavily inspired by the book "Kombinatorische Optimierung".

Please send any bugs in these lecture notes to admin@lukas-prokop.at. This document is released under the terms and conditions of Public Domain.

2 Introduction

2.1 A generic combinatorial optimization problem

Synonym for "Instance". Input, given.

Synonym for "Task". Output.

An instance has a finite base set $E = \{e_1, \ldots, e_n\}$. The set of valid solutions is a subset $\mathcal{F} \subseteq 2^E = \mathcal{P}(E)$. One valid solution is $F \in \mathcal{F}$.

$$c : \mathcal{F} \to \mathcal{R}$$

$$F \mapsto c(F)$$

A task is some $F^* \in \mathcal{F}$ with $c(F^*) = \min_{F \in \mathcal{F}} c(F)$ (minimization problem). Some $F^* \in \mathcal{F}$ with $c(F^*) = \max_{F \in \mathcal{F}} c(F)$ is a maximization problem.

2.2 Possible common cost models

$$w : E \to \mathcal{R}$$

$$e \mapsto w(e)$$

Where w is called weight. Two common cost functions:

$$c(F) := \sum_{e \in F} w(e) \qquad \text{(sum problem)} \tag{1}$$

$$c(F) := \max_{e \in F} w(e) \qquad \text{(bottleneck problem)} \tag{2}$$

3 Problem 1: Drill machine problem

A drill machine must drill holes onto a board. Drill heads move vertically. Boards move horizontally. Movements happen with constant speed. Movements can happen simultaneous. Drilling is not consider as movement. With these assumptions the time necessary to drill all holes is directly proportional to the path taken by the drill head and board.

$$\text{production time} \propto \text{path(drill head, board)}$$

The time taken to move from hole 1 to hole 2 is

$$\max\{|x_1 - x_2|, |y_1 - y_2|\}$$

The total costs to drill all holes are:

$$\sum_{i=1}^{n-1} \max\{|x_i - x_{i+1}|, |y_i - y_{i+1}|\}$$

where the relative coordinates of n holes is denoted as (x_i, y_i) with $1 \leq i \leq n$ and the holes are drilled in order $1, 2, \ldots, n$.

Is π a permutation of $\{1, 2, \ldots, n\}$ $(\pi \in S_n)$. If holes are drilled in order π, then

$$\text{production time} = \sum_{i=1}^{n-1} \max\{|x_{\pi(i)} - x_{\pi(i+1)}|, |y_{\pi(i)} - y_{\pi(i+1)}|\}$$

The drill machine problem as generic problem (sum problem):

$$c = \text{production time}$$

$$\mathcal{F} = S_n$$

$$c : S_n \to \mathcal{R}$$

$$\pi \to \sum_{i=1}^{n-1} \max\{|x_{\pi(i)} - x_{\pi(i+1)}|, |y_{\pi(i)} - y_{\pi(i+1)}|\}$$

$$E = \{l_\infty(P_i, P_j) : 1 \leq i, j \leq n, i \neq j\}$$

4 Problem 2: Scheduling problem

Given. We have m workers and n tasks. We assume that every task must be completed by some worker. Not necessarily every task can by done by any worker. Let $S_i \subseteq \{1, 2, \ldots, m\}$ the set of workers, that can complete job i where $1 \leq i \leq n$. All workers of S_i complete task i with the same speed. Let t_i be the required time to complete job i. Every task can be completed by several workers. Every worker can work on several tasks, but *not* simultaneously.

Find. Define a work schedule to minimize the total time to complete all tasks (bottleneck problem).

We define t_{ij} as the length of time interval in which worker j completes job i such that $t_i = \sum_{j \in S_i} t_{i,j}$ $\forall 1 \leq i \leq n$. The work time of worker j is defined as $\sum_{i:j \in S_i} t_{i,j}$ and time to complete is defined $\max_{1 \leq j \leq m, i:j \in S_i} \sum t_{ij} \to \min$. The completeness time is called "makespan".

5 Partial enumeration

Given. $n \in \mathcal{N}, n \geq 3, \{p_1, p_2, \ldots, p_n\}$ are points on a plane

Find. a permutation $\pi^* \in S_n$ with $c(\pi^*) = \sum_{i=1}^{n-1} d_\infty(P_{\pi^*(i)}, P_{\pi^*}(i+1))$ minimized

1. Let $\pi(i) = i, \pi^*(i) = i, \; \forall\, 1 \leq i \leq n, i = n-1$
2. Let $k = \min(\{\pi(i)+1, \ldots, n+1\} \setminus \{\pi(1), \pi(2), \ldots, \pi(i-1)\})$
3. if $k \leq n$ then
   (a) Let $\pi(i) = k$
   (b) if $i = n$ and $c(\pi) < c(\pi^*)$ then $\pi^* = \pi$
   (c) if $i < n$ then set $\pi(i+1) = 0$ and $i = i+1$.

   if $k = n+1$ then $i = i-1$
   if $i \geq 1$ then goto 2

5.1 Working principle

In every step the algorithm finds the (lexicograpical) next possible value for $\pi(i)$ without duplicates of $\pi(1), \pi(2), \ldots, \pi(i-1)$. If this is impossible, then reduce i by 1 (backtracking appraoch). Otherwise we set $\pi(i)$ to our new value k. If $i = n$ then a new permutation is given and costs are evaluated and compared, otherwise the algorithm tries all possible values $\pi(i+1) \ldots \pi(n)$ and starts with $\pi(i+1) = 0$ with i being incremented successively.

Hence the algorithm generates all permutation in lexicographical order.

5.2 Algorithm efficieny

The actual costs can only be computed relative to n. We define costs in terms of steps. One step is defined as one arithmetic operation, assignment, comparison, logical statements, goto jump or value lookup ("elementary step", ES). We look at the algorithm in terms of costs:

1. $2n+1$ ES
2. $\mathcal{O}(n)$ with helper vectors auxiliary$(j) = 1$ if $j < i$.
3. Constant number of ES unless $i = n$, then $2n+1$ additionally. In any case not more than $\leq \mathcal{O}(n)$ ES.

How often are steps 2 and 3 of the algorithm executed? $\mathcal{O}(n^2)$ if new permutation is not created (without backtracking) and $\mathcal{O}(n)$ with backtracking. In total $\mathcal{O}(1)$ every time $\mathcal{O}(n^2)$ (without backtracking) or $\mathcal{O}(n)$ every time $\mathcal{O}(n)$ each $\mathcal{O}(n^2)$ (with backtracking).

So the algorithm has computational complexity of $\mathcal{O}(n^2 \cdot n!)$ ES.

6 Analysis of algorithms

This lecture took place on 6th of Oct 2014.

A finite, deterministic algorithm is a sequence of valid inputs and instructions which consists of elementary steps such that the computational task gets completed for every possible input. For every possible input the algorithm computes a finite, deterministic output.

Given. A sequence of numbers. If rational, then binary encodable:

$$e \in \mathbb{Z} \to_{\text{encodes}} \log |a| + 2$$

Logarithms are always considered with base 2 here.

Inputsize. We denote the size of the input for x with $\text{size}(x)$. For some rational input x the size $\text{size}(x)$ is the number of 0 and 1 in the binary representation.

1. Let A be an algorithm which accepts inputs $x \in \mathcal{X}$. Let $f : \mathbb{N} \to \mathbb{R}_+$. If there is some constant $\alpha > 0$, such that A $\forall x \in X$ terminates the computation after at maximum $\alpha f(\text{size}(x))$ elementary steps, we say "A has a *time complexity* of $\mathcal{O}(f)$".
2. An algorithm A with rational inputs has a *polynomial* runtime (or "is polynomial") iff $\exists k \in \mathbb{N}_{\nvdash}$
   (a) A has a time complexity of $\mathcal{O}(n^k)$
   (b) all intermediate values of the computation can be stored with $\mathcal{O}(n^k)$ bits
3. An algorithm A with arbitrary input is called *strongly polynomial* if $\exists k \in \mathbb{N}_0$ such that A
   (a) requires for every of m numbers of the input a runtime of $\mathcal{O}(n^k)$
   (b) is polynomial for every rational input
4. An algorithm A which is polynomial, but not strongly polynomial, is called "weakly polynomial".
5. Let A be an algorithm which computes for every input $x \in X$ output $f(x) \in Y$. We state that A computes the function $f : X \to Y$. Is a function computable by a polynomial algorithm, we call it a *polynomial computable function.*

The runtime of a polynomial algorithm is a function of the input. The runtime of a strongly polynomial algorithm is a function of the *number* of input elements.

Remark. Let A have runtime complexity $\mathcal{O}(n^2)$. This means not all instances of input length n require $\theta(n^2)$ elementary steps. $\mathcal{O}(n^2)$ is an upper bound (worst-case time complexity).

7 Spanning trees and arborescences

$$G = (V, E) \qquad e \in E \text{ is } e = \{x, y\} \text{ with } x, y \in V$$

where $V(G)$ is the set of vertices of G and $E(G)$ is the set of edges of G. One of the earliest problems in combinatorial optimization is the computation of minimal spanning trees.

7.1 Minimal spanning tree problem (MST)

Given. Undirected graph G, c, $E(G) \to (R)$

Find. Find a spanning tree T with minimal weight $c(T) = \sum_{e \in T} c(e)$ in G or determine "G is not connected".

7.2 Maximum weight forest problem (MWF)

Given. Undirected graph G, $c : E(G) \to (R)$

Find. A spanning forest F (cyclefree subgraph with vertices set $V(G)$) with maximum weight

$$c(F) := \sum_{e \in F} c(e) \in G$$

7.3 Equivalence of problems

Two problems are called *equivalent* if P is reducible to Q and Q is reducible to P. P is reducible to Q, if there are two linear computable functions f and g such that

1. for every instance I of P, $f(I)$ is an instance of Q
2. for every solution L of Q, $g(L)$ is a solution of P

$$(P[I]) \xrightarrow{f} (Q[f(I)], L) \xrightarrow{g} g(L)$$

7.4 MST and MWF are equivalent

Theorem 1. *The MWF problem and MST problem are equivalent.*

7.4.1 Reduce MWF to MST

MWF is reducible to MST. Let (G, c) be an instance of MWF. Remove all $e \in E(G)$ with $c(e) < 0$. Let $c'(e) = -c(e)$ for all remaining edges of $E(G)$. Insert a minimal set of edges F with arbitrary weights, such that the resulting graph is connected. Denote this graph with G'.

The computationally most intense task is insertion of the minimal set of edges. Determination of connected components is possible in linear time (eg. with DFS).

Consider instance (G', c') of the MST problem. Let T' be an optimal solution of MST with (G', c').

Remove F of T'. Let T be the resulting subgraph. Show that T is a spanning forest with maximum weight in (G, c). ($F \subseteq E(T')$ results from the definition of F as *minimum* ...).

T must be spanning forest because T' is a spanning tree.

$$c(T) = -(c'(T') - c'(F)) = -c'(T') + c'(F)$$

$c'(F)$ is the constant available in all spanning tree of G'. If T' minimizes $c'(T')$ such that T of $c(T)$ is maximized.

7.4.2 Reduce MST to MWF

Let (G, i) be an instance of the MST problem. Let

$$c'(G) = \mathcal{K} - c(e) \text{ with } \mathcal{K} = \max_{e \in E(G)} c(e) + 1 \Rightarrow c'(e) > 0 \,\forall\, e \in E(G)$$

Consider (G, c') as instance of MWF (linear runtime). Let F be a maximum weight spanning forest in (G', c'). Case distinction:

F is not a tree G is not connected

F is a spanning tree F is the optimal solution of MST because $c'(F) = \sum_{e \in E(F)} (\mathcal{K} - c(e)) = (|V(G)| - 1)\mathcal{K} - \sum_{e + E(F)} c(e) = (|V(G)| - 1)\mathcal{K} - c(F)$

Theorem 2. *(Optimality conditions.) Let (G, i) be an instance of MST and T be a spanning tree in G. In this case the following statements are equivalent:*

- *T is optimal*
- *$\forall\, e = \{x, y\} \in E(G) \setminus E(T)$: no edge of the x-y-path in T has greater weight than e*
- *$\forall\, e \in E(T)$: If C is one of the connected components of $T \setminus \{e\}$, then e is an edge from $\delta(V(c))$ with minimal weight.*

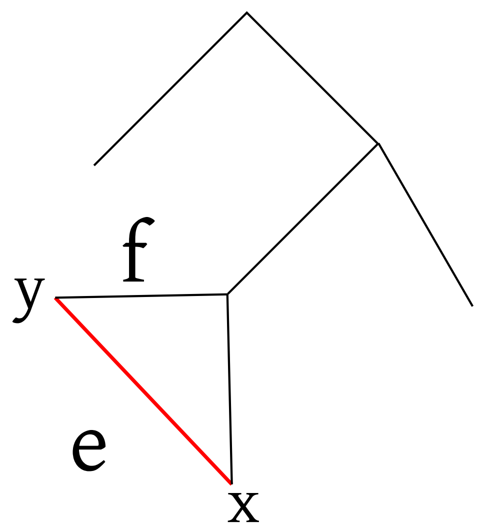


Figure 1: Sketch for $a \Rightarrow b$

- *$E(T) = \{e_1, e_2, \dots, e_{n-1}\}$ can be ordered such that $\forall\, i \in \{1, 2, \dots, n-1\}$ there is a set $X \subseteq V(G)$ such that $e_i \in \delta(X)$ with minimal weight ad $e_j \neq \delta(X)\ \forall\, j \in \{1, 2, \dots, i-1\}$.*

Cut.

$$X \subset V(G)$$

$$\delta(X) = \{e \in E(G) : |e \cap X| = 1\}$$

Theorem 3. *$a \Rightarrow b \Rightarrow c \Rightarrow d \Rightarrow a$.*

7.4.3 $a \Rightarrow b$

$c(e) \geq c(f)$ for every f in x-y-path in T, because otherwise $T - e + f$ is a spanning tree with $c(T - e + f) = c(T) - c(e) + c(f) < c(T)$ which contradicts.

7.4.4 $b \Rightarrow c$

Show $\forall\, e \in E(T) : c(e) \leq c(f)\ \forall\, f \in \delta(V(c))$. Every edge $e \in T$ defines an cut $\delta(V(c)) = \delta(e)$. Hence we improve the tree.

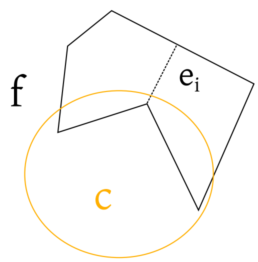


Figure 2: Sketch for $b \Rightarrow c$

7.4.5 $c \Rightarrow d$

Show there exists some weighted order and cuts with properties like in c. Select a random order $\{e_1, e_{i-1}, e_i, e_{n-1}\}$. $\forall e \in \{1, 2 \ldots, n-1\}$ consider cut $\delta(c_{e_i})$. It has the desired properties.

7.4.6 $d \Rightarrow a$

Assumption $E(T) = \{e_1, \ldots, e_{n-1}\}$ is satisfied.

Let T^* be an optimal spanning tree such that $i(T^*) = \min\{h \in \{1, 2, \ldots, n-1\} : e_n \notin E(T^*)\}$ is maximum.

We show $i = +\infty$ and hence $T^* \equiv T \Rightarrow T$ is optimal. Assumption $i < +\infty$. Then $X \subset V(G)$ with $e_i \in \delta(v)$ with minimal weight with $e_j \notin \delta(X) \, \forall j < i$.

$$\exists f \neq e_i \text{ with } f \in T^* \cap \delta(X)$$

$$c(f) \geq c(e_i)$$

$$c(f) \leq c(e_i)$$

$$\Rightarrow c(f) = c(e_i)$$

$T^* - f + e_i$ is an optimal spanning tree and has $i(T^* - f + e_i) > i(T^*)$. This is a contradiction.

7.5 Kruskal's algorithm

This lecture took place on 7th of Oct 2014.

Theorem 4. *Krukal's algorithm is correct.*

Algorithm 1 Kruskal's algorithm

Given. G is a connected undirected graph, $c : E(G) \to \mathbb{R}$

Find. minimal spanning tree

Sort edges $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m) \qquad (m = |E(G)|)$
Set $T := (V(G), \phi)$
for i from 1 to m **do**
    if $T \cup \{e_i\}$ is cycle-free **then**
        $E(T) = E(T) \cup \{e_i\}$
    end if
end for

From the previous theorem we can derive: $\forall\, c = \{x, y\} \notin E(T)$ we can say $c(f) \leq c(e)\ \forall\, f$ edges from the x-y-path in T.

$$(x, y) = e \notin E(T) \Rightarrow e_i \text{ closes cycle with } E(T) \cap \{e_1, \ldots, e_{i-1}\}$$

$$\Rightarrow E(x - y - \text{path} \in T) \subseteq \{e_1, \ldots, e_{i-1}\}$$

$$\Rightarrow\ \forall\, f \in E(x - y - \text{path}) : c(f) \leq c(e_i)$$

Trivial implementation. $\mathcal{O}(m \cdot n)$ because there are m iterations and per iteration one check whether the current edge with the given T ($\leq n$ edges) creates a cycle ($\mathcal{O}(n)$ with DFS).

Definition. A digraph G is called *branching*, if it is cycle-free and every $v \in V(G) : \text{indegree}(v) \leq 1$.

Notation. $\text{indegree}(v) = \deg^-(v)$.

Definition. A connected branching is called *arborescence*. The vertex r with $\deg^-(r) = 0$ is called *root*. An arborescence is the directed-graph form of a rooted tree.

Notation.

$$\delta(\{v\}) = \delta(v)$$

$$\delta^+(v) = \{e = (v, y) \in E(G)\}$$

$$\delta^-(v) = \{e = (x, v) \in E(G)\}$$

Theorem 5. *Let G be a digraph with n vertices. The following 7 statements are equivalent:*

1. *G is an arborescence with root r.*
2. *G is a branching with $n - 1$ edges and $\deg^-(r) = 0$.*
3. *G has $n - 1$ edges and every vertices is reachable from r.*
4. *Every vertex is reachable from r and removal of one edge destroys this property.*

5. *G satisfies $\delta^+(X) \neq 0 \;\forall X \subset V(G)$ with $r \in X$. The removal of one arbitrary edge destroys this property.*
6. *$\delta^-(r) = 0$ and $\forall v \in V(G) \setminus \{r\} \,\exists$ one distinct directed $r-v$-path in G*
7. *$\delta^-(r) = 0$ and $|\delta^-(v)| = 1 \;\forall v \in V(G) \setminus \{r\}$ and G is cycle-free.*

A proof for Theorem 5 is not provided. It will be provided in the practicals.

Theorem 6. *Kruskal's algorithm can be implemented with time complexity $\mathcal{O}(m \log n)$.*

Proof. The implementation keeps a branching B with

- $V(B) = V(G)$
- connected components of B vertex-correspond with the connected components of T

□

How can we create such a branching?

1. At the beginning (after initialization): $B = (V(G), 0)$.

Be aware that checking whether $\{v, w\}$ create a cycle with T, consider that w must be in the same connected component in T (or B). We can check this in $\mathcal{O}(\log n)$.

In branching:

- Let $r_v(r_w)$ be the root of $v(w)$ contained solutions of B.

$$r_v = r_w$$

  The computational effort for checking is equivalent to the effort for the determination of r_v and r_w which is proportional to the sum of the lengths of r_v-v-path or r_w-w-path in B.

Hypothesis. *In B it holds that $h(r) \leq \log n$ for every root r where $h(r)$ is the maximum length of an r-v-path in B.*

Proof. Induction over number of edges in B.

Base. For zero edges this is trivial to prove.
Step. We will add a new edge $\{x, y\}$ and beforehand $h(r) \leq \log n$ is satisfied. We have to show that $h(r) \leq \log n$ is satisfied after the insertion of $\{x, y\}$.

Case distinction:

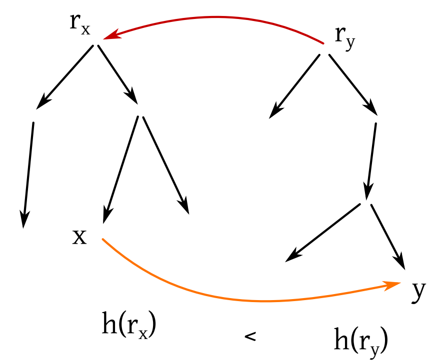


Figure 3: Branching insertion operation

- $h(r_x) = h(r_y)$. Insert (r_x, r_y) or (r_y, r_x) In the figure we have to ensure $h(r_x) \leq \log r_x \Leftrightarrow m_x \leq 2^{h(r_x)}$.

$$\hat{h}(r_x) = h(r_x) + 1$$

$$\begin{aligned} n_{x,y} = n_x + n_y &\geq 2^{h(r_x)} + 2^{h(r_y)} \\ &= 2 \cdot 2^{h(r_x)} = 2^{h(r_x)+1} \\ &= 2^{\hat{h}(r_x)} \end{aligned}$$

- $h(r_x) < h(r_y)$. Insert (r_y, r_x).

$$\hat{h}(r_y) = h(r_y)$$

$$n_{xy} \geq n_y \geq 2^{h(r_y)} = 2^{\hat{h}(r_y)}$$

□

7.6 Prim's algorithm

Theorem 7. *Prim's algorithm is correct and can be implemented with time complexity of $\mathcal{O}(n^2)$. Correctness follows from theorem 2.2.d ($a \Rightarrow b \Rightarrow c \Rightarrow d \Rightarrow a$): Spanning tree is optimal $\Leftrightarrow$ order of edges $e_1, \ldots, e_{n-1}$ such that $\forall\, i \in \{1, 2, \ldots, n-1\}\, \exists x_i \subset V(G)$ with $e_i \in \delta(X_i)$ is the minimum edge in $\delta(X_i)$ and $e_j \notin \delta(X_i)$ is the cheapest edge of $\delta(X_i)$ and $e_j \notin \delta(X_i)\, \forall\, 1 \leq j \leq i-1$. This is satisfied by construction.*

Algorithm 2 Prim's algorithm

Given. G is a connected undirected graph, $c: E(G) \to \mathbb{R}$
Find. minimal spanning tree

1: Set $T = (\{v\}, 0)$ for an arbitrary $v \in V(G)$
2: **while** $V(T) \neq V(G)$ **do**
3: Select one edge $e \in \delta_G(V(T))$ with minimum weight
4: $T := T + e$
5: **end while**

The desired order (or cuts) will be created by the algorithm.

Time complexity. The number of iterations is the number of edges in the tree which is $n-1$. We have to show that every iteration is completed in $\mathcal{O}(n)$ time.

Maintain a list of candidate edges: $\forall\, w \notin V(T)$ is the candidate edge $K(w)$ the minimal edge between w and $V(T)$. In the general case we add the minimal edge $K(w)\ \forall\, w \notin V(T)$ $(T = T + k(w))$.

Definition. $|V(G) \setminus V(T)| = \mathcal{O}(n)$ can be computed in $\mathcal{O}(n)$ time.

Update of candidate edges: If v_k was added to T in the last iteration then compare $c(v_k, w), c(k(w))$ and if $c(v_k, w) < c(k(w))$ then $k(w) = (v_k, w)$. This requires $\mathcal{O}(n)$ time.

Theorem 8. *Is Prim's algorithm implemented with Fibonacci-Heaps we can solve the MST problem in $\mathcal{O}(m + n \log n)$ time.*

$$\mathcal{O}(n^2) \qquad \mathcal{O}(m + n\log n) \qquad m = \theta(n^2) \qquad G \textit{ is dense}$$

A proof for Theorem 8 is not provided.

8 Number of spanning trees

This lecture took place on 9th of Oct 2014.

Theorem 9. *(Arthur Cayley) The complete graph K_n has n^{n-2} spanning trees.*

Proof. (J. Pitman, Coalescent random forests, Journal of Combinatorial Theory A 85, 1999, 165–193) Double-counting approach counting the number of labelled rooted trees (LRT). In labelled trees every edge has a label. Two LRTs are equivalent if and only if their tree structure is the same and labels are equivalent.

1. Let $\tau(n)$ be the number of spanning trees in K_n. Every tree can have n root candidates and $(n-1)!$ labels. In conclusion we can create $n \cdot (n-1)! \cdot \tau(n)$ LRTs with n vertices.

2. Insert edges successively such that adding $n-1$ vertices creates a LRT. For one edge we have ...

$$2 \cdot \frac{n(n-1)}{2} = n(n-1) \text{ possibilities}$$

We added k edges ($k < n-1$). Followingly the graph has $n-k$ connected components with $n_1, n_2, \ldots, n_{m-k}$ vertices each. Every connected component is a LRT. If the $k+1$-th edge to be added starts at component 1, then this edge must have the root as source. This edge can have every other vertex as destination. There are $n - n_1$ possible such edges. In total we start with an arbitrary component and get:

$$(n-n_1) + (n-n_2) + \ldots + (n - n_{n-k}) = n(n-k) - n$$

This is the number of possilities for edge $k+1$. In total for all $n-1$ edges to add:

$$\prod_{k=0}^{n-2} (n \cdot (n-k) - n) = \prod_{k=0}^{n-2} n(\underbrace{n-k-1}_{1 \leq t \leq n-1}) = n^{n-1}(n-1)$$

It holds:

$$n(n-1)!\delta(n) = n^{n-1}(n-1)! \Rightarrow \delta(n) = n^{n-2}$$

□

8.1 Minimum Weight Arborescence Problem (MWA)

Given. Digraph $G = (V, E), c : E(b) \to \mathbb{R}$

Find. Spanning arborescence with minimum weight or claim $\nexists$ spanning arborescence in G

8.2 Minimum Weighted Rooted Arborescence Problem (MWRA)

Given. Digraph $G = (V, E), r \in V(G), c : E(G) \to \mathbb{R}$

Find. Spanning arborescence with root r and with minimum weight in G or claim $\nexists$ spanning tree with root r in G

8.3 Maximum Weighted Branching Problem (MWB)

Given. Digraph $G = (V, E), c : E(G) \to \mathbb{R}$

Find. Branching B with maximum weight

8.4 Equivalence of MWA, MWRA and MWB

Hypothesis. *The three problems MWA, MWRA and MWB are equivalent.*

Partially the proof is given in the practicals.

Proof. We assume without loss of generality that $c(e) \geq 0 \; \forall \, e \in E(G)$ because negative edges cannot occur in a maximal branching.

$$\deg^{-}(v) \leq 1 \; \forall \, v \in V(G)$$

Greedy approach: $\forall \, v \in V(G)$ select one $e_v \in \operatorname{argmax}\{c(e) : e = (x, v) \in E(G)\}$ let $B := \{e_v : v \in V(G)\}$.

If B_0 is cycle-free: B_0 is branching with maximum weight. Otherwise cycles have to be avoided / destroyed.

Theorem 10. *Let B_0 be a subgraph of G with maximum weight and $\deg^{-}_{B_0}(v) \leq 1 \; \forall \, v \in V(G)$. Then $\exists$ an optimal branching $B \in G$ with properties $\forall$ cycle $C \in B_0 : |E(C) \setminus E(B)| = 1$.*

Proof. **Assumption.** Such an optimal branching B does not exist.

Let B be a maximal branching in G with maximum number of common edges with B_0. Let C be a cycle in B_0 with $|E(C) \setminus E(B)| \geq 2$.

$$E(C) \setminus E(B) = \{(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)\}$$

in order they appear inside the cycle.

Hypothesis.

$$\forall \, 1 \leq i \leq k \exists b_i - b_{i-1} - \mathit{path\ in}\ B (b_0 \equiv b_k)$$

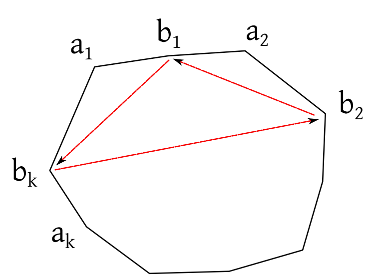


Figure 4: Red cycle

The existence of a red cycle in B shows a contradiction.

Consider a fixed $i \in \{1, 2, \ldots, k\}$. Let B_i' be a subgraph of G with $V(B_i') = V(G)$ and $E(B_i') = \{(x, y) \in B : y \neq b_i\} \cup \{(a_i, b_i)\}$. If so, then $c(B_i') \geq c(B)$ and thus B_i' would be an optimal branch with one common edge (a, b) more in B_0. This is a contradiction.

B_1' is no branching. So there is a cycle in B_i', which contains (a_i, b_i). So a b_i-a_i-path in B_i' is exists in B. $\square$

This b_i-a_i-path does not exist in C. Otherwise he would be forced to use an edge (a_i, b_i) which he is not allowed to do so, because $(a_i, b_i) \not\ni B$.

Let $e = (x, y)$ be the last edge of the b_i-a_i-path which is outside of the cycle. $y = b_j$ must hold because otherwise there are two discharging edges in B, which is a contradiction.

8.5 Edmonds' branching algorithm

$\square$

Theorem 11. *Edmonds' Branching Algorithm is correct and computes the branching in $\mathcal{O}(m \cdot n)$.*

Proof. The algorithm constructs a sequence of graphs $G_0 = G, G_1, \ldots, G_k$ are edge-weight-free $c_0 = c, c_1, \ldots, c_k$ until the "greedy solution" in the current G_k is also a branching (B_k). Then the algorithm transform this B_k into a branching B_{k+1} in G_k successively until G_0. It suffices to show that the transformation of a branching B_i into B_{i+1} provides a branching. With induction it immediately follows that the B_0 branching in $G_0 = G$.

Let B_i be a branching in G. We show that the algorithm's B_{i-1}^* is an optimal branching in G_i with $\left|E(C) \setminus E(B_{i-1}^*)\right|$.

The number of cycles C in $B_{i-1} = 1$ (B_{i-1} exists according to our previous theorem).

We derive B_i^* from B_{i-1}^* by the constructed cycle of B_{i-1}. Then B_i^* is a branching in G_i.

$$c_{i-1}(B_{i-1}^*) = c_i(B_i^*) + \sum_{C' \text{cycle} \in B_{i-1}} (c_{i-1}(C') - c_{i-1}(e(C')))$$

$$c_{i-1}(B_i) \geq c_i(B_i^*)$$

$$c_{i-1}(B_{i-1}^*) \leq c_i(B_i) + \sum_{C' \text{cycle} \in B_{i-1}} [c_{i-1}(C') - c(e(C'))]$$

$$= c_{i-1}(B_{i-1})$$

so B_i is an optimal branching of G_i according to the induction hypothesis.

Runtime analysis. n iterations, $\mathcal{O}(m)$ time per iteration $\square$

Algorithm 3 Edmonds' branching algorithm (book: page 153)

Given. Digraph G, weights $c : E(G) \to \mathbb{R}_+$
Find. A branching B of G with maximum weight
Remark: $\alpha(e, C)$ is the edge (u, v) inside C, which shares v as destination with e but not u. $\psi_i(e)$ is the edge e in the iteration i of the algorithm.

1: Set $i := 0, G_0 := G, c_0 := c$.
2: Let B_i be a subgraph of G_i with maximum weight and $\left|\delta^-_{B_i}(v)\right| \leq 1$ for all $v \in V(B_i)$
3: **if** B_i is cycle-free **then**
4:     $B := B_i$ **go to** 18
5: **end if**
6: Let $\mathcal{C}$ be the set of cycles in B_i. Select one $C \in \mathcal{C}$.
7: Let $V(G_{i+1}) := \mathcal{C} \cup (V(G_i) \setminus \bigcup_{C \in \mathcal{C}} V(C))$.
8: **for** $e = (v, w) \in E(G_i)$ **do**
9:     $e' = (v', w') \in E(G_{i+1})$
10:     $\Phi_{i+1}(e') := e$ where
$v' = C$ if $v \in V(C)$ for $C \in \mathcal{C}$ and $v' = v$ if $v \notin \bigcup_{C \in \mathcal{C}} V(C)$ and
$w' = C$ if $w \in V(C)$ for $C \in \mathcal{C}$ and $w' = w$ if $w \notin \bigcup_{C \in \mathcal{C}} V(C)$.
11: **end for**
12: Let $E(G_{i+1}) := \{e' = (v', w') : e \in E(G_i), v' \neq w'\}$ ▷ (parallel edges might occur)
13: **for** $e = (v, w) \in E(G_i)$ with $e' = (v', w') \in E(G_{i+1})$ **do**
14:     $c_{i+1}(e') := c_i(e)$ if $w' \notin \mathcal{C}$
15:     $c_{i+1}(e') := c_i(e) - c_i(\alpha(e, C)) + c_i(e_C)$
if $w' \in C \in \mathcal{C}$ where $\alpha(e, C) \in \delta^-_C(w)$ and e_C is the cheapest edge of C.
16: **end for**
17: Set $i := i + 1$ **go to** 2
18: **while** $i > 0$ **do**
19:     $B' := (V(G_{i-1}), \{\Phi_i(e) : e \in E(B)\})$
20:     **for** every cycle C of B_{i-1} **do**
21:         **if** some edge $e \in \delta^-_{B'}(V(C))$ exists **then**
22:             $E(B') := E(B') \cup (E(C) \setminus \{\alpha(e, C)\})$ ▷ Delete $\alpha(e, C)$
23:         **else**
24:             $E(B') := E(B') \cup (E(C) \setminus \{e_C\})$ ▷ Delete cheapest edge in cycle
25:         **end if**
26:     **end for**
27:     $B := B'$
28:     $i := i - 1$
29: **end while**

9 Shortest path problems in graphs

This lecture took place on 13th of Oct 2014.

Given. $G = (V, E)$ is a digraph.

A sequence of vertices $v_1, v_2, \ldots, v_k$ is called *edge sequence* if $(v_i, v_{i+1}) \in E(G)\ \forall\, 1 \leq i \leq k-1$. A sequence of vertices $v_1, \ldots, v_k$ such that $\forall\, e \in E(G)$ there is at most one index $1 \leq i \leq k-1$ with $e = (V_i, V_{i+1})$ is called a *walk*. A walk which does not use any vertex twice is called *path*.

We distinguish between inner vertices and end vertices. A path with end vertices u and v is a u-v-path. Let $P = (v_1, \ldots, v_k)$ a v_1-v_2-path and $1 \leq i < j \leq k$. Then compute only $(v_i, v_{i+1}, \ldots, v_j)$ as $P_{[v_i, v_j]}$.

A *cycle* (a path with the same start and end vertex) is a *closed path*. Let $s, t \in V(G)$ then

$$d(s,t) := \begin{cases} \text{length (number of edges) of the shortest } s\text{-}t\text{-path in } G & \\ +\infty & \nexists s - t - \text{path} \end{cases}$$

If $c \in E(G) \to \mathbb{R}$

$$d^G(s,t) = \begin{cases} \sum_{e \in P} c(e) & \text{where P is the shortest path in G with c} \\ +\infty & \nexists s - t - \text{path} \end{cases}$$

9.1 Single source shortest path problems (SSSP)

Given. $G = (V, E)$ is a digraph, $c : E(G) \to \mathbb{R}, s \in V(G)$

Find. $\forall\, v \in V(G)$ find a shhortest s-t-path in G

Consider a digraph G where $c(e) = -1\ \forall\, e \in E(G)$ and $s, t \in V(G)$. The shortest path is to visit all edges infinitely. A s-t-path has at maximum $n-2$ vertices

$$\geq d^G(s,t) \geq -(n-1)$$

$$d^G(s,t) = -(n-1) \Leftrightarrow \exists \text{Hamiltonian } s-t\text{-path in } G$$

Detection of Hamiltonian paths is a NP-complete problem.

Definition. A weighting w in a graph D is called *conservative*, if the sum of weights in every cycle of D is non-negative. In other words:

- $\nexists$ negative cycles in G.
- Let K be the set of cycles in a graph G. For every cycle $C \in K$, $w(C) \geq 0$ holds with $w(C) := \sum_{e \in E(C)} c(e)$.
- In case of digraphs, directed cycles have to be considered.

Remark. Analogous problem in undirected graphs are in general more difficult that in digraphs.

If $c(e) \geq 0 \;\forall\, e \in E(G)$ the graph can be transformed into a digraph.

Theorem 12. *Let G be a digraph with conservative weights. $c : E(G) \to \mathbb{R}$. Let $s, w \in V(G)$ and $k \in \mathbb{N}$. Let P be the shortest among all s-w-pathes with at most k edges. Let $e = (v, w)$ be the last edge of P. Then $P_{[s,w]}$ is the shortest s-v-path with at most $(k-1)$ edges.*

Proof. We assume $\exists$ s-v-path Q with $c(Q) < c(P_{[s,v]})$. Case distinction:

1. $w \notin V(Q)$. Consider s-w-path P_1 as chain of Q with (v, w). Then

$$c(P_1) = c(Q) + c(v, w) < c(P_{[s,v]}) + c(v, w) = c(P)$$

   This is a contradiction.

2.

$$c(Q_{[s,w]}) = c(Q) - c(Q_{[w,v]}) = \underbrace{c(Q)}_{c(P_{[s,v]})} + c(v, w) - [c(Q_{w,v}) + c(v, w)]$$

$$< \underbrace{c(P)}_{P_{[s,v]}(v,w)} - \underbrace{\big(c(Q_{[w,v]}) + c(v, w)\big)}_{\text{cycle } K} \leq c(Pc)$$

   We select P because $Q_{[s,w]}$ is shorter than P and has at most $k-1$ edges as part of a s-v-path Q. This is also a contradiction.

□

In the following we assume $c(e) \geq 0 \;\forall\, e \in E(G)$.

9.2 Dijkstra's algorithm for SSSP

Theorem 13. *Dijkstra's algorithm is correct and can be implemented in $\mathcal{O}(n^2)$.*

Proof. The following statements are invariants of the algorithms:

1. $\forall\, v \in V(G) \backslash \{v\}$ with $l(v) < +\infty$ results in $p(v) \in R$, $e(p(v)) + c(p(v), v) = l(v)$ and $v, p(v), p(p(v)), \ldots$ contains 1.

2. $\forall\, v \in R$ it holds that $l(v) = \mathrm{dist}^{(G,c)}(s, v)$.

Why?

Algorithm 4 Dijkstra's algorithm

Given. $G = (V, E)$ is a digraph. $c : E(G) \to \mathbb{R}^+, s \in V(G)$
Find. A shortest path of s to v, $\forall\, v \in V(G)$. Let $l(v)$ be the length of a shortest path $s - v$ in G and $p(v)$, such that $p(v)$ is the predecessor of v of the shortest s-v-path in G provided by the algorithm $\forall\, v \in V(G)$. If v is not reachable from j, then $l(v) = +\infty$ and $p(v)$ is not defined.

1: Let $l(s) = 0$, $l(v) = \infty\ \forall\, v \in V(G) \setminus \{s\}$, $p(s) = 0$, $R = \{\}$
2: Find $v \in V(G) \setminus R$ with $l(r) = \min\{e(x) : x \in V(G) \setminus R\}$
3: Let $R = R \cup \{v\}$
4: **for** $w \in V(G) \setminus R$ with $(v, w) \in E(G)$ **do**
5:     **if** $l(w) > l(v) + c(v, w)$ **then**
6:         $\{l(w) := l(v) + c(v, w); p(w) = v\}$
7:     **end if**
8: **end for**
9: **if** $R \neq V(G)$ **then go to** 2
10: **end if**

- After step 1 of the algorithm, both invariants are inherently satisfied. In step 4 we update $l(w)$ and $p(v) = v$ for some $v \in R$ and also $l(u) = l(v) - c(v, w) = l(p(w)) + c(p(w), w)$. $v, p(v), \ldots$ contains s because at the beginning s was the only vertex with $e(\ldots) < \infty$.

- $s \in R$ holds $c(s) = 0 = C(\text{shortest path})$. Induction over $|R|$

  induction base $R = \{s\}$ (trivial).

  induction step Statement is satisfied until the execution of step 3. We have to show that the statement holds after step 3. We assume that this statement is not satisfied. Then for v in step 3 there is some s-v-path P in G with $c(P) < l(v)$. Let y be the first vertex in P which is part of $(V(G) \setminus R) \cup \{v\}$ and x is the predecessor of y in P. Then it holds $x \in R$ in P. From the induction assumption if follows that $\operatorname{dest}^{G,C}(s, x) = l(x)$.

  $$l(y) \leq l(x) + c(x, y) = \operatorname{dest}^{G,C}(s, x) + c(x, y) \leq c(P_{[s,y]}) \leq c(P) < l(v)$$

  This is a contradiction. Directly after $R = R \cup \{x\}$ it holds that after execution of step 4 of the algorithm $l(y) = l(x) + c(x, y)$. Followingly $l(x)$ does not change any more because $x \in R$ and $l(y)$ can only be reduced.

$\square$

9.2.1 Analysis

$\mathcal{O}(n^2)$ because we have n iterations and an effort of $\mathcal{O}(n)$ per iteration.

Theorem 14. *(Fredman and Tarjan, 1987) A Fibonacci-Heap implementation of Dijkstra's algorithm runs in $\mathcal{O}(m + n \log n)$ time.*

A proof for Theorem 14 is not provided.

For planar graphs Dijkstra's algorithm can be implemented with $\mathcal{O}(n)$ time (Henzinger, 1997). For weights of integers ≥ 0 it can be implemented in $\mathcal{O}(n)$ (Thomp, 1999).

9.3 Moore-Bellman-Ford algorithm

Algorithm 5 Moore-Bellman-Ford algorithm

Given. A digraph $G = (V, E)$, conservative $c : E(G) \to \mathbb{R}$, $s \in V(G)$
Find. $l(v), p(v)$ like Dijkstra's algorithm $\forall\, v \in V(G)$

$l(s) := 0, \quad l(v) := \infty \;\forall\, v \in V(G) \setminus \{s\} \quad p(s) = 0$
for $i = 1$ to $n - 1$ **do**
    for $(u, w) \in E(G)$ **do**
        if $l(w) > l(u) + c(u, w)$ **then**
            $l(w) := l(u) + c(u, w)$
            $p(w) := u$
        end if
    end for
end for

9.3.1 Analysis

Theorem 15. *The Moore-Bellman-Ford algorithm is correct and has runtime $\mathcal{O}(nm)$.*

We have $\mathcal{O}(n)$ iterations and need $\mathcal{O}(m)$ time per iteration.

This lecture took place on 14th of Oct 2014.

Proof. Let $R := \{v \in V(G) : l(v) < +\infty\}$ and $F := \{(x, y) \in E(G) : x = p(y)\}$. The following statements are invariants of the algorithm (even for non-conversative weights):

1. $l(y) \geq l(x) + c(x, y) \;\forall\, (x, y) \in F$
2. If F contains a cycle K, then K has a negative weight $c(K) < 0$.

Assumption. A cycle K in F is created by insertion of edge (x, y). At this point in time $p(y) = x$ will be set in the second step. Before $l(y) = l(x) + c(x, y)$

was set because previously it holds that $l(y) > l(x) + c(x,y) \; \forall \, (v,w) \in F$ and $l(w) \geq l(v) + c(v,w)$.

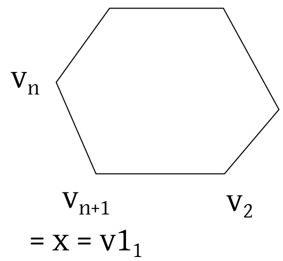


Figure 5: Cycle creation by edge insertion

$$\sum_{i=1}^{n} l(v_i) = l(v_i) + \sum_{i=1, i\neq 2}^{n} l(v_i) + c(v_1, v_2) = \sum_{i=1, i\neq 2}^{n} [l(v_i) + c(v_i, v_{i+1})]$$

$$\Leftrightarrow \sum_{i=1}^{n} l(v_i) > \sum_{i=1}^{n} l(v_i) + \sum_{i=1}^{n} c(v_i, v_{i+1})$$

$$\Leftrightarrow C(k) = \sum_{i=1}^{n} c(v_i, v_{i+1}) < 0$$

Hence if c is conservative, then F is cycle-free.

$$x \in R \Rightarrow l(x) < \infty \Rightarrow l(p(x)) < \infty \Rightarrow p(x) \in R$$
$$\Rightarrow p(p(x) \in R) \text{ until s}$$

Followingly $\forall \, x \in R \exists$ some s-x-path in (R, F) and F is cycle-free. We conclude that (R, F) is some arborescence with root s.

Hypothesis. *After k iterations the length (sum of weights) of the shortest s-x-path with at most k edges is at least $l(x) \, \forall \, x \in R$.*

We prove it by induction over k. Let $P_{s,x}$ be the s-x-path in (R, F).

$$l(x) \geq^a l(p(x)) + c(p(x), x) \geq^a l(p(p(x))) + c(p(p(x)), p(x)) + c(p(x), x)$$
$$\geq^a \ldots \geq^a l(s) + \sum_{e \in P_{s,x}} c(e) = c(P_{s,x}) \, \forall \, x \in R$$

Induction base. For $k = 1$ all edges start at the same vertex.

$$l(v) = l(s) + c(s,v) \qquad \text{thus } l(v) = c(s,v) \, \forall \, v \text{ with } l(v) < \infty$$

Induction step. Assumption. Assumption holds after k iterations.

Show. Assumption holds after $k+1$ iterations.

Let P be a shortest s-v-path with at most $k+1$ edges. Let (w,x) be the last edge in P. From the proposition 12 (every subpath of the shortest path is also a shortest path) it follows that $P_{[s,w]}$ is the shortest s-w-path with at most k edges.

$$l(w) \leq c(P_{[s,w]})$$

Iteration $k+1$ will also analyze (w,x). Followingly it must hold that

$$l(x) \leq l(w) + c(w,x) \leq c(P_{[s,w]}) + c(w,x) = c(P)$$

From the proofs of both previous assumptions we conclude that at termination it holds that

$$\begin{aligned} l(x) = &\text{ length of the shortest } s\text{-}x\text{-path with at most } (n-1) \text{ edges} \\ = &\text{ length of the shortest s-x-path (conservative!)} \end{aligned}$$

Remark. (R,F) is the so-called "shortest paths tree".

□

10 Potential

Definition. Let $G = (V,W)$ be a digraph with $c: E \to \mathbb{R}$

A mapping $\pi : V(G) \to \mathbb{R}$ with $c(u,v) + \pi(u) - \pi(v) \geq 0 \;\forall\, (u,v) \in E(G)$ is called "potential".

Theorem 16. *Let G be a digraph with $c : E(G) \to \mathbb{R}$. A potential of (G,c) exists iff c is conservative.*

Proof.

$$\Rightarrow \exists \text{ potential } \pi \Rightarrow c \text{ is conservative}$$

Let k be a cycle

$$c(k) = \sum_{e \in E(k)} c(e) = \sum_{i=1}^{l} c(v_i, v_{i+1}) = \sum_{i=1}^{l} [c(v, v_{i+1}) + \pi(v_i) - \pi(v_{i+1})] \geq 0$$

$$\Leftrightarrow \text{ c is conservative } \Rightarrow \exists \text{ potential } \pi$$

Apply Moore-Bellman-Ford algorithm to $(\overline{G}, \bar{i})$ where $V(\overline{G}) = V(G) \,\dot\cup\, \{s\}$ and $E(\overline{G}) = E(G) \cup \{(s,v) : \;\forall\, v \in V(G)\}$.

$$\bar{c}(e) = \left\{ \begin{array}{cl} c(e) & e \in E(G) \\ 0 & e = (s,v) \;\forall\, v \in V(G) \end{array} \right\}$$

Let $l(v) \, \forall \, v \in V(\overline{G})$ be the output.

$$\overline{c}(v,w) + l(v) - l(w) \geq 0$$
$$l(w) \leq l(v) + c(v,w)$$

$\square$

Remark. The Moore-Bellman-Ford algorithm (as shown above) can be used to determine whether c is conservative and possibly to compute a potential.

Theorem 17. *Let $G = (V, E)$ be a digraph with $c : E(G) \to \mathbb{R}$. The Moore-Bellman-Ford algorithm can either determine a desired potential or find a negative cycle in $\mathcal{O}(m \cdot n)$.*

Proof. We apply Moore-Bellman-Ford algorithm to $(\overline{G}, \bar{i})$ as shown above and retrieve $l(v) \, \forall \, v \in V(G)$.

If l is a potential, we are done.

Otherwise $\exists (u, w) \in E(G)$ with $c(u,w) + l(u) < l(w) \Rightarrow l(u)$ was changed in the last iterations $\Rightarrow l(p(u))$ was changed in the last 2 iterations $\Rightarrow l(p(p(u)))$ was changed in the last 3 iterations. It holds that the length $l(\ldots)$ of each of these vertices was changed while the algorithm was running.

$$w, u, p(u), p(p(u)), p(p(p(u))), \ldots$$

Because $e(s)$ was not changed, $\{w, n, p(u), \ldots, p(\ldots p(u) \ldots)\}$ must contain a cycle. K in $F \Rightarrow c(k) < 0$ from invariant of b of the previous proof 3.3. $\square$

11 All Pairs Shortest Paths Problem

11.1 All pairs shortest paths problems (APSP)

Given. $G = (V, E)$ is a digraph. Weights $c : E(G) \to \mathbb{R}$ are conservative

Find. Find numbers $l_{s,t}$ and vertices $p_{s,t} \, \forall \, s, t \in V(G)$ such that $l_{s,t}$ is the length of a shortest s-t-path in G and $(p_{s,t}, t)$ is the last edge of the shortest path.

This is solvable with n repetition of the Moore-Bellman-Ford algorithm for each vertex: $\mathcal{O}(n^2 \cdot m)$ runtime or find potential π with Moore-Bellman-Ford algorithm

$$\overline{c}(u,v) := c(u,v) + \pi(u) - \pi(v) \geq 0 \, \forall \, (u,v) \in E(G)$$

For every s-x-path P in G it holds that

$$\overline{c}(P) = \sum_{i=1}^{l-1} \overline{c}(v_i, v_{i+1}) = \sum_{i=1}^{l-1}[c(v_i, v_{i+1}) + \pi(v_i) - \pi(v_{i+1})]$$

$$= \sum_{i=1}^{l-1} c(v_i, v_{i+1}) + (\pi(v_1) - \pi(v_2) + \pi(v_3) + \ldots + \pi(v_{l-1} - \pi(v_l)))$$

$$= c(P) = \pi(v_i) - \pi(v_i) = c(P) + \pi(s) - \pi(v)$$

$$\min C(P) = \min_{P:s\text{-}x\text{-path}} [E(P) - \pi(s) + \pi(x)]$$

$$= \min_{P:s\text{-}x\text{-path}} \overline{C}(P) + \pi(x) - \pi(s)$$

Apply Dijkstra's algorithm with $(G, \overline{c})$ for every vertex: $\mathcal{O}((m + n\log n) \cdot n)$ (best known complexity).

11.2 Floyd-Warshall algorithm

Algorithm 6 Floyd-Warshall algorithm

Given. $G = (V, E)$ is a digraph, $c : E(G) \to \mathbb{R}$ is conservative, $n = |V(G)|$
Find. Matrices $(l_{i,j})_{1\leq i,j\leq n}$ and $(p_{i,j})_{1\leq i,j\leq n}$ where $l_{i,j}$ and $p_{i,j}$

Let $l_{i,j} := c_{i,j} \;\forall\, (i,j) \in E(G)$
$l_{i,j} := +\infty \;\forall\, (i,j) \in (V(G) \times V(G)) \setminus E(G)$ with $i \neq j$
$l_{i,i} = 0 \;\forall\, i \in V(G)$
$p_{i,j} := i \;\forall\, (i,j) \in V(G)$
for $j = 1$ to n **do**
    for $i = 1$ to n **do**
        if $i \neq j$ **then**
            for $k = 1$ to n **do**
                if $k \neq i \wedge k \neq j$ **then**
                    if $(l_{i,k} > l_{i,j} + l_{j,k})$ **then**
                        $l_{i,k} := l_{i,j} + l_{j,k}$
                        $p_{i,k} := p_{j,k}$
                    end if
                end if
            end for
        end if
    end for
end for

Theorem 18. *The Floyd-Warshall algorithm works correctly and has a runtime of* $\mathcal{O}(n^3)$

Proof. Proving $\mathcal{O}(n^3)$ is trivial.

After execution of the most-outer loop j it holds that $\forall\, i,k : l_{i,k}$ is the length of the shortest i-k-path whose inner vertices are from $\{1,2,\ldots,j_0\}$ and where $(p_{i,k},k)$ is the last edge.

Induction over j_0.

Induction base. $j_0 = 0$ means considering only paths without inner vertices

Induction step.

Assumption. Hypothesis holds for $j_0 \in \{1,2,\ldots,n-1\}$
Show. Hypothesis holds for j_0+1. Execute outer loop with $j = j_0+1$

$$l_{i,k} > l_{i,j_0+1} + l_{j_0+1,k} \Rightarrow l_{i,k} = l_{i,j_0+1} + l_{j_0+1,k}, P_{i,k} = P_{j_0+1,k}$$

If P_{i,j_0+1} and $P_{j_0+1,k}$ are vertex-disjoint the assumption holds. But P_{i,j_0+1} and $P_{j_0+1,k}$ can only be vertex-disjoint otherwise we will get a contradiction.

Assumption. P and Q are not vertex-disjoint.

Remove the maximum (= inclusive maximal) closed walk from $P \cup Q$. This walk has weight ≥ 0 (as union of cycles with c are conservative). Maintain a i-k-path R with inner vertices from $\{1,2,\ldots,j_0\}$ and it holds that

$$C(R) + C(\text{closed walk}) = l_{i,j_0+1} + l_{j_0+1,k} \Rightarrow C(R) \leq l_{i,j_0+1} + l_{j_0+1,k} < l_{i,k}$$

$\square$

12 Cycles with minimal mean edge weight

This lecture took place on 21st of Oct 2014.

12.1 Minimal mean-cycle problem (MMC)

Given. Digraph $G_i \in t(G) \to \mathbb{R}$.

Find. Cycle k with mean edge weight minimizing

$$\frac{c(E(k))}{|E(k)|} \text{ or decide “A is acyclic”}$$

Remark. G is strongly connected $\Leftrightarrow$

$$\forall\, u,v \in V(G) \quad u \neq v$$

$$\exists \text{directed } u\text{-}v\text{-path and } \exists \text{ directed } u\text{-}v\text{-path}$$

Strongly connected components can be computed in $\mathcal{O}(n+m)$ with any searching algorithm in G. Without loss of generality: G is strongly connected, otherwise solve MMC in every strongly connected component.

Let s be a vertex in G such that $\exists$ s-v-path $\forall v \in V(G)$.

Theorem 19. *(Karp 1978.) Let G be a digraph with $c : E(G) \to \mathbb{R}$. Let $s \in V(G)$ such that $\forall v \in V(G) \setminus \{s\} \exists$ directed s-v-path in G.*

$$\forall x \in V(G) \, \forall K \in \mathbb{Z}_+ :$$

$$F_K(x) := \min \left\{ \sum_{i=1}^{k} c(v_{i-1}, v_i) : v_0 = s, v_k = x, (v_{i-1}, v_i) \in E(G), \ \forall 1 \leq i \leq k \right\}$$

If there is no sequence of edges of length k from s to x, then $F_K(x) = \infty$. Set $\mu(G,c)$ be the minimal mean edge weight of a cycle in (G,i) and $\mu(G,c) = \infty$ if G is acyclic. Then it holds that

$$\mu(G,c) = \min_{x \in V(G)} \max_{0 \leq k \leq n-1} \frac{F_n(x) - F_k(x)}{n-k}$$

Proof. 1.
If G is acyclic, $\mu(G,c) = \infty$.

Hypothesis.

$$\forall x \in V(G) : \max_{0 \leq k \leq n-1} \frac{F_n(x) - F_k(v)}{n-k} = \infty$$

Proof by contradiction.

$$\exists x \in V(G) \text{ with } \max_{0 \leq k \leq n-1} \frac{F_n(x) - F_k(x)}{n-k} < \infty$$

$$\Leftrightarrow F_n(x) \text{ is finite}$$

$$\Rightarrow \text{ edge repetition in order defined by } F_n(x)$$

This constitutes a cycle and we have a contradiction.

2. G is cyclic.

   (a) $\mu(G,c) = 0 \Rightarrow G$ does not have negative cycles. So c is conservative.

$$\Rightarrow F_n(x) \geq \text{dist}_{(G,c)}(s,x) = \min_{0 \leq k \leq n-1} F_k(x) = F_{k_0}(x)$$

$$\Rightarrow \max_{0 \leq k \leq n-1} \frac{F_n(x) - F_k(x)}{n-k} \geq 0 \, \forall x \in V(G)$$

We show $\exists x \in V(G)$ with

$$\max_{0\leq k\leq n-1} \frac{F_n(x) - F_k(x)}{n-k} = 0 = \frac{F_n(x) - F_{K_0}(x)}{n-k}$$

$$\Rightarrow F_n(x) = F_{K_0}(x) = \mathrm{dist}_{(s,x)}(G, c)$$

Let K be any cycle with $G)K = \infty$ in G and v is an arbitrary vertex in K. Let P be a shortest s-x-path in (G, c); c is conservative. Let P' be an edge sequence which starts with P and ends with n repetitions of K: $e(P') = c(P) = \mathrm{dist}_{(G,c)}(1, x)$. Let P'' be a subset of edges of P' which has exactly n edges. Let w be the final vertex of P''. Because P' is a shortest edge of s to $x \Rightarrow P''$ is the shortest edge sequence of s to w with n edges.

$$F_n(w) = \mathrm{dist}_{(G,c)}(s, w)$$

(b) $\mu(G, c) \neq 0$. Consider $c : E(G) \to \mathbb{R}$ with $c'(e) = c(e) - \mu(G, c)\ \forall\, e \in E(G)$.

$$\begin{aligned}
\mu(G, c') &= \min_{\text{cycle K in G}} \frac{c'(E(k))}{|E(k)|} \\
&= \min_{\text{cycle K in G}} \frac{c(E(k)) - |E(k)| \cdot \mu(G, c)}{|E(k)|} \\
&= \min_{\text{cycle K}} \left\{ \frac{c(F(K))}{|E(K)|} - \mu(G, c) \right\}
\end{aligned}$$

$$\mu(G, c') = \min_{\text{cycle K in G}} \frac{c(E(k))}{|E(k)|} - \mu(G, c) = \mu(G, c) - \mu(G, c) = 0$$

Let $F'_k(x)$ be analogous to $F_k(x)$ but computed with weights c'. It holds that $F'_K(x) = F_K(x) - K\mu(G, c)\ \forall\, x\ \forall\, 0 \leq k \leq n$.

$$\begin{aligned}
\mu(G, c') &= \min_x \max_{0\leq x\leq n-1} \frac{F'_n(x) - F'_k(x)}{n-k} \\
&= \min_x \max_{0\leq k\leq n-1} \frac{(F_n(x) - n\mu(G, c)) - (F_k(x) - k\mu(G, c))}{n-k} \\
&= \min_x \max_{0\leq k\leq n-1} \frac{F_n(x) - F_k(x)}{n-k} - \mu(G, c)
\end{aligned}$$

□

12.2 Algorithm for minimal mean cycle problem

Theorem 20. *The minimal mean cycle works correctly and can be implemented with a runtime of* $\mathcal{O}(n \cdot \max\{m, n\})$.

Algorithm 7 Minimal mean-cycle algorithm

Given. Digraph G, $c : E(G) \to \mathbb{R}$

Find. Cycle K with $c(k) = \mu(G, c)$ or "G is acyclic"

Insert $s \notin V(G)$ and all edges $\forall v \in V(G) : (s, v)$. Let $c(s, v) = 0 \; \forall v \in V(G)$. Let G' be this extended digraph.
Let $n = |V(G')|, F_0(s) = 0, F_0(x) = \infty \; \forall x \in V(G') \setminus \{s\}$.
for $k = 1$ to n **do**
    for all $x \in V(G)$ **do**
        $F_K(x) := \infty$
        for all $(|w|, x) \in \delta^-(x)$ **do**
            if $F_{K-1}(w) + c(w, x) < F_K(x)$ **then**
                $F_K(x) := F_{K-1}(w) + c(w, x)$
                $P_K(x) := w$
            end if
        end for
    end for
end for
if $F_n(x) := \infty \; \forall x \in V(G') \setminus \{s\}$ **then**
    return "G is acyclic"
end if
Let $x \in V(G)$ with $\max_{0 \leq k \leq n-1} \frac{F_n(x) - F_k(x)}{n-k} = \mu$ is minimal
Let K be a cycle which is contained in the sequence

$$x, \; p_n(x), \; p_{n-1}(x), \; p_{n-1}(p_n(x)), \; p_{n-2}(p_{n-1}(p_n(x))), \; \ldots$$

return k and μ

Proof. In G' there are only cycles contained which are also in G contained (there are no edges with end vertex s). So it is enough to prove that $\mu(G', c)$ is correctly computed.

1. In steps 2 and 3, $F_K(x)\ \forall 0 \leq k \leq n$ is computed correctly (proof is analogous to proof of Bellman-Ford-algorithm).
2. If algorithm terminates in step 4, then G is acyclic. Otherwise $\exists x in V(G)$ where $F_n(x)$ is finite $\Rightarrow$ no termination in step 4.
3. Consider (G', c') with $c'(e) = c(e) - \mu(G, c)\ \forall e \in E(G)$. Algorithm runs with (G', c') exactly the same like (G, c) only with $F'_k(x) = F_k(x) - \mu(G', c)\ \forall x\ \forall k$.
4. If we select some x in step 5, then $\mu(G', c') = 0 = \max_{0 \leq k \leq n} \frac{F_n(x) - F_K(x)}{n-k}$ is satisfied (follows from Karp's theorem). So $F_n(x) = \mathrm{dist}_{(G', c')}(s, x)$.
5. So every shortest edge sequence of n edges from s to x consists of a shortest s-x-path and a few cycles of length 0. The "last" cycle K will be found in step 5. It holds that

$$\frac{c'(E(k))}{|E(k)|} = \mu(G', c') = 0 \Rightarrow \frac{c(E(k))}{|E(k)|} = \mu(G', c)$$

□

13 Network flows

13.1 Definition

A *network* with source s and sink t is a quadruple (G, u, s, t) where

1. G is a digraph
2. $u : E(G) \to R_+$
3. $s, t \in V(G)$ with $s \neq t$

A *flow* f is a function $f : E(G) \to \mathbb{R}_+$ with $f(e) \leq u(e)\ \forall e \in E(G)$. An excess of a flow f in a vertex v is

$$\mathrm{ex}_f(v) := \sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e)\ \forall v \in V(G)$$

The *flow conservation condition* in $V \in V(G)$ is $\mathrm{ex}_f(v) = 0$. A flow f with $\mathrm{ex}_f(v) = 0\ \forall v \in V(G)$ is called *circulation.* A s-t-flow is a flow with $\mathrm{ex}_f(s) \leq 0$ and $\mathrm{ex}_f(v) = 0\ \forall v \in V(G) \setminus \{s, t\}$. The value of a s-t-flow f is $\mathrm{value}(f) := -\mathrm{ex}_f(s)$.

13.2 Maximum flow problem (MF)

Given. network (G, u, s, t) with source s and sink t

Find. a s-t-flow with maximum vale

13.3 Example: Job assignment problem

Given. n jobs, n workers, $S_i \subseteq \{1, 2, \ldots, m\}$ is the set of workers which complete job $i \; \forall \, i \in \{1, 2, \ldots, n\}$. t_i is the time it takes to complete job $i \; \forall \, i \leq i \leq n$.

Find. Which part of which jobs should be complete by which worker to achieve maximum efficiency?

Additional assumption:

1. All workers are equally efficient.
2. One job at a time per worker.
3. Jobs can have multiple simultaneous workers.

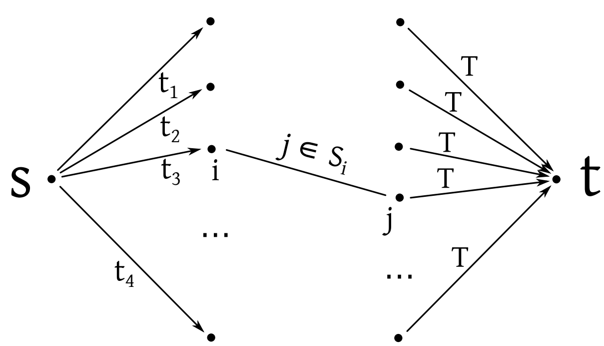


Figure 6: Job assignment problem

We create a graph with three layers. We have one $s$ource which is connected to every job. Every job is connected to every worker that can complete this job. All workers are connected to one destination t.

$$u((s, i)) = t_i \quad i \in \{1, 2, \ldots, n\}$$
$$u((i, j)) = T \; \forall \, i, j \text{ with } j \in S_i$$
$$u(j, t) = T \; \forall \, j \in \{1, 2, \ldots, n\}$$

where T is a configuration parameter.

Assuming f_T is the maximum flow from s to t in (G, u, s, t). If value$(f_T) = \sum_{i=1}^{n} t_i$. We claim: If worker j completes portion $f(i,j)$ of job j $\forall i, j$ then all jobs are completed within time T

$$[0, T_{\max}] \sum t_i = T_{\max}$$

This lecture took place on 27th of Oct 2014.

Assignment:

1. Worker j should complete $f_{i,j}$ units of work of the i-th job $\forall\, i\, \forall\, j$.

$$\text{value}\,(f) = \sum_{i=1}^{n} t_i \quad \Rightarrow \quad f(s,i) = t_i \;\forall\, \text{job } i$$

$$\underbrace{\Rightarrow}_{\text{flow conservation condition}} \sum_{j \in S_i} f_{i,j} = f(s,i) = t_i$$

   hence job i will be complete $\forall\, i$.

2. Why is everything completed *in time* (ie. $\leq T$)?

$$\forall\, \text{worker } j : \quad \underbrace{\sum_{i, j \in S_i} f(i,j)}_{\text{total working time of worker i}} = f(j,t) \leq T$$

13.4 Maximum-flow problem (cont.)

Linear programming definition of the maximum-flow problem

Given. A network (G, u, s, t). Let x_e be the flow carried by edge e in the network $\forall\, e \in E(G)$ with $0 \leq x_e \leq u_e$.

$$\sum_{e=(i,v)} x_e - \sum_{e=(v,j)} x_e = 0 \;\forall\, v \in V(G) \setminus \{s, t\}$$

Find. Compute the maximum flow $\sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e$.

We are looking for a combinatorial solution.

Remark. MFP is polynomially solvable as linear program eg. via ellipsoid method.

Theorem 21. *MFP always has an optimal solution. Linear programming always provides an optimal solution and is limited by* $\sum_{e \in E(G)} u_e$.

Definition. A *s-t-cut in G* is an edge set $\delta^+(X)$ with $X \subsetneq V(G), s \in X, t \notin X$. The set of all edges going from X to X^c.

The capacity of $\delta^+(x)$ is $u(\delta^+(x)) := \sum_{e \in \delta^+(x)} u_e$. A minimal s-t-cut is a s-t-cut with smallest possible capacity.

Theorem 22. *$\forall A \subsetneq V(G)$ with $s \in A, t \notin A$ and for every s-t-flow it holds that:*

1. $\text{value}\,(f) = \sum_{e \in \delta^+(A)} f(e) - \sum_{e \in \delta^-(A)} f(e)$
2. $\text{value}\,(f) \leq \sum_{e \in \delta^+(A)} u_e$

Proof.

- Firstly,

$$\text{value}(f) = \sum_{e \in \delta^+(s)} f_e - \sum_{e \in \delta^-(s)} f_e = \sum_{v \in A} \left(\underbrace{\sum_{e \in \delta^+(v)} f_e - \sum_{e \in \delta^-(v)} f_e}_{=0\ \forall\ v \neq s} \right)$$
$$= \sum_{e \in \delta^+(A)} f_e - \sum_{e \in \delta^-(A)} f_e$$

  All edges in A (that stay in A) cancel each other out (incoming $= +1$, outgoing $= -1$).

- Secondly,

$$\sum_{e \in \delta^+(A)} f_e - \sum_{e \in \delta^-(A)} f_e \leq \sum_{e \in \delta^+(A)} u_e = u(\delta^+(A))$$

$\square$

Remark. From theorem 22 (2) it follows that

$$\underbrace{\max \{\text{value}(f)\}}_{\text{f: } s\text{-}t\text{-flow}} \leq \underbrace{\min u(\delta^+(A))}_{\text{A: } s\text{-}t\text{-cut}}$$

Definition. *(dt. "Inkrementnetzwerk", residual network)* Assume a network (G, u, s, t) and a s-t-flow f. Define the set of edges as a multiset:

$$\overleftrightarrow{E} = \{e : e \in E(G)\} \cup \{\overleftarrow{e} := (j, i) : (i, j) \in E(G)\}$$

In words "all edges and edges in opposing directions including duplicates". We now consider the new graph

$$\overleftrightarrow{G} = (V(G), \overleftrightarrow{E})$$

$$G_f := (V(G), \left\{e \in \overleftrightarrow{E} : u_f(e) > 0\right\})$$

where $u_f(e)$ is remaining capacity defined as

$$u_f(e) = \begin{cases} u_e - f_e & e \text{ is edge in direction forward} \\ f_{\overleftarrow{e}} & e \text{ is edge in direction backward } (f_{\overleftarrow{e}} = f_e) \end{cases}$$

(G_f, u_f, s, t) is then called "residual network".

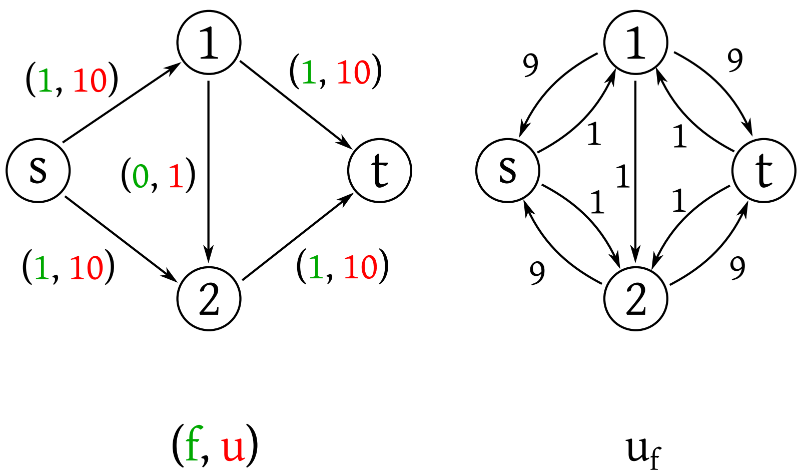


Figure 7: Residual network

Remark. $2 \to 1$ was not introduced in G_f, because $u_f = 0$

A directed s-t-path in G_f is called *augmented path*.

How do you augment a path?

$$f'(e) = \begin{cases} f(e) + \delta & e \in P \cap E^+ \\ f(e) - \delta & e \in P \cap E^- \\ f(e) & e \notin P \end{cases}$$

$$f' := f \oplus P \qquad \text{with } \delta := \min_{e \in E(P)} u_f(e)$$

f' is called augmented path along P.

$$\begin{array}{ll} P \cap E^+ & \text{forward edge in path} \\ P \cap E^- & \text{backward edge in path} \end{array}$$

Show:

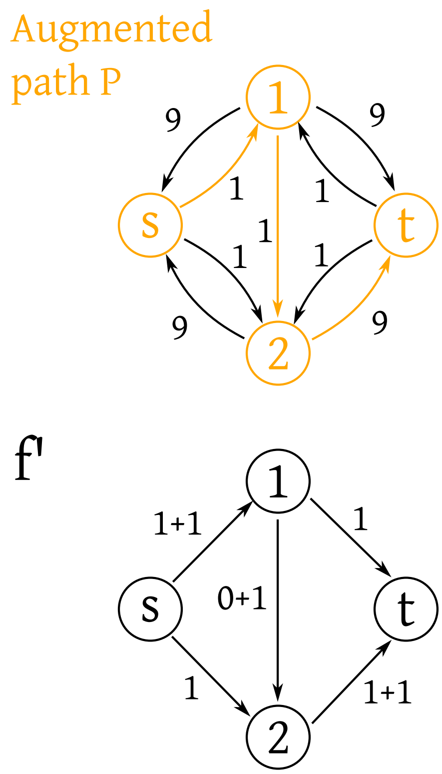


Figure 8: Augmented path

1. f' is a flow
2. $\text{value}(f') = \text{value}(f)$

Let $e \in P \cap E^+$.

$$0 \leq f'(e) \leq u(e)$$
$$0 \leq f(e) + \delta \leq u(e)$$
$$0 \leq f(e) + \delta \leq f_e + u_e - f(e)$$

Let $e \in P \cap E'^-$.

$$0 = f(e) - f(e) \leq f'(e) = f(e) - \delta \leq u(e)$$

$$\text{value}(f') = \sum_{e \in \delta^+(s)} f'(e) - \sum_{e \in \delta^-(s)} f'(e) = f'(e_0) + \sum_{e \in \delta^+(s)} f'(e) - \sum_{e \in \delta^-(s)} f'(e)$$

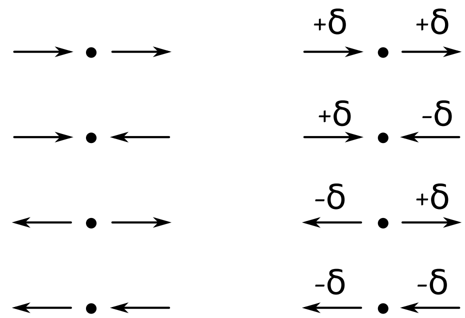


Figure 9: f' is a flow

Let $e_e \in \delta^-(s) \cap P = \text{value}(f) + \delta > \text{value}(f)$.

Flow augmented by δ along path P such that the value increases.

Remark. If f is a max s-t-flow, then there is no s-t-path in G_f. If there would be a s-t-path P then $f' = f \oplus P$ is better than f (contradiction).

Theorem 23. *Let (G, u, s, t) be a network and f be a flow. If there is no s-t-path in G_f, then f is optimal. Hence* value(f) *is at maximum.*

Proof. Let $A = \{v \in V(G) : v \text{ is reachable from s in } G_f\}$. A defines a cut between a set A containing s and a set B containing t with $t \notin A$. An edge on the border of those sets must satisfy $f(e) = u(e)$ (*saturating edge*), otherwise $u(e) - f(e) > 0$ and the edge would be part of G_f.

A defines a s-t-cut $\delta^+(A)$.

$$u(\delta^+(A)) = \sum_{e \in \delta^+(A)} u_e$$

$$\text{value}(f) = \underbrace{\ldots}_{\text{theorem 22 (a)}} = \sum_{e \in \delta^+(A)} f(e) - \sum_{e \in \delta^-(A)} f(e)$$

So this end going from set B to set A has $f(e) = 0$ otherwise $\overleftarrow{e}$ is in G_f. It follows that $u(\delta^+(A)) = \text{value}(f)$.

So f is optimal (assumption that cut $\geq$ flow, hence equality, won't get better) $\square$

Theorem 24. *(*Max flow, min cut problem*, Ford & Fulkerson, 1956) Let (G, u, s, t) be a network than there exists a maximal s-t-flow f and a minimal cut (s-t-cut) $\delta^+(A)$ with* $\text{value}(f) = u(\delta^+(A))$*. Especially the value of a maximal flow and the capacity of a minimal s-t-cut is equal.*

13.5 Algorithm by Ford & Fulkerson

Proof. Directly follows from theorems 22 and 23. □

Algorithm 8 Algorithm by Ford & Fulkerson

Given. G, u, s, t
Find. max s-t-flow

Set $f(e) = 0 \; \forall \, e \in E(G)$
while $\exists$ s-t-path P in G_f **do**
  $f = f \oplus P$
end while
return f

As a result many steps are required. Better paths would be $(s, 2, t)$ or $(s, 1, t)$ instead of using 1–2.

Conclusions of F&F algorithm

If $u_0 \in \mathbb{Z}_+$ then F&F algorithm terminates. After at most $U(|V(G)| - 1)$ iterations, where $U = \max_{e \in \delta^+(s)} u_e$, because

1. Firstly

$$\text{value}(f) \leq \sum_{e \in \delta^+(s)} f(e) \leq \sum_{e \in \delta^+(s)} U \leq U \cdot (|V(G)| - 1)$$

2. Secondly the flow value is increased by δ in every iteration; where $\delta \geq 1$ because $\delta > 0$ and $\delta \in \mathbb{Z}$.

If $u_e \in Q_+$ then affiliation to $u_e \in \mathbb{Z}_+$ (multiply all capacities with common denominator).

If $u_0 \in Q_+$ then F&F algorithm must not terminate. It also must not converge and if it does converge, it must not converge towards optimality. So the algorithm might create a series f_n with $\text{value}_{n \to \infty}(f) \nrightarrow \text{opt}$.

In the algorithm the series converges only if it converges towards optimality.

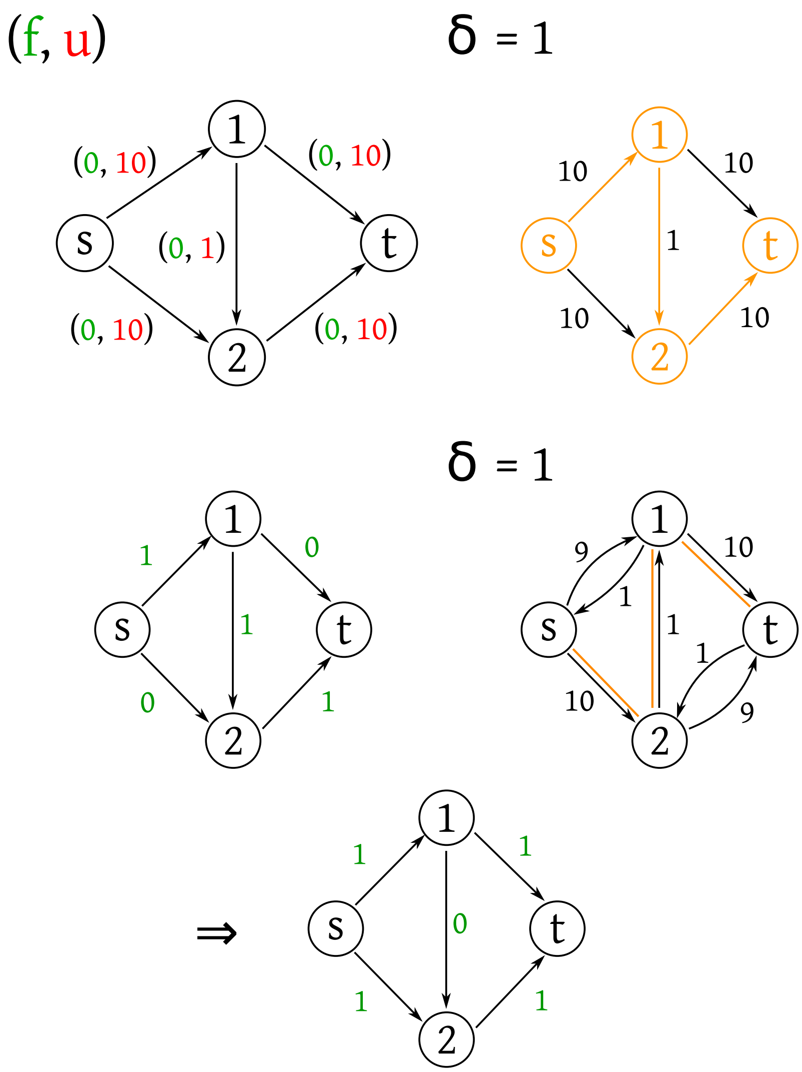


Figure 10: Ford & Fulkerson algorithm

This lecture took place on 28th of Oct 2014.

13.5.1 Analysis

Runtime per iteration:	$\mathcal{O}(n)$ determination of a s-t-path (eg. breadth-first search) if such one exists
Number of iterations:	$\mathcal{O}(U(n-1)) = \mathcal{O}(Un)$

Total runtime (for $u_e \in \mathbb{Z}_+ \; \forall \, e \in E(G)$:

$$\mathcal{O}(mnU)$$

pseudo-polynomial runtime

Theorem 25. ***Flow decomposition theorem*** *(Galler 1956, Ford and Fulkerson 1962) Let (G, u, s, t) be a network and f be a s-t-flow. Then $\exists$ a family $\mathcal{P}$ of s-t-paths and a family $\mathcal{C}$ of cycles in G and the weights in $\mathcal{P} \cup \mathcal{C} \to \mathbb{R}_+$ $(P \mapsto w(P), C \mapsto w(C))$ such that*

$$f(e) = \sum_{P \in \mathcal{P} \cup \mathcal{C} : e \in E(P)} w(P) \; \forall \, e \in E(G)$$

$$\text{value}(f) = \sum_{p \in \mathcal{P}} w(P) \quad \textit{and} \quad |\mathcal{P}| + |\mathcal{C}| \leq |E(G)|$$

Proof. Induction on number of flow-carrying edges, ie. $|e \in E(G) : f(e) > 0|$.

Hypothesis. *No flow-carrying edge: trivial (select some s-t-path and sort by value)*

Induction step.

Let $i = 0$ and $(V_0, w_0) \in E(G)$ otherwise $f(v_0, w_0) > 0$. If $w_0 \neq t \exists (w_0, w_1) \in E(G)$ with $f(w_0, w_1) > 0$. If $w_1 \notin \{t, v_0, w_0\}$ then $i = i + 1$ and repeat this step until t is reached or some vertex is repeated (cycle case).

Analogously you can think of $(v_1, v_0) \in E(G)$ with $f(v_1, v_0) > 0$. If $v_1 \notin \{s, v_0, w_0, w_1, \ldots, w_k\}$ then repeat the step until s is reached or some cycle gets created.

This construction provides either a flow-carrying s-t-path P or a flow-carrying cycle C. Case distinction:

1. $w(P_2) = \min_{l \in E(P_0)} f(e)$ and

$$f(e) = \begin{matrix} f(e) - w(P) & e \in E(P), \forall e \in E(G) \\ f(e) & e \notin E(P) \forall e \in E(G) \end{matrix}$$

2. $w(C_2) = \min_{e \in E(C)} f(e)$ analogously.

f' is a s-t-flow with at least one flow-carrying edge less. With the induction hypothesis f' can be decomposed, hence $\exists \mathcal{P}_1 (\mathcal{C}_1)$ family of s-t-paths (cycles) and $w : \mathcal{P}_1 \cup \mathcal{C}_1 \to \mathcal{R}_+$ with $f'(e) = \sum_{P \in \mathcal{P}_1 \cup \mathcal{C}_1, e \in E(P)} w(P)$ and value$(f') = \sum_{p \in \mathcal{P}_1} w(P)$ and $|\mathcal{P}_1 \cup \mathcal{C}_1| \leq E(G)$.

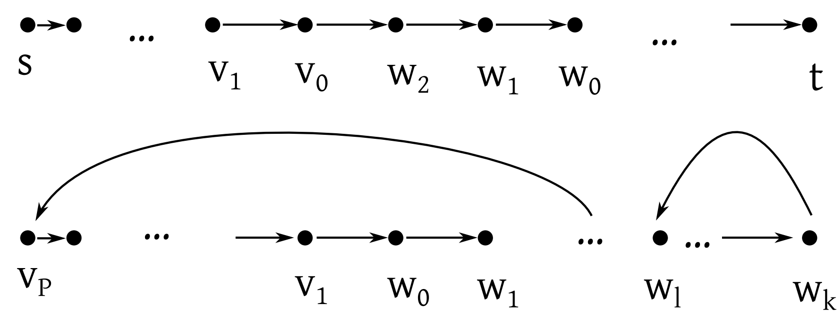


Figure 11: Flow decomposition theorem

1.
$$\mathcal{P} := \mathcal{P}_1 \cup \{P_0\}, \mathcal{C} := \mathcal{C}_\infty$$
$$\underbrace{f(e) = f'(e) + w(P_0)}_{\text{if } e \in E(P)} = \sum_{P \in \mathcal{P}_1 \cup \mathcal{C}_1, e \in E(P)} w(P) + w(P_0) = \sum_{P \in \mathcal{P} \cup \mathcal{C}, e \in E(P)} w(P)$$

2.
$$\mathcal{P} := \mathcal{P}_1, \mathcal{C} := \mathcal{C}_\infty \cup \{C_0\} \text{ with corresponding weights}$$
$$\text{value}(f) = \text{value}(f') + w(P_0) = \sum_{P \in \mathcal{P}_1} w(P) + w(P_0) = \sum_{P \in \mathcal{P}} w(P)$$

3. $|\mathcal{P} \cup \mathcal{C}| \leq |E(G)|$ because the induction step will be executed at most $|E(G)|$ times. In every step a new non-flow-carrying edge is created and in every step i $(\mathcal{P}_i \cup \mathcal{C}_i)$ is incremented by one and we can start with $\mathcal{P} \cup \mathcal{C} = \varnothing$ for flow $f \equiv 0$.

$\square$

13.6 Edmonds and Karp algorithm (E&K)

This algorithm distinguishes from F&F only by the selection of the s-t-path: In every iteration i a shortest s-t-path P_2 in G_{f_i} is selected (shortest in terms of number of edges).

Theorem 26. *Let $f_0, f_1, \ldots, f_k, \ldots$ be a sequence of flows created by the E&K algorithm, where $f_{i+1} = f_i + P_i$ and P_i is a shortest s-t-path in G_{f_i} $\forall i$. Then it holds that*

Algorithm 9 Edmonds and Karp algorithm

Given. G, u, s, t

Find. max s-t-flow

1: Let $f(e) = 0 \; \forall \, e \in E(G)$
2: Determine a shortest f-augmenting path P
3: **if** P does not exist **then**
4:     **return** f
5: **end if**
6: Determine $\gamma = \min_{e \in E(P)} u_f(e)$. Augment f along P by γ
7: **go to** 2

- $|E(P_k)| \leq |E(P_{k+1})| \;\; \forall \, i$
- $|E(P_k) + z \leq |E(P_r)||$ *for all* $k < r$ *such that* $P_k \cup P_r$ *contains at least one pair of edges of opposing direction.*

Proof. 1. Let $G_k := (V, E_k)$ where $E_k = P_k \cup P_{k+1} \setminus \{\text{pairs of edges in opposing direction}\}$. It holds that $E_k \subseteq E(G_{f_k})$ (where G_{f_k} is an incremental network to f_k), because $P_k \cup P_{k+1} \subseteq E(G_{f_k}) : P_k \subseteq E(G_{f_k})$ is obvious.

Assumption. $\exists e \in P_{k+1} \setminus E(G_{f_k}) : e \in E(G_{f_{k+1}})$. Hence e was added to the incremental network in the k-th iteration.

2. So e is an edge of opposing direction of an edge of P_k. This is a contradiction because $e \notin E_k$. So $E_k \subseteq E(G_{f_k})$ holds.

   Add two artificial edges (t, s) to E_k ($G_k = (V, E_k)$) and retrieve $\bar{G}_k$. $\bar{G}_k$ is eulerian (in a directed sense), ie. $\deg^-_{\hat{C}_k}(v) = \deg^-_{\hat{C}_k}(v) \; \forall \, v$. $\Rightarrow \hat{G}_k$ can be decomposed into edge-disjoint cycles. Let $C_1 (C_2)$ be the cycles that contain the artificial edges 1 and 2.

   Let P_1 and P_2 be the edge-disjoint s-t-paths in those cycles C_1 and C_2.

   P_1 and P_2 are in $\hat{G}_k$ and do not require any artificial edges.

$$P_1 \text{ and } P_2 \in E(G_k) \subseteq E(G_{f_k})$$

$$\begin{gathered} 2\,|E(P_k)| \leq |E(P_1)| + |E(P_2)| \\ \leq |E(G_k)| \leq |E(P_k)| + |E(P_{k+1})| \\ \Rightarrow |E(P_k)| \leq |E(P_{k+1})| \end{gathered}$$

3. It suffices to show this statement for $k < r$, where P_r and P_i contain no edges of opposing direction $\;\forall \, k < i < r$.

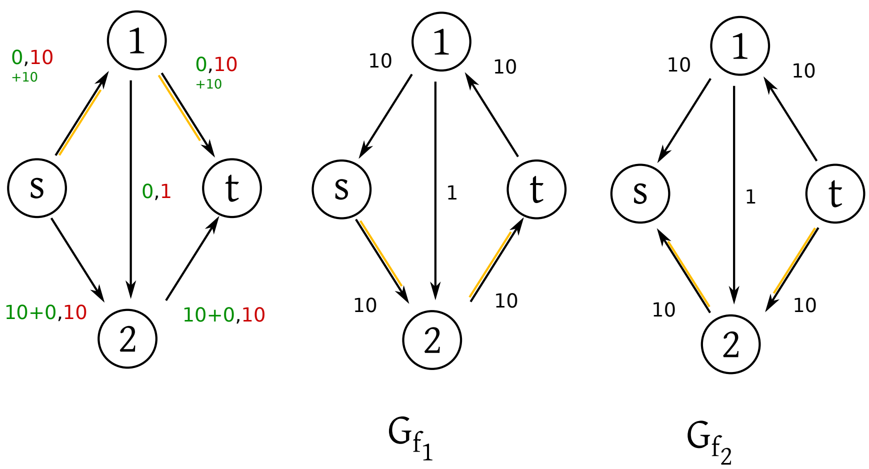


Figure 12: Drawing for EK algorithm

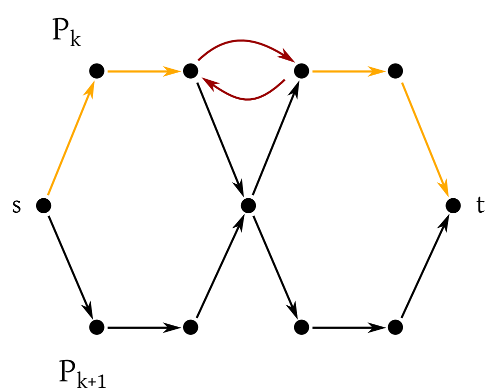


Figure 13: Drawing for EK algorithm

Reasoning. Let P_{k+1} be the last path $P_k, P_{k+1}, \ldots, P_{k+i}, \ldots, P_r$ with edges of opposing direction to P_r. From $|E(P_{k+1})| + 2 \subseteq |E(P_r)|$ it follows (with $|E(P_k)| \leq |E(P_{k+1})|$):

$$|E(P_k)| + 2 \leq |E(P_{k+1}) + 2 \leq |E(P_r)||$$

We assume without loss of generality that P_k and P_r contain edges of opposing direction but not $P_i, P_r \ \forall\, k < i < r$. Construct G_k with all edges from $P_k \cup P_r$ without pairs of edges of opposing direction. It holds that $E(G_k) \subseteq E(G_{f_k})$ because otherwise $\exists e \in P_r : |E(G_{f_k})|$ hence e is not contained in $E(G_{f_k})$, but in $E(G_{f_r})$. Hence e was added in iterations k, $k+1$, …, $r-1$. So e has an edge of opposing direction in P_k, P_{k+1}, …, P_{r-1}.

$\square$

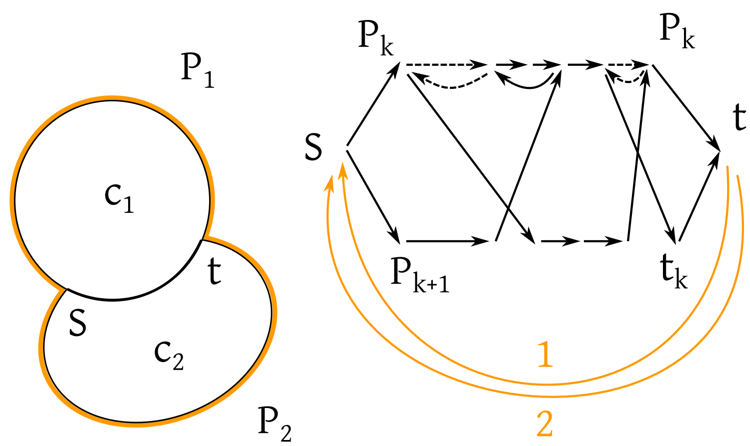


Figure 14: Edmonds and Karp algorithm

Analogously as in case a we find 2 edge-disjoint s-t-paths P_1 and P_2 in G_{f_k}:

$$2\,|E(P_k)| \leq |E(P_1)| + |E(P_2)| \leq |E(G_k)| \leq |E(P_k)| + |E(P_r)| - 2$$

$$\Rightarrow |E(P_k)| + 2 \leq |E(P_r)|$$

Theorem 27. *(Edmonds and Karp, 1972) The algorithm of Edmonds and Karp requires at most $\frac{nm}{2}$ augmented paths (equals to the number of iterations) and determines a maximum flow correctly. The algorithm has a runtime complexity of $\mathcal{O}(m^2 \cdot n)$.*

Proof. An edge $e \in P$ (where P is an augmented path in G_f) is called *bottleneck edge* if

$$\min_{g \in E(P)} u_f(g) = u_f(e)$$

Question. How often can some edge e occur as bottleneck edge during the algorithm run?

Let e be a bottleneck edge in P_k. Then it holds that $e \notin E(G_{f_{k+1}})$. If the edge becomes a bottleneck edge again of P_k with $e > k$, then the edge goes into the residual network; hence the edge is of opposing direction of another edge of the previous augmented path; without loss of generality to an edge of P_k:

$$|E(P_k)| + 2 \leq |E(P_e)|$$

Introduction of e as bottleneck $\Rightarrow$ augmented path gets extended.

$$\begin{gathered} \text{at most } n-1 \text{ extensions} \\ \Rightarrow \text{per } e \text{ at most } n-1 \text{ bottleneck edges} \\ \Rightarrow \text{number of iterations: } \mathcal{O}(m \cdot n) \end{gathered}$$

□

This lecture took place on 3rd of November 2014.

13.6.1 Runtime analysis

The number of iterations is $\mathcal{O}(nm)$ because every edge $\mathcal{O}(n)$ becomes a bottleneck at one point in time. A more precise upper bound is $\frac{n}{4}$.

Let k be the iteration in which $e = (v, w)$ becomes a bottleneck edge. Then let $r > k$ be the index of the next iteration in which e becomes a bottleneck edge again.

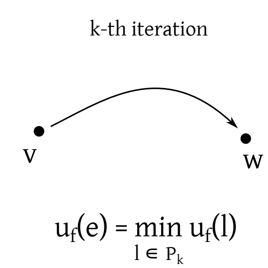


Figure 15: Edmonds and Karp algorithm – Bottlenecks

$$\begin{gathered} k+1 \text{ iteration with } e \notin G_{f_{k+1}} \\ \Rightarrow \quad \overleftarrow{e} \in G_{f_p} (k+1 \leq p) \end{gathered}$$

…must be satisfied and flow must go through $\overleftarrow{e}$ such that e can recur in G_f.

$$E(P_p) \geq E(P_k) + 2$$
$$r \text{ iterations} \Rightarrow E(P_r) \geq E(P_p) + 2$$
$$e \in G_{f_r} : E(P_r) \geq E(P_k) + 4$$

Every time when e becomes a bottleneck edge, the length of the shortest s-t-path by at least 4. So e is at most $\frac{n}{4}$ times a bottleneck edge.

$$\#\text{iterations} \leq 2m\frac{n}{4} = \frac{mn}{2}$$
$$(|E| \cup |\overleftarrow{E}| \leq 2m)$$

This bound is the best we can achieve. In some cases this bound is reached.

Total runtime: $\mathcal{O}(m)$ per iteration $\times \mathcal{O}(mn)$ iterations $= \mathcal{O}(m^2 n)$

13.7 Blocking flows and Dinitz's algorithm (1970)

Given. Network (G, u, s, t) and s-t-flow f. Its level graph G_f^L is a subgraph of G_f with

$V(G_f^L) := V(G_f), E(G_f^L) := \{e = (x, y) \in E(G_f) : \text{dist}_{G_f}(s, x) + 1 = \text{dist}_{G_f}(s, y)\}$

Observation. G_f^L is ayclic. $v_0 \to v_1 \to v_2 \to v_k \to v_0$. $\text{dist}(v_0) = \text{dist}(v_0) + k$ is a contradiction.

Constructable with $\mathcal{O}(m)$ runtime (eg. BFS).

Definition. s-t-flow f is called *blocking*, if graph $(V(G), \{e \in E(G_F^L) : f(e) < u(e)\})$ does not contain a s-t-path.

Not every blocking flow is optimal.

- First example:

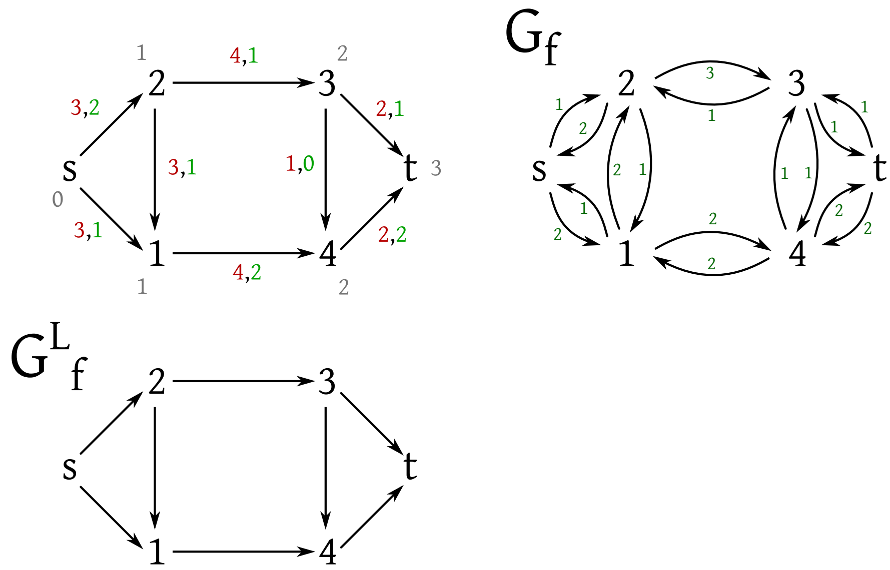


Figure 16: Example (a) graphs

$$\exists s\text{-}t\text{-path in } G_f^L \Rightarrow f \text{ is non-blocking}$$

- Second example:

$$\nexists s\text{-}t\text{-path} \Rightarrow f \text{ blocking, but there is a } s\text{-}t\text{-path in } G_f \Rightarrow \text{non-optimal}$$

$$\text{in } \left(V(G), \left\{e \in E(G_f^L) : f(e) < u(e)\right\}\right)$$

Recall that

$$|E(P_{k+1})| \geq |E(P_k)|$$
$$E(G_k) \subseteq E(G_{f_k})$$

Theorem 28. *Dinitz' algorithm finds a maximum flow in $\mathcal{O}(n^2 m)$ runtime.*

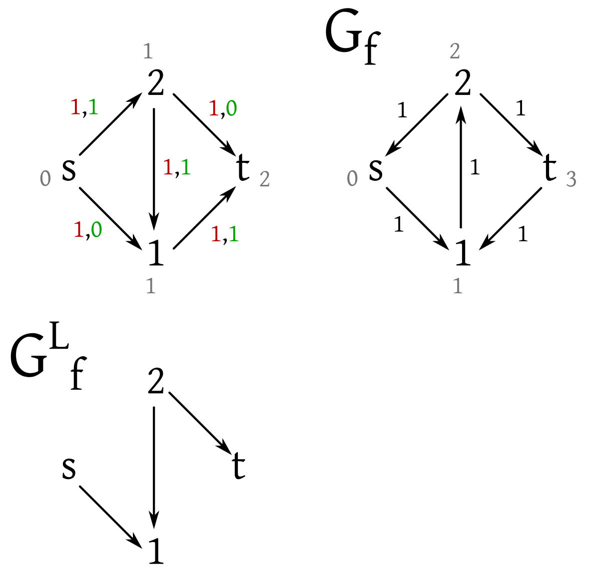


Figure 17: Example (b) graphs

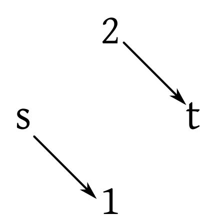


Figure 18: Example (b) schematic

Proof. **Runtime analysis.** Number of iterations: $\mathcal{O}(n-1)$

Because length of shortest s-t-path increases after every iteration by at least 1. This is analogous to proof of theorem 24: The blocking flow "contains" all shortest paths of same length.

Determination of a blocking flow in G_f^L in $\mathcal{O}(nm)$ runtime (see practicals)

$$\mathcal{O}(n^2m)$$

Correctness. Blocking flow $= 0$ in $G_f^L \Rightarrow$ no s-t-path in $G_f \Rightarrow f$ is optimal.

Algorithm 10 Dinitz's algorithm

Given. Network (G, u, s, t)

Find. s-t-flow f with maximum value

1: Let $f(e) := 0 \; \forall \, e \in E(G)$

2: Build level-graph G_f^L (subgraph of G_f)

3: Determine a blocking flow f' in G_f^L. If $f' = 0$, then stop "f is optimal".

4: Augment $f \oplus f'$, goto step 2

$$f \oplus f'(e) = \begin{cases} f(e) + f'(e) & e \in E(G) \\ f(e) + f'(e) & e \in \overleftarrow{E}(G) \end{cases} \; \forall \, e \in E(G)$$

(Sleator & Tarjan, 1984). $\mathcal{O}(m \log n)$ for determination of a blocking flow in a level graph. □

13.8 Goldberg & Tarjan: Push-Relabel algorithm

Algorithm by Goldberg & Tarjan (1988).

Definition. Let (G, u, s, t) be a network with source s and sink t. $f : E(G) \to \mathbb{R}_+$ is called *pre-flow* if $f(e) \leq u(e) \; \forall \, e \in E(G)$ and $\mathrm{ex}_f(v) \geq 0 \; \forall \, v \in V(G) \setminus \{s\}$. A vertex $v \in V(G) \setminus \{s, t\}$ is called *active* if $\mathrm{ex}_f(v) > 0$.

Idea. Work with preflow f satisfying $\exists$ s-t-path in G_f and try iteratively to reduce the excess for active vertices until no more active vertices exist.

Definition. Let (G, u, s, t) be a network. A mapping $\psi : V(G) \to \mathbb{Z}_+$ is called *distance marker* if $\psi(t) = 0$ and $\psi(s) = n$ and $\psi(v) \leq \psi(w) + 1 \; \forall \, (v, w) \in G_f$.

An edge $e = (v, w) \in E \cup \overleftarrow{W}$ is called *permitted edge* if $\psi(v) = \psi(w) + 1$.

Observation. If ψ is a distance marker then $\psi(v)$ is a lower bound for the number of edges in a shortest v-t-path in G_f.

$$v_0 \to v_1 \to \cdots \to v_k = t$$

$$\psi(v_0) \leq \psi(v_1) + 1 \leq \psi(v_2) + 1 + 1 \leq \ldots \leq \underbrace{\psi(v_k)}_{\psi(t)=0} + k = k$$

Algorithm applies push or relabel operations. Starts with preflow which saturates all edges (s, v) $(f(s, v) = u(s, v)) \Rightarrow$ in G_f there is no outgoing edge from $s \Rightarrow \nexists$ s-t-path

$$f(e) = 0 \text{ if } e \cap \{s\} = \varnothing$$

$$f(v, s) = 0$$

Has active vertex? If not, then done (optimality criterion satisfied)

Algorithm 11 Push-Relabel algorithm (book: page 201)

Given. G, u, s, t

Find. Maximum s-t-flow f

```
f(e) := u(e) ∀ e ∈ δ⁺(s)
f(e) := 0 ∀ e ∈ E(G) \ δ⁺(s)
ψ(s) := n := |V(G)| and ψ(v) := 0 ∀ v ∈ V(G) \ {s}
while there exists an active vertex do
    Let v be an active vertex
    if no e ∈ δ⁺_{G_f}(v) is a permitted edge then
        Relabel
    else
        Let e ∈ δ⁺_{G_f}(v) be some permitted edge
        Push(e)
    end if
end while
```

Push(e)

```
Let γ = min {ex_f(v), u_f(e)} where e starts in v
Augment f along e in γ
```

Relabel(v)

```
Let ψ(v) := min {ψ(w) + 1 : (v, w) ∈ δ⁺_{G_f}(v)}
```

Condition for $\psi(v) \leq \psi(w) + 1$ is only satisfied for edges in G_f.

$$\begin{array}{ll} \text{permitted:} & \psi(v) = \psi(w) + 1 \\ \text{not permitted:} & \psi(v) < \psi(w) + 1 \end{array}$$

Theorem 29. *The push-relabel algorithm has two invariants:*

- *f is always an s-t-preflow*
- *ψ is always a corresponding distance marker*

Proof. After initialization both invariants are given (trivial case).

PUSH of edge (v, w)

$$f'(e) = f(e) + y$$

$$\text{ex}_{f'}(v) = \text{ex}_f(v) - y \geq 0$$

$$\text{ex}_{f'}(w) = \text{ex}_f(w) + y \geq 0$$

Distance marker: If (v, w) is cancelled, condition is still satisfied.

If (w, v) is added, then $\psi(w) \leq \psi(v) + 1$ but PUSH only if permitted: $\psi(v) = \psi(w) + 1$.

RELABEL does not influence f, stays a preflow

$$\psi(v) = \min\left\{\psi(w) + 1 : w \in \delta^+(v)\right\}$$

$$\Rightarrow \psi(v) \text{ increases}$$

G_f does not change. Inequations can only be invalidated for edges.

$v \to w$	$\psi(v) \leq \psi(w) + 1$	$\psi(v)$ is never increased too much
$w \to v$	$\psi(w) \leq \psi(v) + 1$	$\psi(v)$ is larger (which is fine)

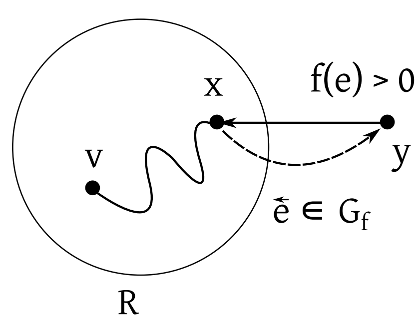
□

This lecture took place on 4th of November 2014.

Theorem 30. *Let f be a preflow and ψ be a distance marker in regards of f. Then the following statements hold:*

1. *s is reachable from every active vertex v in G_f.*
2. *If $v, w \in V(G)$ with w being reachable from v in G_f, then $\psi(v) \leq \psi(w) + n - 1$*
3. *t is not reachable in G_f*

Proof. • Part 1 of proof: Let v be active. Let R be the set of vertices reachable vertices in G_f from v. The following does not exist:



Observation. $\forall\, e = (y, x) \in \delta^-(R)$ it holds that $f(e) = 0$. Otherwise $y \in R$ which contradicts.

$$\mathrm{ex}_f(w) = \sum_{e \in \delta^-(w)} f(e) - \sum_{\delta^+} f(e)$$

$$\sum_{w\in R} \mathrm{ex}_f = \sum_{e\in\delta^-(R)} f(e) - \sum_{e\in\delta^+(R)} f(e) \leq 0$$

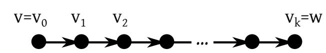
$$f(e) = 0 \text{ in case } e \in \delta^-(R)$$

$$\text{For } v \text{ it holds that } \mathrm{ex}_f(v) > 0 \text{ because } v \text{ is active}$$

$$\Rightarrow \exists w \in R \text{ with } \mathrm{ex}_f(w) < 0$$

$$\Rightarrow w = s \Rightarrow s \in R$$

- Part 2 of proof:



v-w-path in G_f

$$\psi(v_0) \leq \psi(v_1) + 1 \leq \psi(v_2) + 1 + 1 \cdots \leq \underbrace{\psi(w)}_{v_k=w} + k \leq \psi(w) + n - 1$$

k : length $\leq n-1$ because of n vertices in G.

- Part 3 of proof - contradiction: Assume that t is reachable from s

$$\Rightarrow \exists s\text{-}t\text{-path in } G_f \stackrel{b}{\Rightarrow} \underbrace{\psi(s)}_{n} \leq \underbrace{\psi(t)}_{0} + n - 1 \Rightarrow n \leq n-1 \Rightarrow \text{contradiction}$$

□

Theorem 31. *When PR algorithm terminates, f is a maximal s-t-flow.*

Proof. No active vertices at termination $\Rightarrow$ preflow is flow and from Theorem 30 (c) “t is not reachable from s in G_f” is an invariant of the algorithm. Hence at termination the optimality criterion is satisfied. □

Theorem 32. *(number of relabel operations)*

- $\forall\, v \in V(G) : \psi(v)$ *is increased in every relabel operation by at least one (strong monotonicity, no decrement)*
- $\psi(v) \leq 2n - 1$ *is an invariant* $\forall v \in V(G)$
- *No vertex exists which is relabelled more than* $2n - 1$ *times. Hence the maximum number of relabel operations is* $2n^2 - n$

Proof.
- Active v has edges to $w_1, w_2, \ldots, w_k$.

$$\psi(v) < \psi(w_i) + 1 \qquad \Rightarrow \text{edge not permitted}$$

Relabel:

$$\overline{\psi(v)} = \min_{i:w_i\in\delta^+(v)} \psi(w_i) + 1 = \psi(w_l) + 1 > \psi(v)$$

- Initialization: $\psi(v) = 0 \, \forall \, v \neq s$, $\psi(s) = n$. Initialization okay.
  Inequation can possibly be invalidated by relabelling?

$$\text{Relabel}(v) \Rightarrow v \text{ active} \overset{\text{theorem 30 (1)}}{\Rightarrow} s \text{ is reachable in } G_f \text{ from } v$$
$$\overset{\text{theorem 30 (2)}}{\Rightarrow} \psi(v) \leq \psi(s) + n - 1 = n + n - 1 = 2n - 1$$

- Follows trivially from the previous 2 points

□

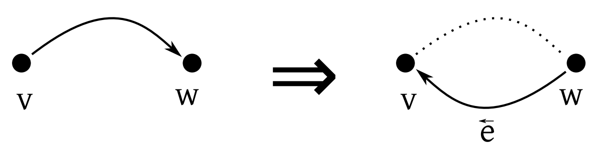
Definition. Let $e = (x, y)$. push(e) is called *saturating* push operation if

$$y = \min \{ \text{ex}_f(x), u_f(e) \} = \underbrace{u_f(e)}_{\neq \text{ex}_f(x)}$$

otherwise push(e) is a non-saturating operation.

Theorem 33. *The number of saturating push operations is* $2nm$.

Proof. Consider $e = (v, w)$. How many times will a saturating push operation push(e) be executed?



Edge must be be reintroduced in G_f to be pushed again.

$$\psi(v) = \psi(w) + 1$$

Before the next push(e), e must be added to G_f. How? This is only possible through push$(\overleftarrow{e})$. It holds:

$$\psi'(w) = \psi'(v) + 1 \quad \Rightarrow \quad \psi'(w) \geq \psi(v) + 1$$

So $\phi(w)$ has been incremented by at least 2. If the next push(e) happens, it furthermore holds that

$$\psi''(v) = \psi''(w) + 1 \geq \psi(w) + 2 + 1$$

In conclusion: Between every two continuous saturating push operations push(e), $\phi(v)$ was incremented by at least 2. From theorem 32 (b) we conclude that we have at most $n - 1$ jumps (by 2) of $\phi(v)$. This also means that we have at most n saturating push operations.

So for all edges of $E \cup \overleftarrow{E}$ we have at most $2mn$ saturating push operations. □

Theorem 34. Number of non-saturating push operations. *The number of non-saturating push operations is $\mathcal{O}(n^2 m)$.*

Proof. Let A be the set of all active vertices.

$$D := \sum_{v \in A} \psi(v) \qquad \text{potential function}$$

Start with $D = 0$. At termination $A = \varnothing, D = 0$. How does D change with execution of various operations?

- Non-saturating operation push(v, w):

  Transitions:

  $$\begin{array}{rcl c}
  \text{active} & \to & \text{active} & \text{D is less or equal} \\
  \text{non-active} & \to & \text{active} & \triangle D = -\psi(v) + \psi(w) = -(\psi(w)+1) + \psi(w) = -1 \\
  & & & \text{D is less}
  \end{array}$$

- Saturating operation push(v, w): Transitions:

  $$\begin{array}{rrcl c}
  w: & \text{active} & \to & \text{active} & \text{D stays the same} \\
  w: & \text{non-active} & \to & \text{active} & \text{D stays the same or increases} \\
  v: & \text{active} & \to & \text{active} & \text{D stays the same or increases}
  \end{array}$$

Relabel D increases, because $\psi(v)$ increases.

So in conclusion only non-saturating push is responsible for decreasing D.

□

By how much does D increment during the algorithm's run?

$$2nm \text{ saturating pushes } + 2n^2 - n \text{ relabels}$$

It increases with $\mathcal{O}((nm + n^2) \cdot n) = \mathcal{O}(n^2 m)$.

Theorem 35. Better analysis for number of non-saturating push operations. Cheriyan and Mehlhorn 1999. *If the algorithm always select an active vertex with maximum $\psi(v)$, then the push-and-relabel algorithm only requires $8n^2\sqrt{m}$ non-saturating push operations.*

Theorem 36. *The push-and-relabel algorithm solves the maximum-flow problem correctly and can be implemented with $\mathcal{O}(n^2\sqrt{m})$ runtime. (with selection of active vertices as in Theorem 35)*

Proof. Correctness is given in theorem 31 and theorem 29. What about the runtime?

We use a doubly-linked list L_i where

$$L_i := \{v \in V(G); v \text{ is active}; \psi(v) = i \quad 0 \leq i \leq 2n-1\}$$

At the begining we start with L_0 := all active vertices. Let $i = 0$ (at the beginning) and scan L_i for some selected $v \in L_i$. As an invariant i is the greatest index j with $L_j \neq 0$.

- i must be updated in constant time
- lists as well

$$k = \psi(v)$$

Relabel:

$$\psi(v) = \min_{w \in \delta^+(v)} \{\psi(w) + 1\} = \hat{k}$$

$$\mathcal{O}(\left|\delta^+(v)\right|)$$

If $k > i$, then $i = k$.

$$L_{\hat{k}} = L_k \cup \{v\}$$

$$L_k = L_k \setminus \{v\}$$

push$(e) : v \to w$

- v = active, switch to
  - active
  - inactive: $L_{\psi(v)} = L_{\psi(v)} \setminus \{v\}$
- w = active, switch to
  - active: $i = \psi(v) \wedge L_{\psi(v)} = \varnothing, i \leftarrow i', i'$ next $L_{i'} \neq \varnothing$
- w: inactive, switch to
  - active: $L_{\hat{\psi}(w)} = \hat{L}_{\psi(w)} \cup \{w\}, \hat{\psi}(w) > i$ then $i = \hat{\psi}(w)$

Data structure $A_v \; \forall \, v \in V(G)$ is doubly linked.

$$A_v = \big\{w \in \delta^+(v) : (v, w) \text{ permitted}\big\}$$

Saturating push: edge is removed from A_v:

$$A_v := A_v \setminus \{w\}$$

Unsaturating push: A_v stays unmodified

$$A_v := A_v \setminus \{w\}$$

$$\text{Relabel}(v) := \psi(w) = \min_{w \in \delta^+(v)} \{\psi(w) + 1\} = \psi(w_e) + 1 \Rightarrow A_v = A_v \cup \{w_e\}$$

Per edge $2n - 1$ times.

$$\begin{aligned} 2n - 1 \mathcal{O}(\bigcup_{v \in V(G)} \left|\delta^+(v)\right|) \\ = \mathcal{O}(nm) \end{aligned}$$

□

This lecture took place on 10th of November 2014.

13.9 Minimum-capacity cut problem

Given. Instance (G, u, s, t), $u : t \to R_t$

Find. A s-t-cut in G with minimum capacity

Simple solution. Determine maximum flow f from s to t and all reachable vertices v from s in G_f. Let C be that vertex. From the maximum-flow min-cut theorem it follows that

$$\text{value}(f) = u(\delta(L)) \quad \to \quad \delta(c) \text{ minimal } s\text{-}t\text{-cut}$$

If you want to compute one minimum cut per pair (s, t), solve $\binom{1}{2}$ max-flow problems and determine the corresponding cuts (as above)

$$\binom{n}{2} \mathcal{O}(n^2 \sqrt{n})$$

$(n-1)$ flow computations actually suffice; see Gomor-Hu-Baum (1962).

For undirected graphs. Let G be an undirected graph and $u : E(G) \to \mathbb{R}_+ \; \forall \, s, t \in V(G)$. Let $\lambda_{s,t}$ be the *local node connectivity* defined as minimum capacity of a splitting s-t-cut. The *node connectivity of a graph* is defined as $\min_{s,t \in V(G), s \neq t} \lambda_{s,t}$ where $\lambda_{s,t}$ is computed with $u(e) = 1 \; \forall \, e \in E(G)$.

$$\begin{aligned} \lambda_{1,2} = \lambda_{2,3} = \lambda_{1,3} = 2 \\ \min \{\lambda_{12}, \lambda_{23}, \lambda_{13}\} = 2 \\ \Rightarrow \text{ node connectivity of } G = 2 \end{aligned}$$

Alternative definition: $G = (V, E)$ is called t-node-connected if the graph G stays connected after removal of $t - 1$ vertices. The node connectivity number $K(G)$ of a graph G is defined as $K(G) = \max_{t \in \{1,2,\ldots,|V(G)|-2\}} \{t : G \quad t \text{ node connectivity}\}$.

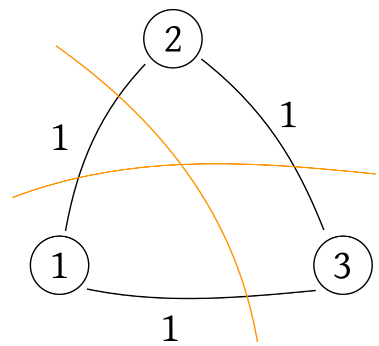


Figure 19: Vertex connection

Theorem 37. *For every triple of vertices $i, j, k \in V(G)$ (G is an undirected graph) it holds that*

$$\lambda_{i,k} \geq \min\{\lambda_{i,j}, \lambda_{j,k}\}$$

Proof. Let $\delta(C)$ be a minimal i-k-cut with $\lambda_{i,k} = u(\delta(C))$. If $j \in C$ then $\lambda_{j,k} \leq u(\delta(C)) = \lambda_{i,k}$. If $j \in V(G) \setminus C$ then $\lambda_{i,j} \leq u(\delta(C)) = \lambda_{i,k}$. So $\lambda_{i,k} \geq \lambda_{j,k}$ or $\lambda_{i,k} \geq \lambda_{i,j}$. $\Rightarrow \lambda_{i,k} \geq \min\{\lambda_{j,k}, \lambda_{j,i}\}$. □

Remark. The condition of theorem 37 is required for $(\lambda_{i,j})_{i,j \in 1,2,\dots,n}$ node connectivity numbers of a graph with vertex set $\{1, 2, \dots, n\}$. But this condition is not sufficient; hence $\exists$ numbers $\lambda_{i,j}$ for $i, j \in \{1, 2, \dots, n\}$ that satisfy these conditions, but cannot be retrieved as node connectivity number of a graph.

If $\lambda_{i,j} = \lambda_{j,i} \forall i, j \in \{1, 2, \dots, n\}$ and the conditions of theorem 37 hold, then it holds that a graph G with $V(G) = \{1, 2, \dots, n\}$ and $\exists u : E(G) \to \mathbb{R}_+$, such that $\lambda_{i,j}$ for $i, j \in \{1, 2, \dots, n\}$ that are local node connectivity numbers of (G, u).

Definition. Let G be an undirected graph and $u : E(G) \to \mathbb{R}_+$, E in tree T is called Gomory-Hu-tree of (G, u) iff

$$V(T) = V(G) \wedge \lambda_{s,t} = \min_{e \in E(P_{s,t})} u(\lambda_G(c_e)) \; \forall \, s, t \in V(G), s \neq t$$

$$K_{3,3} \qquad n \equiv 1$$

$$\lambda_{s_i,t_j} = \lambda_{s_1,t_1} = 3 \forall i \in \{1,2,3\}, j \in \{1,2,3\}$$

$$\lambda_{s_1,s_2} = \lambda_{s_1,s_3} = \lambda_{s_2,s_3} = \lambda_{t_1,t_2} = \lambda_{t_2,t_3} = \lambda_{t_1,t_3} = 3$$

$$\lambda_{s_2,t_2} = \min\left\{\underbrace{u(\delta(c_{e_1}))}_{=3}, \underbrace{u(\delta(c(e_2)))}_{=3}\right\} = 3$$

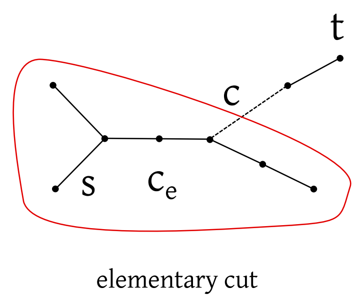


Figure 20: Elementary cut

13.10 Gomory-Hu algorithm

Basic idea: Select vertex pair $s, t \in V(G)$ and determine minimal s-t-cut $\delta(A)$ where $B := V(G) \setminus A$. Contract A (or B) to one vertex. Then select s', $t' \in B$ (or A).

Build a minimal s'-t'-cut $\delta(A')$ in contracting graph G' with $V(G') \setminus A'$. Observe that a minimal cut in the contracting graph corresponds to a minimal cut in the original graph. Repeat this step as long as not-separated vertices exist.

Theorem 38. *Let G be an undirected graph and $u : E(G) \to \mathbb{R}_+$. Let $s, t \in V(G)$ and $\delta(A)$ a minimal s-t-cut in (G', u'). (G', u') results from (G, u) by contraction of A by a single vertex K. Let $s', t' \in V(G) \setminus A$. Then it holds that*

$$\forall \min s\text{'-}t\text{'-}cuts : \delta(K \cup \{A\}) \text{ } is \text{ } \delta(K \cup A) \text{ } a \text{ } minimal \text{ } s\text{'-}t\text{'-}cut \text{ } in \text{ } (G, u)$$

Remark. It's a minimal cut. It does not preserve capacity.

So K is the set of non-contracting vertices that are on the same halfplane like the s'-t'-cut like A.

Proof. We show: there exists a minimal s'-t'-cut $\delta(A')$ in (G, u') with $A \subseteq A'$. Let $\delta(c)$ be a minimal s'-t'-cut in (G, u). Without loss of generality let $s \in C$. If $A \subseteq C$, we are done. Otherwise build a second minimal s'-t'-path in G, which contains A. It holds that

$$u(\delta(A)) + u(\delta(c)) \geq u(\delta(A \cap C)) + u(\delta(A \cup C))$$

Because $\delta(A \cap C)$ is a s-t-cut G it holds that

$$u(\delta(A \cap c)) \geq \lambda_{s,t} = u(\delta(A))$$

$$u(\delta(A \cap C) \leq u(\delta(c)) = \lambda_{s',t'} \Rightarrow u(\delta(A \cap C)) = \lambda_{s',t'}$$

Algorithm 12 Gomory Hu algorithm

Given. undirected graph G, $u : E(G) \to \mathbb{R}_+$

Find. A Gomory-Hu tree T for (G, u)

1: $V(T) = \{V(G)\}$, $E(T) = \varnothing$ (a vertex that corresponds to all vertices in G)

2: Choose $X \in V(T)$ with $|X| \geq 2$. If $\nexists X$, then goto step 6.

3: Choose $s, t \in \mathcal{X}$ $(s \neq t)$, H as connected component C of $T - X$

4: Select $S_c := \bigcup_{Y \in V(C)} Y$. (G', u') results from (G, u) by contraction of s_c to a single vertex V_c for every connected component C of $T - x$. So $V(G') = X \cup \{V_c : c \text{ connected component of } T - X\}$.

5: Determine minimal s-t-cut $\delta(A')$ in (G', u'). Let $B' = V(G') \setminus A'$. Set

$$A = \bigcup_{V_c \in A' \setminus X} S_c \cup (A' \cap X)$$

and

$$B = \bigcup_{V_c \in B' \setminus X} S_c \cup (B' \cap X)$$

6: Set $V(T) := (V(T) \setminus X) \cup (A \cap X) \cup (B \cap X)$. For every edge $e = \{X, Y\} \in E(T)$ incident with X do,

7: if $V \subseteq A$, set $e' := \{A \cap X, Y\}$

8: else $e' := \{B \cap X, Y\}$.

9: Set $E(T) := (E(T) \setminus e) \cup \{e'\}$, $w(e') := w(e)$. Set $E(T) := E(T) \cup \{A \cap X, B \cap X\}$, $w(\{A \cap X, B \cap W\}) = u'(\delta_G, (A'))$. Goto step 2.

10: Replace all $\{X\} \in V(T)$ by X and all $\{\{X\}, \{Y\}\}$ by $\{X, Y\}$.

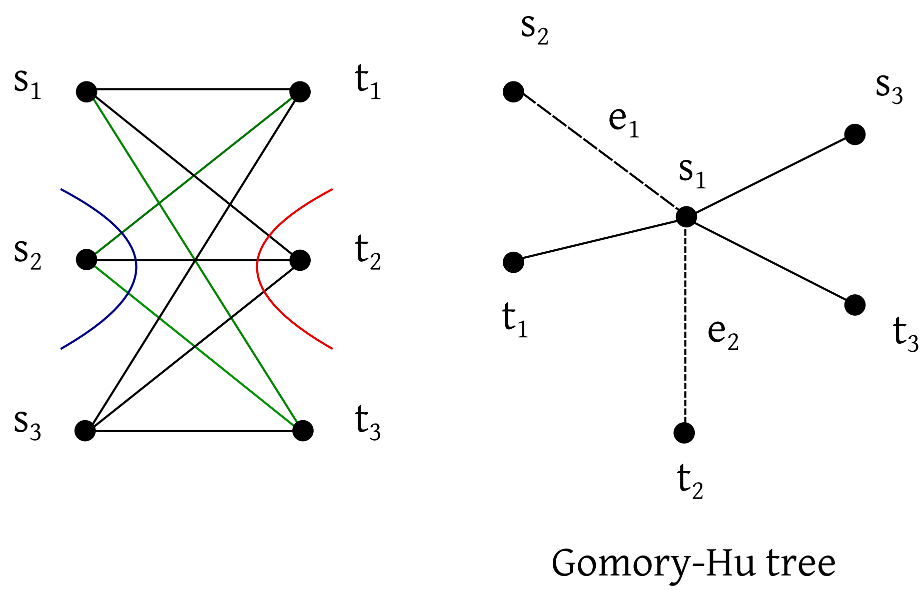


Figure 21: Gomory Hu algorithm

$$A \cap C \text{ is minimal s'-t' cut with } A \cap C \subseteq A$$

□

Definition. $f : 2^M \to \mathbb{R}$ where M is the set and 2^M is the power set of M. f is called submodular if $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)\ \forall\, A, B \in 2^M$.

Theorem 39. *After every iteration of step 4, the following conditions hold:*

- $A \dot{\cup} B = V(G)$
- *$E(A, B)$ is a minimal s-t-cut in (G, u)*

$$A, B \subseteq V(G) \qquad E(A, B) := \{e \in E(G) : e = (x, y) \quad x \in A, y \in B\}$$

A proof for Theorem 39 is not provided.

Theorem 40. *Invariant of the algorithm:*

$$w(e) = u(\delta_G(\bigcup_{z \in C_e} Z))\ \forall\, e \in E(T)$$

where c_e and $V(T) \setminus c_e$ are the two connected components of $T - e$. Furthermore it holds that

$$\forall e = \{P, Q\} \in E(T) \quad \exists p \in P \quad \exists q \in Q \text{ with } \lambda_{p,q} = w(e)$$

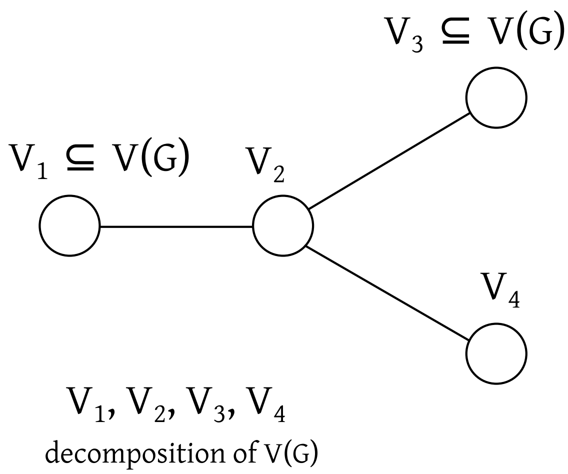


Figure 22: Gomory Hu algorithm idea

A proof for Theorem 40 is not provided.

Theorem 41. *The Gomory-Hu algorithm works correctly. Every undirected graph contains a Gomory-Hu tree which can be computed in runtime $\mathcal{O}(n^3\sqrt{m})$.*

A proof for Theorem 41 is not provided.

13.11 Minimal capacity of a cut in an undirected graph / MA-order

Given. G as undirected graph, $u : E(G) \to \mathbb{R}_+$

Find. Determine a $A^* \subsetneqq V(G), A^* \neq \varnothing$ such that

$$u(\delta(A^*)) = \min_{A \subsetneqq V(G), A \neq \varnothing} u(\delta(A))$$

Solution 1. Let $s \in V(G)$ be random. Determine $\lambda_{s,t} \forall t \in V(G) \setminus \{s\}$. The cut minimizing $\lambda_{s,t}$ is the optimal cut. The solution with Gomory-Hu algorithm: The optimal cut is computed using the edge $e^* \in$ GH-tree T with $w(e^*) = \min_{e \in E(T)} w(e)$.

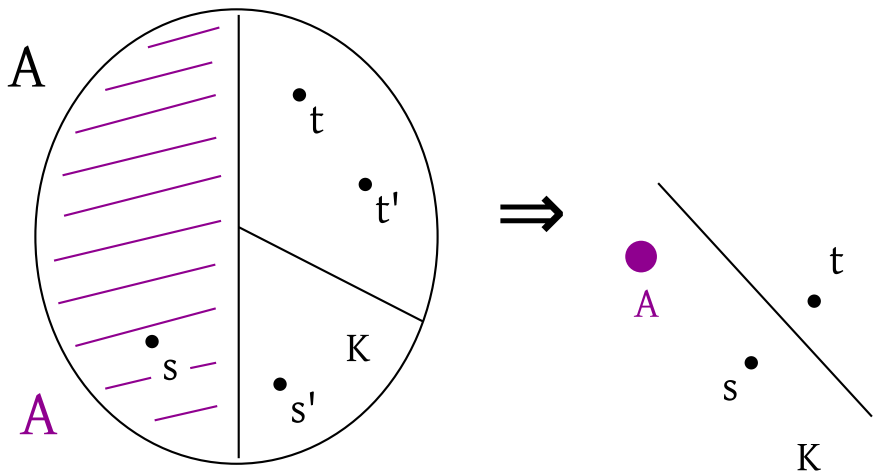


Figure 23: Lemma 4.16

Solution 2. (Frank 1994, Stoer & Wagner 1997.) Let G be an undirected graph with $u : E(G) \to \mathbb{R}_+$. An order $v_1, v_2, \ldots, v_u$ of vertices is called *MA-order* (maximum adjacency order) if $\forall\, i \in \{1, 2, \ldots, n\}$ it holds that:

$$\sum_{e \in E(\{v_1\}, \{v_1, v_2, \ldots, v_{i-1}\})} u(e) = \max_{j \in \{i, i+1, \ldots, n\}} \sum_{e \in E(\{v_1\}, \{v_1, \ldots, v_{i-1}\})} u(e)$$

This order is not distinct.

Theorem 42. *In an undirected graph G with $u : E(G) \to \mathbb{R}_+$ we can compute a MA-order in $\mathcal{O}(m + n \log n)$ time.*

Proof. Algorithmically. Let $\alpha(v) := 0\, \forall\, v \in V(G)$. Apply the following solution for $i = 1$ to n. Select some $v_i \in \operatorname{argmax} \{\alpha(v) : v \in V(G) | \{v_1, \ldots, v_{i-1}\}\}$. “tie branching arbitrarily”.

$$\alpha(v) := \alpha(v) + \sum_{e \in E(\{v_i\}, \{v\})} u(e) \qquad \forall v \in V(G) \setminus \{v_1, v_2, \ldots, v_i\}$$

$\square$

Proving correctness is trivial this is an invariant:

$$\alpha(v) = \sum_{c \in E(\{v\}, \{v_1, \ldots, v_i\})} \qquad \forall v \in V(G) \setminus \{v_1, \ldots, v_i\}$$

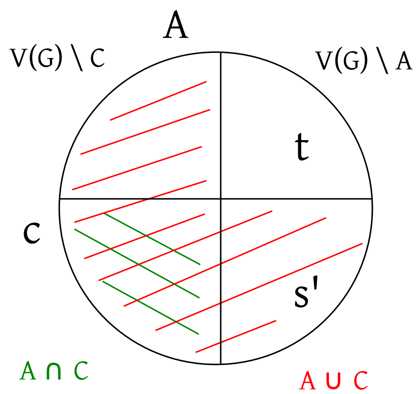


Figure 24: Lemma 4.16 proof

Time complexity: Fibonacci heaps with key $-\alpha(v)$.

delete min	$\mathcal{O}(\log n)$	(amortized)
insert	$\mathcal{O}(1)$	
decrease-key	$\mathcal{O}(1)$	(armortized)

$$\Rightarrow \mathcal{O}(m + n \log n)$$

Theorem 43. *Let G be an undirected graph with $u : E(G) \to \mathbb{R}_+$ and MA-order $u_1, \ldots, u_n$. Then it holds that*

$$\lambda_{v_{n-1},v_n} = \sum_{e \in E(\{v_n\},\{v_1,\ldots,v_{n-1}\})}$$

Proof. Proving direction "≤" is trivial. For "≥" via induction over $|V(G)| + |E(G)|$.

induction base $|V(G)| \leq 2$ is trivial.

induction step $|V(G)| \geq 3$. Without loss of generality $(V_{n-1}, V_n) \notin E(G)$, because if $e \in E(G)$ with $e = (v_{n-1}, v_n)$. Then e is optimal cut between v_{n-1} and v_n. Furthermore $e \in E(\{v_n\}, \{v_1, \ldots, v_{n-1}\})$. Hence removal of edge e reduces both sides of theorem 43 by $u(e)$. Followingly the induction hypothesis is applicable.

Let

$$R := \sum_{e \in E(\{v_n\},\{v_1,\ldots,v_{n-2}\})} u(e) \overset{(u_n,u_{n-1}) \notin E}{=} \sum_{e \in E(\{v_n\},\{v_1,\ldots,v_{n-1}\})} u(e)$$

$$(v_1, \ldots, v_n) \text{ is MA-order } \in G \Rightarrow (v_1, \ldots, v_{n-1}) \text{ is MA-order } \in G \setminus v_n$$

From the induction hypothesis for $G - \{v_n\}$ it follows that

$$\lambda_{n-2,n-1} = \sum_{e \in E(\{v_{n-1}\}, \{v_1, \ldots, v_{n-2}\})} u(e) \geq \sum_{e \in E(\{v_n\}, \{v_1, \ldots, v_{n-2}\})} u(e) = R$$

$$\Rightarrow \lambda^G_{v_{n-2}, v_{n-1}} \geq R \tag{3}$$

$$v_1, v_2, \ldots, v_{n-2}, v_n \text{ is a MA-order in } G - \{v_{n-1}\}$$

From induction hypothesis it follows:

$$\lambda^G_{v_{n-2}, v_n} \geq \lambda^{G - \{v_{n-1}\}}_{v_{n-2}, v_n} = \sum_{e \in E(\{v_1\}, \{v_1, \ldots, v_{n-1}\})} u(e) = R \tag{4}$$

From the last two statements regarding R we conclude,

$$\lambda_{v_{n-1}, v_n} := \lambda^G_{v_{n-1}, v_n} \geq \min \left\{ \lambda^G_{v_{n-1}, v_{n-2}}, \lambda^G_{v_{n-2}, v_n} \right\} \geq R$$

(triangle inequation)

□

Theorem 44. *A cut of minimal capacity in an undirected graph G with $u : E(G) \to \mathbb{R}_+$ can be computed with $\mathcal{O}(nm + n^2 \log m)$ runtime.*

Proof. Constructive proof. Several edges of capacity u with same source and destination can be combined to a single edge. So we can can assume wlog no multi-edges.

Denote λ_G as the minimal capacity of a cut in G. Let $G_0 := G$. Apply n steps. In the i-th step $(i = 1, 2, \ldots, n)$ select vertex $x, y \in V(G_{i-1})$ with

$$\lambda^{G_{i-1}}_{x,y} = \sum_{e \in \delta_{G_{i-1}}(x)} u(e)$$

It's existence is given by theorem 43 and x last vertex and y pre-last vertex in concern of a MA-order in G_{i-1}.

Let $y_i = \lambda^{G_{i-1}}_{x,y}$ and $z_i = x \; \forall \, i = 1, 2, \ldots, n$. Build G_i from G_{i-1} by contraction of edge (x, y).

Observation.

$$\lambda(G) = \lambda(G_0) = \min \{\lambda(G_1), \gamma_1\} = \min \{\min \{\lambda(G_2, \gamma_2)\}, \gamma_1\}$$

$$= \min\{\lambda(G_2), \gamma_2, \gamma_1\} = \min\left\{\lambda(\underbrace{G_{n-1}}_{=+\infty}), \gamma_1, \gamma_2, \ldots, \gamma_{n-1}\right\}$$

$$= \min_{i=1,\ldots,n-1} \gamma_i \qquad \text{value of optimal cut}$$

Let $\lambda(G) = \gamma_k = \lambda^{k-1}_{Z_k,\gamma}$ (z_k is a vertex connected by an edge with y and this edge is part of the cut; z_k is alone, but could be a multi-vertex).

The optimal cut is determine by the subset of the vertex set, that can be contracted in z_k.

Complexity. n iterations. Per iteration we compute the MA-order in $\mathcal{O}(m + n \log n)$ and contraction and defining new capacities takes $\mathcal{O}(n+m)$. This gives us

$$\Rightarrow \mathcal{O}(mn + n^2 \log n)$$

Example.

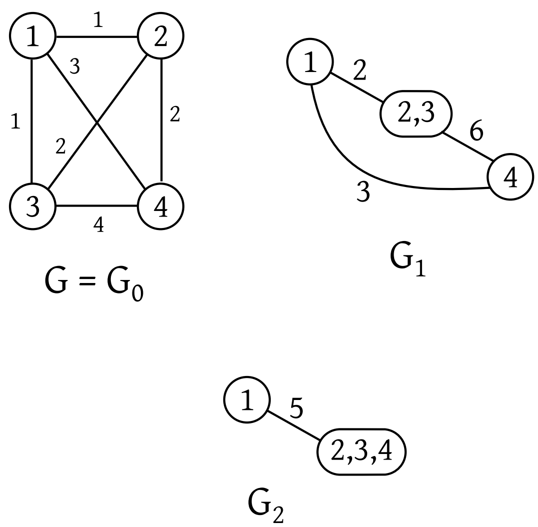


Figure 25: Example for minimal-capacity cut

$$\text{MA in } G_0 : \underbrace{1}_{\text{arbitrary}}, \underbrace{4}_{\text{most edges to 1}}, \underbrace{3}_{y}, \underbrace{2}_{x}$$

$$z_1 = 2 \qquad y_1 = 1 + 2 + 2 = 5$$

$$\text{MA in } G_1 : 1, \underbrace{4}_{y}, \underbrace{(2,3)}_{x}$$

$$z_2 = (2,3) \qquad y_2 = 2 + 6 = 8$$

$$\text{MA in } G_2 : \underbrace{1}_{y}, \underbrace{(4,2,3)}_{x}$$

$$z_3 = (4,2,3) \qquad y_3 = 5$$

$$\gamma(G) = \min\{\gamma_1, \gamma_2, \gamma_3\} = \gamma_1 (\text{or } \gamma_3)$$

In G_3 we would have only 1 vertex remaining. $\square$

14 Flows with minimum costs

Definition. Let G be a digraph.

$$u : E(G) \to \mathbb{R}_+ \qquad c : E(G) \to \mathbb{R}$$

Let $b : V(G) \to \mathbb{R}$ with $\sum_{v \in V(G)} b(v) = 0$. A *b-flow of* G is a mapping $f : E(G) \to \mathbb{R}_+$ such that

$$f(e) \le u(e) \qquad \forall e \in E(G) \quad \text{“capacity restrictions”}$$

and

$$\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = b(v) \qquad \forall v \in V(G) \qquad \text{flow preservation conditions}$$

hold.

A v with $b(v) > 0$ is called “offering vertex”. A v with $b(v) < 0$ is called “demanding vertex”.

14.1 Minimum cost flow problem (MCFP/MKFP)

Given. Digraph G

$$u : E(G) \to \mathbb{R}_+ \qquad c : E(G) \to \mathbb{R} \qquad b : V(G) \to \mathbb{R}$$

with $\sum_{v \in V(G)} b(v) = 0$

Find. Determine a b-flow with minimum costs,
ie. the b-flow which minimizes $\sum_{e \in E(G)} c(e) f(e)$

Remark. A b-flow is determined by the maximum-flow-problem.

$$u(s,v) = \max\{0, b(V)\}\ \forall\, v \in V(G)$$
$$u(v,t) = \max\{0, -b(v)\}\ \forall\, v \in V(G)$$

A maximal s-t-flow f in G' has value $\sum_{v\in V(G)} b(v)$ iff a b-flow exists in G. If b-flow exists, then f restricted by $E(G)$ by b-flow.

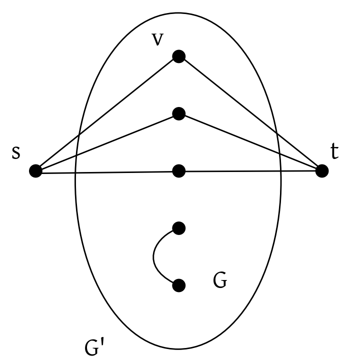


Figure 26: b-flow

Observation. There exists a max s-t-flow with value

$$\sum_{v\in V(G), b(v)>0} b(v) \text{ in G'}$$

$$\Leftrightarrow b - \text{flow in } G$$

$\Rightarrow$ Let f' be a max-s-t-flow with

$$\text{value}(f') = \sum_{v\in V(G), b(v)>0} b(v)$$

Let $f = f'|_{E(G)}$ hence $f : E(G) \to \mathbb{R}$ with $f(e) = f'(e)\ \forall\, e \in E(G)$. Show that f is a b-flow.

$$\forall\, v \in V(G) : \sum_{l\in\delta_G^+(v)} f(e) - \sum_{l\in\delta_G^-(v)} f(e) = \forall_{\,e\in\delta_G^+(v)} f'(e) - f'(v,t) - \sum_{e\in\delta_G^-(v)} f'(e) + f'(s,v)$$

$$= f'(s,v) - f'(v,t) = \max\{b(v), 0\} - \max\{0, -b(v)\} = b(v) \Rightarrow f \text{ is } b\text{-flow}$$

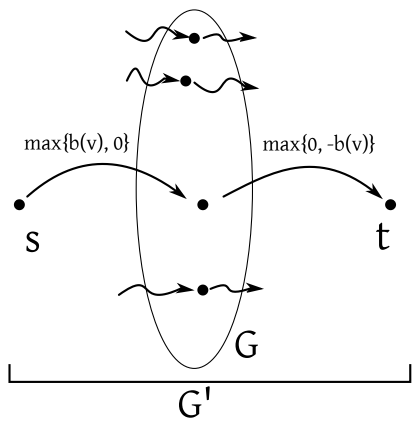


Figure 27: MFCP observation

$\Leftarrow$ Let f be a bf-low in G. Build $f' : E(G') \to \mathbb{R}$ with

$$f'(e) = \begin{cases} f(e) & e \in z(G) \\ \max\{0, b(v)\} & e = (s, v) \\ \max\{0, -b(v)\} & e = (v, t) \end{cases}$$

Exercise: Show that f' is an s-t-flow with value

$$\sum_{v \in V(G)} \max\{0, b(v)\} = \sum_{v \in V(G), b(v) > 0} b(v)$$

14.2 The transportation problem

Given. Digraph G with $V(G) = A \cup B$ and $E(G) \subseteq A \times B$. $b(v) \geq 0 \; \forall v \in A, b(v) < 0 \; \forall v \in B, \sum_{v \in V(G)} b(v) = 0$. Capacities $u(e) = \infty \; \forall e \in E(G)$ and $c : E(G) \to \mathbb{R}$.

Find. b-flow in G with minimum costs

Remark. Without loss of generality $c(e) \geq 0 \; \forall e \in E(G)$. Otherwise α is large enough $(c(e) + \alpha \geq 0 \; \forall e \in E(G))$.

$$c'(e) = c(e) + \alpha \geq 0 \; \forall e \in E(G)$$

Let f be a b-flow in G. Compare $c'(f)$ and $c(f)$.

$$c'(f) = \sum_{e \in E(G)} c'(e) f(e) = \sum_{e \in E(G)} (c(e) + \alpha) f(e) = c(f) + \alpha \sum_{e \in E(G)} f(e)$$

Consider

$$\sum_{(u,v)\in E(G)} f(u,v) = \sum_{u\in V(G)} \sum_{e\in\delta^+_G(u)} f(e) = \sum_{u\in A} \left(\sum_{e\in\delta^+_G(u)} f(e) - \sum_{e\in\delta^-_G(u)} f(e) \right) = \sum_{u\in A} b(u) =: \bar{b}$$

$$c'(f) = c(f) + a\bar{b}$$

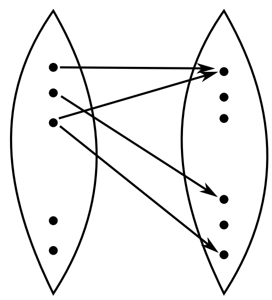


Figure 28: Transportation problem

14.3 An optimality criterion

Let f be a b-flow. G_f is defined analogously like in the max-flow problem. Costs in G_f:

$$c_f : E(G_f) \to \mathbb{R}$$

$$c_f(e) = \begin{cases} c(e) & e \in E(G) \\ -c(e) & \overleftarrow{e} \in E(G) \end{cases}$$

Definition. Let G be a diagraph with capacity $u : E(G) \to \mathbb{R}_+$ and let f be a b-flow in G. A f*-augmenting cycle* is a cycle in G_f.

Theorem 45. *Let G be a digraph with capacity $u : E(G) \to \mathbb{R}_+$. Let f and f' be b-flows in G. Then $g : \overleftrightarrow{E}(G) \to \mathbb{R}$ with $g(e) = \max\{0, f'(e) - f(e)\}$ and $g(\overleftarrow{e}) = \max\{0, f(e) - f'(e)\}$ $\forall\, e \in E(G)$ is a* circulation *in $\overleftrightarrow{G} := (V(G), \overleftrightarrow{E}(G))$. Furthermore it holds that $g(e) = 0\ \forall\, e \in \overleftrightarrow{E}(G) \setminus E(G_f)$ and $c(g) = c(f') - c(f)$.*

Proof.

$$\forall\, v \in V(\overleftrightarrow{G}) = V(G)$$

$$\sum_{e\in\delta^+_{\overleftrightarrow{G}}(v)} g(e) - \sum_{e\in\delta^-_{\overleftrightarrow{G}}(v)} g(e)$$

$$\begin{aligned}
= &\sum_{e\in\delta^+_G(v)} \max\{0, f'(e)-f(e)\} - \\
&\sum_{e\in\delta^-_G(v)} -\max\{0, f(e)-f'(e)\} - \\
&\sum_{e\in\delta^-_G(v)} \max\{0, f'(e)-f(e)\} + \\
&\sum_{e\in\delta^+_G(v)} -\max\{0, f(e)-f'(e)\}
\end{aligned}$$

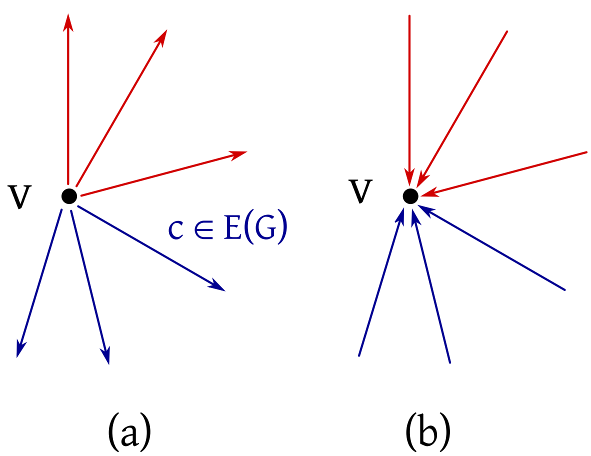


Figure 29: Proof for proposition 5.1

(a) is $\delta^+_{\overleftrightarrow{G}}(v)$. (b) is $\delta^-_{\overleftrightarrow{G}}(v)$.

$$\sum_{e\in\delta^+_G(v)} \underbrace{(\max\{0, f'-f\} - \max\{0, f-f'\})}_{=f'-f} - \sum_{e\in\delta^-_G(v)} (f'(e)-f(e))$$

$$= \sum_{e\in\delta^+_G(v)} (f'(e)-f(e)) - \sum_{l\in\delta^-_G(v)} (f'(e)-f(e)) = 0$$

2 cases for $e \in \overleftrightarrow{G} \setminus E(G_f)$:

forward edge	$e \in E(G)$	$f(e) = u(e)$	$g(e) = \max\{0, f' - f\} = \max\{0, f' - u\} = 0$
backwards edge	$\overleftarrow{e} \in E(G)$	$f(\overleftarrow{e}) = 0$	$g(e) - \max\{0, f - f'\} = \max\{0, 0 - f'\} = 0$

$\square$

$$c(g) = \sum_{e \in \overleftrightarrow{E}(G)} g(e)c(e)$$

$$= \sum_{e \in E(G)} c(e) \max\{0, f'(e) - f(e)\} + \sum_{\overleftarrow{e} \in E(G)} -c(\overleftarrow{e}) \max\{0, f(\overleftarrow{e}) - f'(\overleftarrow{e})\}$$

$$= \sum_{e \in E(G)} c(e) \max\{0, f' - f\} + \sum_{e \in E(G)} c(e)[-\max\{0, f - f'\}]$$

$$= \sum_{e \in E(G)} c(e) \underbrace{[\max\{0, f' - f\} - \max\{0, f - f'\}]}_{f'(e) - f(e)}$$

$$= \sum_{e \in E(G)} c(e)\,(f'(e) - f(e)) = c(f') - c(f)$$

Theorem 46. *For every circulation f in a digraph G there is a family $\mathcal{C}$ of at most $E(G)$ cycles in G and positive numbers $h(C)\ \forall\, c \in \mathcal{C}$ with*

$$f(e) = \sum_{c \in \mathcal{C}, e \in E(C)} h(e)$$

Proof. This follows directly from the flow decomposition theorem for s-t-flows with arbitrary weighted $s, t \in V(G), s \neq t$ because circulation f is also a s-t-flow with flow value 0.

$$f(e) = \sum_{P \in \mathcal{P}, e \in E(P)} h(P) + \sum_{c \in \mathcal{C}, e \in E} h(c)\ \forall\, e \in E(G), h(P) \geq 0\ \forall\, P \in \mathcal{P}, h(C) \geq 0\ \forall\, c \in \mathcal{C}$$

$$\text{value}(f) = \sum_{P \in \mathcal{P}} h(P) = 0 \Rightarrow h(P) = 0\ \forall\, P \in \mathcal{P}$$

$$\text{with } |\mathcal{P}| + |\mathcal{C}| = \mathcal{O}(|E(G)|)$$

$\square$

Theorem 47. *(Klein, 1967) Let (G, u, b, c) be an instance of MKFP. A b-flow g has minimum costs exactly iff there are no f-augmented cycles with negative costs in G_f.*

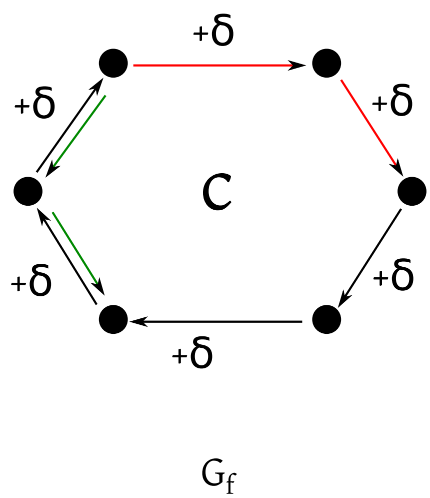


Figure 30: Proof of theorem 47

Proof. Direction of proof: $\Rightarrow$

Assume that an f-augmented cycle C (in G_f) exists with $\gamma < 0$ ($\gamma := \sum_{e \in E(C)} c_f(e)$). Let $\delta = \min_{e \in E(C)} u_f(e) > 0$.

Extension of G_f from forward edge $\rightarrow +\delta$ in G. Backward edge from backward edge $\rightarrow -\delta$ in G.

Augment flow and retrieve f' with

$$f \oplus c = f'(e) = \begin{cases} f(e) & e \in E(C), \overleftarrow{e} \notin E(C) \\ f(e) + \delta & e \in E(C) \\ f(e) - \delta & \overleftarrow{e} \in E(C) \end{cases}$$

f' is a b-flow.

$$c(f') = \sum_{e \in E(G)} c(e) f'(e) = \sum_{e \in E(G)} c(e) f(e) + \sum_{\overleftarrow{,} e \in E(C)} c(e)(f'(e) - f(e))$$

$$= c(f) + \sum_{e \in E(C)} c(e)\delta + \sum_{e: \overleftarrow{e} \in E(C)} -c(e)(-\delta) = c(f) + \delta \left(\sum_{e \in E(C), e \in E(G)} c(e) - 2c(e) \right)$$

$$e \in E(C), \overleftarrow{e} \notin E(C)$$

$$= c(f) + \delta \left(\sum_{c \in E(C)} c(e) + \sum_{\overleftarrow{e} \in E(C)} c(\overleftarrow{e}) \right)$$

$$c(f') = c(f) + \delta\gamma \Rightarrow c(f') \text{ is not optimal}$$

Direction of proof: $\Leftarrow$

Let f be a cost-non-minimal b-flow. Show that there exists some f-augmented cycle K in G_f with $c(k) < 0$.

Let f' be a b-flow with $c(f') < c(f)$.

Definition as in Theorem 45: $g : \overleftrightarrow{E} \to \mathbb{R}_+$ with $g(e) = \max\{0, f' - f\}$ and $g(\overleftarrow{e}) = \max\{0, f - f'\}\ \forall\, c \in E(G)$.

According to Theorem 45, g is a circulation. From Theorem 46 it follows that family $\mathcal{C}$ of cycles in $\overleftrightarrow{E}$ exists with

$$h : \mathcal{C} \to \mathbb{R}_+ \qquad G \mapsto h(C)$$

$$g(e) = \sum_{c \in \mathcal{C}, e \in E(C)} h(c)$$

Because $g(e) = 0\ \forall\, e \notin G_f$, for all $c \in \mathcal{C}$ it holds that $E(C) \subseteq E(G_f)$ (Theorem 45). Also it follows that $c(g) = c(f') - c(f) < 0$.

$$0 > c(g) = \sum_{e \in \overleftrightarrow{E}(G)} c(e)g(e) = \sum_{e \in \overleftrightarrow{E}(G)} c(e) \sum_{C \in \mathcal{C}, e \in E(C)} h(c)$$

$$= \sum_{C \in \mathcal{C}} h(c) \sum_{l \in \overleftrightarrow{E}(G) \cap E(G)} c(e) = \sum_{C \in \mathcal{C}} h(C)$$

$$\Rightarrow \exists C \in \mathcal{C} \text{ with } c(C) < 0$$

□

14.4 Minimum-mean cycle cancelling algorithm

This lecture took place on 18th of November 2014.

Theorem 48. *(Corollary.) A b-flow has minimum costs iff (G_f, C_f) has a (valid) potential function.*

Proof. b-flow f is optimal $\Leftrightarrow \nexists$ cycle K with $c_f(K) < 0$ in $(G_f, C_f) \Leftrightarrow \exists$ potential function $\pi : V(G_f) \to \mathbb{R}$ such that $c_f(n, f) + \pi(n) - \pi(v) \geq 0\ \forall\, (u, v) \in E(G_f)$ □

Algorithm 13 Minimum-mean cycle cancelling algorithm

Given. digraph G, $u : E(G) \to \mathbb{R}_+$, $c : E(G) \to \mathbb{R}$, $b : V(G) \to \mathbb{R}$ with $\sum_{v \in V(G)} b(v) = 0$

Find. A b-flow f^* with minimum costs $c(f^*) = \sum_{e \in E(G)} c(e) \cdot f^*(c)$ or "no b-flow exists in G"

Remark. $f'|_{E(G)}$ denotes f' restricted to the edges $E(G)$

```
Extend G by vertices s,t and edges (s,v), (v,t) ∀ v ∈ V(G) with u((s,v)) =
max{0, b(v)}. u(v,t) = max{0, −b(v)} ∀ v ∈ V(G). Let G' be an extended
network. Determine max-s-t-flow f' in G'.
if value(f') < Σ_{v∈V(G)} b(v) then
    "there does not exist a b-flow in G"
else
    let f = f'|_{E(G)}
end if
while ∃ cycle with negative weights in G_f do
    Determine cycle K with min c(K)/|E(K)| in G_f.
    Augment f along K : f := f ⊕ K
end while
return f
```

Proof. Second proof to prove corollary (using linear programming)

Let $(x_e)_{e \in E(G)}$ which corresponds to a b-flow. Costs:

$$- \sum_{e \in E(G)} c_e x_e \to \max$$

$$\text{Linear programming: } \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = b(v) \; \forall \, v \in V(G)$$

$$x_e \leq u_e \; \forall \, e \in E(G)$$

$$x_e \geq 0 \; \forall \, e \in E(G)$$

Dual problem:

$$(DLP) \min \sum_{v \in V(G)} b(v) y_v + \sum_{e \in E(G)} u_e z_e$$

$$y_u - y_v + z_e \geq -c_e \; \forall \, e = (u,v) \in E(G)$$

$$z \geq 0 \; \forall \, e \in E(G)$$

$$y_v \in \mathbb{R} \; \forall \, v \in V(G)$$

Let $(x_e)_{e \in E(G)}$ be a b-flow.

$$\Rightarrow (x_e)_{e \in E(G)} \text{ is valid for linear programming}$$

Theorem about complementary slacks:

Theorem 49. *x optimal $\Rightarrow \exists$ optimal solution $(2_e)_{e\in E(G)}, (y_v)_v \in V(G)$ of DLP with non-satisfied complementary slack.*

$$\Rightarrow X_e(Y_u - Y_v + z_e + c_e) = 0 \,\forall\, e \in E(G)$$
$$z_e(u_e - x_e) = 0 \,\forall\, e \in E(G)$$

$$0 \le -z_e \le c(e) + y_u - y_v \text{ for } e = (u,v) \in E(G) \text{ with } X(e) < u(e)$$

and

$$c(e) + y_u - y_v = -z_e \le 0 \text{ for } e = (u,v) \in E(G) \text{ with } X_e > 0$$
$$\Rightarrow (y_v)_{v\in V(G)} \text{ is a potential function}$$

because $c(u,v) + y_u - y_v \ge 0 \;\forall\, (u,v) \in E(G_f)$. We have 2 cases for $(u,v) \in E(G_f)$:

1. $x_{uv} < u_{uv}$ $(e = (u,v) \in E(G))$
2. $x_{uv} > 0$

$$-c(u,v) + y_v - y_u \ge 0 \text{ from } c_e + y_u - y_v \le 0$$

Can all be inverted! □

14.4.1 Analysis of MMCC (Min Mean Cycle Cancelling algorithm)

Denote the minimal average weight of a cycle in G_f with $\mu(f)$

$$\mu(f) := \min_{k \text{ is cycle in } G_f} \frac{\sum_{e\in E(K)} c_f(e)}{|E(K)|}$$

Theorem 50. *Let $f_1, f_2, \ldots, f_K$ be a sequence of b-flows such that for all $i = 1,2,\ldots,k-1$: $\mu(f_i) < 0$ and f_{i+1} originates from f_i by augmenting f_i along cycle K_i in G_{f_i} ($f_{i+1} = f_i \oplus K_i$).*

For now let K_i be a cycle with minimal average weight in G_f. Then the following statements hold:

$$\mu(f_i) \le \mu(f_{i+1}) \;\forall\, i$$
$$\mu(f_i) \le \frac{n}{n-2}\mu(f_c) \;\forall\, i < l$$

with property that $K_i \cup K_l$ contains at least one pair of edges of opposing direction.

Proof. Let i be static. Consider $(V(G), E(K_i) \cup E(K_{i+1})$ and remove edges of opposing direction. Results in a digraph H (edges occuring multiple times are counted twice).

H is subgraph of G_f : $(e \in E(K_{i+1}) \setminus E(G_f) \Rightarrow \overleftarrow{e} \in E(K_i))$. H is Eulerian graph $\Rightarrow$ can be decomposed into several cycles $\overline{k_1}, \overline{k_2}, \ldots, \overline{k_e}$ in G_{f_i}.

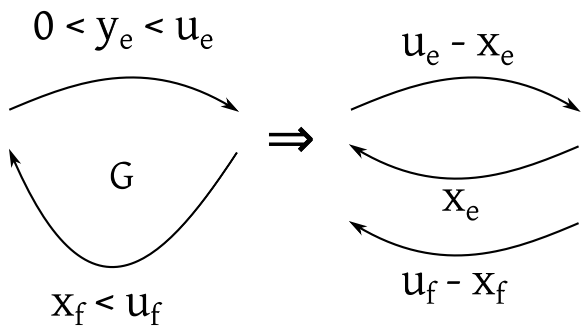


Figure 31: Proof of theorem 50

$$\frac{c(\overline{K_j})}{|E(K_j)|} \geq \mu(f_i) \; \forall \, 1 \leq j \leq l$$

- Equation 1: $|E(H)| = \sum_{j=1}^{l} \left|E(\overline{K_j})\right|$
- Weights of edges of opposing direction would cancel out each other:

$$c(E(H)) = \sum_{j=1}^{l} c(E(\overline{K_j})) \geq \sum_{j=1}^{l} \mu(f_i) \left|E(\overline{K_j})\right| = \mu(f) \left|E(H)\right|$$

Equation 2:

$$c(E(H)) = c(E(K_i)) + c(E(K_j)) = \mu(f_i) \left|E(K_i)\right| + \mu(f_{i+1}) \left|E(K_{i+1})\right|$$

From both equations it follows that

$$\rightarrow c(E(H)) \overset{\text{Equation 2}}{=} \mu(f_i) \left|E(K_i)\right| + \mu(f_i) \left|E(K_{i+1})\right| \overset{\text{Equation 1}}{\geq} \mu(f_i) \left|E(H)\right|$$
$$\geq \mu(f_i) \left|E(K_i)\right| + \mu(f_{i+1}) \left|E(K_{i+1})\right|$$

How can we show the last inequality $\geq$?

$|E(H)| \leq |E(K_i)| + |E(K_{i+1})|$ $\qquad$ $\mu(f_i) < 0$ holds otherwise algorithm terminates

$$\mu(f_i)\,|E(H)| \geq \mu(f_i)\,|E(K_i)| + \mu(f_i)\,|E(K_{i+1})|$$
$$\Rightarrow \mu(f_i) \leq \mu(f_{i+1})$$

Proving point (2) of Theorem 50: Consider without loss of generality i, j such that $\mu(f_i) \leq \frac{n}{n-2}\mu(f_e)$ and K_e and K_j have no edges of opposing direction $\forall\, i < j < l$.

Build H from $(V(G), E(K_i) \cup E(K_e))$ by removal of opposing edges.

H is subgraph G_{f_i}

$e \in E(K_l)\backslash G_{f_i} \Rightarrow \overleftarrow{e} \in E(K_i) \cup E(K_{i-1}) \cup \ldots \cup E(K_{l-1}) \Rightarrow \overleftarrow{e} \in E(K_i) \Rightarrow e \in E(H)$

This is a contradiction.

Eulerian H is decomposed into $\overline{K_1}, \ldots, \overline{K_t}$ cycles in G_{f_i}. Analogously as in point (1):

$$c(E(H)) \geq u(f)\,|E(H)|\,, c(E(H)) = \mu(f_i)\,|E(K_i)| + \mu(f_e)\,|E(K_l)|$$

Furthermore it holds that

$$\begin{aligned}
|E(H)| &\leq |E(K_i)| + |E(K_l)| - 2 \\
&= |E(K_i)| + |E(K_l)| - \frac{2n}{n} \\
&\leq |E(K_i)| + |E(K_l)| - \frac{2}{n}\,|E(K_e)| \qquad \text{with } n \leq |E(K_e)|
\end{aligned}$$

$$\Rightarrow E(H) \leq |E(K_i)| + \frac{n-2}{n}\,|E(K_e)|$$

$$\mu(f_i)\,|E(H_i)| \leq c(E(H_i)) = p(f_i)\,|E(K_i)| + \mu(f_e)\,|E(K_e)|$$
$$\mu(f_i)\,|E(K_i)| + \mu(f_i)\frac{n-2}{n}\,|E(K_2)|$$
$$\Rightarrow \mu(f_i) \leq \frac{n}{n-2}\mu(f_e)$$

Theorem 51. ***(Corollary)*** *During the MMCC algorithm $|\mu(f)|$ is decremented all $m \cdot n$ iterations by at least factor $\frac{1}{2}$.*

Proof. Let $K_i, K_{i+1}, \ldots, K_{i+m}$ be augmented cycles in m continuous iterations of the algorithm. Every cycle has a bottleneck edge $\Rightarrow$ $(n+1)$ bottleneck edges $\Rightarrow$ at least one repetition of an edge $\Rightarrow$ at least one pair K_j, K_l exists for $i \leq j < l \leq i+m$ which contain edges of opposing direction. From Theorem 50 it follows that

$$\mu(f_j) \leq \mu(f_j) \leq \frac{n}{n-2}\mu(f_e) \leq \frac{n}{n-2}\mu(f_{i+m})$$

$$|\mu(f_i)| \geq \frac{n}{n-2} |\mu(f_{i+m})|$$

$$\frac{n-2}{n} |\mu(f_i)| \geq |\mu(f_{i+m})|$$

After n such sequences of m iterations (= after $n \cdot m$ iterations), $\mu(f)$ is decremented by at least $\left(\frac{n-2}{n}\right)^n < \frac{1}{e^2} < \frac{1}{2}$.

$$|\mu(f_{\text{new}})| \leq \left(\frac{n-2}{n}\right)^n \mu(f_{\text{old}}) < \frac{1}{2} |\mu(f_{\text{old}})|$$

□

□

Theorem 52. *Assume $c : E(G) \to Q$ (without loss of generality: $c : E(G) \to \mathbb{Z}$) it holds that: after $\mathcal{O}(nm \log_2 n \, |c_{min}|)$ iterations the MMCC algorithm terminates with $c_{min} = \min \{\pm c_e | e \in E(G)\}$.*

Proof. **Correctness.** Start with b-flow f_0:

$$u(f_0) = \min_{K \text{ cycle in } G_{f_0}} \frac{\sum_{e \in E(K)} c(e)}{|E(K)|} \geq \min \frac{|E(K)| \cdot c_{\min}}{|E(K)|}$$

$$\mu(f_0) \geq c_{\min}$$

$$|\mu(f_0)| \leq |c_{\min}|$$

After $m \cdot n \cdot \log(n \, |C_{\min}|)$ iterations it holds that

$$|\mu(f)| < \frac{1}{2}^{\log_2 n |c_{\min}|} |\mu(f_0)| = \frac{1}{n \, |c_{\min}|} |\mu(f_0)| \leq \frac{1}{n}$$

$$|\mu(f)| < \frac{1}{n}$$

Show that

$$|\mu(f)| < \frac{1}{n} \Rightarrow \nexists \text{ negative cycle in } G_f$$

$$\mu(f) = \frac{\sum_{e \in E(K^*)} c(e)}{|E(K^*)|} > -\frac{1}{n}$$

$$|E(K^*)| \leq n \Rightarrow \frac{1}{|E(K^*)|} \geq \frac{1}{n}$$

$$\mu(f) \leq \frac{\sum_{e \in E(K^*)} c(e)}{n} \text{ if } \sum_{e \in E(K^*)} c(e) < 0$$

$$\Rightarrow \sum_{e \in E(K^*)} c(e) \geq n \cdot \mu(f) > n \left(-\frac{1}{n}\right) = -1$$

$$\Rightarrow \sum_{e \in E(K^*)} c(e) > -1$$

and $c(e) \in \mathbb{Z} \, \forall \, e \in E(G)$ where K^* are cycles of minimal average weight

$$\Rightarrow \sum_{e \in E(K^*)} c(e) \geq 0$$

This is a contradiction with $\sum_{e \in E(K^*)} c(e) < 0$.

Time complexity of MMCC. $\mathcal{O}(mn)$ per iteration (determination of K^* as optimal cycle) and $\mathcal{O}(nm \log n \, |C_{\min}|)$ iterations.

$$\Rightarrow \mathcal{O}(m^2 n^2 \log n \, |c_{\min}|) \text{ runtime}$$

$\square$

Theorem 53. *(Tarjan, Goldberg, 1989) The MMCC algorithm can be implemented with $\mathcal{O}(m^3 n^2 \log n)$ runtime.*

A proof for Theorem 53 is not provided.

This lecture took place on 24th of Nov 2014.

15 Successive shortest path algorithm

Theorem 54. *Let (G, u, b, c) an instance of MKFP and f be a b-flow with minimum costs. Let P be a shortest s-t-path in regards of c_f in G_f for any $s, t \in V(G_f)$. f' results from f by augmentation along P by $\gamma \leq \min\{u_f(e) : e \in E(P)\}$, hence*

$$f'(e) = \left\{ \begin{array}{ll} f(e) & e \notin E(P), \overleftarrow{e} \notin E(P) \\ f(e) + \gamma & e \in E(P) \\ f(e) - \gamma & \overleftarrow{e} \in E(P) \end{array} \right\}$$

Then f' is a b'-flow with minimum costs where

$$b'(v) = \left\{ \begin{array}{ll} b(v) & \forall \, v \notin \{s, t\} \\ b(v) + \gamma & v = s \\ b(v) - \gamma & v = t \end{array} \right\}$$

Proof. Assume f' is not a cost-minimal b'-flow. So there exists a cycle K with negative weights in $G_{f'}$. Consider $H_1 = \{v(G), E(K) \cup E(P)\}$. H shall be derived from H_1 by removing any pairs of edges of opposing direction.

Observation. H is subgraph of G_f because for $e \in E(K) \setminus E(G_f)$ it must hold that $\overleftarrow{e} \in E(G_f)$ and $(e, \overleftarrow{e})$ was already removed.

$$\Rightarrow \overleftarrow{e} \notin E(H)$$

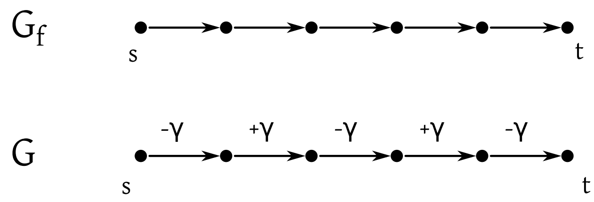


Figure 32: Proof of theorem 54

- $c(E(H)) = c(E(K)) + c(E(P)) < c(E(P))$
- $\deg_H(v) \equiv 0 \mod 2$ for $v \neq s, t$ (because of cycle and path).

H can be decomposed into cycles $k_1, \dots, k_l$ and a s-t-path.

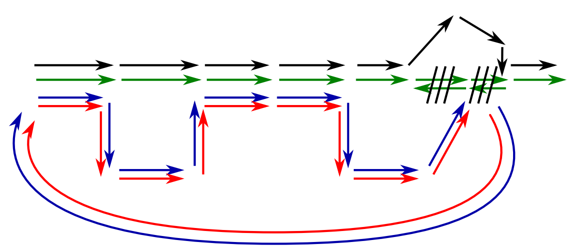


Figure 33: Proof of theorem 54 (cont.)

$$c(E(P_1)) \leq c(E(H)) = \underbrace{\sum_{i=1}^{l} c(E(K_i))}_{\geq 0} + c(E(P_1)) < c(E(P))$$

This is contradictory to the selection of P. $\square$

15.0.2 Initial flow for successively shortest path algorithm

- If c is conservative, start with 0-flow as optimal b-flow with $b = 0$.
- If c is not conservative, start with

$$f(e) = \begin{cases} u(e) & c(e) < 0 \\ 0 & c(e) \geq 0 \end{cases}$$

This is optimal for $b(v) = \sum_{\delta^+} f(e) - \sum_{\delta^-} f(e) \; \forall\, v \in V(G)$ because in G_f it holds that $c_f(e) = c(\overleftarrow{e}) \geq 0 \; \forall\, e \in E(G_f)$ with $\overleftarrow{e} \in E(G)$ and $c(\overleftarrow{e}) \leq 0$ and $c_f(e) \geq 0 \; \forall\, e \in E(G_f)$ with $e \in E(G)$ and $c(e) = c_f(e) \geq 0$.

15.1 Successively shortest path algorithm

Algorithm 14 Successively shortest path algorithm

Given. (G, u, b, c) network with $u : E(G) \to \mathbb{R}_+$ and $b : V(G) \to \mathbb{R}$ such that $\sum_{V(G)} b(v) = 0$. $c : E(G) \to \mathbb{R}$ is conservative

Find. Cost-minimal b-flow f or report "there is no b-flow in G"

1: Let $f(e) = 0 \quad \forall\, e \in E(G) \quad b'(v) = b(v) \; \forall\, v \in V(G)$
2: **if** $b'(v) = 0 \; \forall\, v \in V(G)$ **then**
3:     stop and return f
4: **else**
5:     chose $s \in V(G)$ with $b'(s) > 0$ and t with $b'(t) < 0$, which is reachable from s in G_f.
6:     **if** there does not exist s, t **then**
7:         stop and report there does not exist a b-flow in G.
8:     **end if**
9: **end if**
10: Determine a shortest s-t-path P in G_f (shortest path in regards of weights c_f).
11: Compute $\gamma = \min\{b'(s), u_f(e) \; \forall\, e \in P, -b'(t)\}$ (we are not allowed to overwrite b's).
12: Let $b'(s) = b'(s) - \gamma$ and $b'(t) = b'(t) + \gamma$.
13: Augment f along P: $f = f \oplus P$ (with value γ). **go to** 2

Theorem 55. *Let G be a digraph with $u : E(G) \to \mathbb{R}_+$ and $b : V(G) \to \mathbb{R}$*

$$\sum_{v \in V(G)} b(v) = 0$$

$$\exists \text{ } b\text{-flow in } G \Leftrightarrow \forall X \subseteq V(G) \text{ it holds that:}$$

$$\sum_{e \in \delta^+(X)} u(e) \geq \sum_{v \in V(X)} b(v)$$

The proof for Theorem 55 is given in the practicals.

Theorem 56. *If the algorithm terminates with "there does not exist a b-flow in G", this statement is correct.*

Proof. Let $X \subseteq V(G)$ such that $\forall v \in X$ it holds that, v is reachable from s in G_f, where f is the current flow resulting in "there does not exist a b-flow in G" and $s \in V(G)$ is a vertex with $b(s) > 0$ for which no $t \in V(G)$ with $b(t) < 0$ reachable from s in G_f exists.

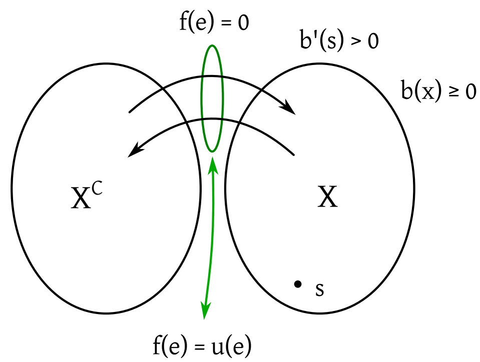


Figure 34: Proof of theorem 56

Then:

$$b'(v) \geq 0 \; \forall \, v \in X \text{ (otherwise we would have found some } t)$$

$$\sum_{e \in \delta^+(X)} u(e) \subset \sum_{e \in \delta^+(X)} f(e) + \underbrace{\sum_{e \in X \times X \cap E(G)}}_{\text{all edges in } X} - \sum_{e \in X \times X \cap E(G)} f(e)$$

$$= \sum_{v \in X} \left(\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) \right)$$

$$= \sum_{x \in X} b(v) - b'(v) < \sum_{v \in X} b(v)$$

because $b(v) - b'(v) \leq 0$ and one of them is smaller, because otherwise the algorithm would have terminated. f is a b'' flow where $b - b'' = b'$.

$$\sum u(e) < \sum b(v)$$

is a contradiction.

From Theorem 55 it follows then that no b-flow exists.

Does the algorithm terminate? Given $b(v) \in \mathbb{Z}\ \forall\, v \in V(G)$ and $u : E(G) \to \mathbb{Z}_+$ with the initial flow using integers. Then it holds that $b' \in \mathbb{Z}^{|V(G)|}$, $u_f \in \mathbb{Z}_+^{|E(G)|}$ and $y \in \mathbb{Z}_{*+}$ during the algorithm run.

$$\mathbb{Z}_{*+} = \{1, 2, 3, \ldots\}$$

After each iteration:

$$\underbrace{\sum_{v \in V(G)} |b'_{\text{new}}(v)|}_{\text{before augm.}} = \underbrace{\sum_{v \in V(G)} |b'_{\text{old}}(v)| - 2y}_{\text{after augm.}} \leq \sum_{v \in V(G)} |b'(v)| - 2$$

$$\text{after at most } \frac{1}{2} \sum_{v \in V(G)} \sum |b(v)| \text{ iterations, it holds that } b' = 0$$

$$B := \frac{1}{2} \sum_{v \in V(G)} |b(v)|$$

Time complexity.

$$\mathcal{O}(B \cdot m \cdot n)$$

B iterations, $\mathcal{O}(mn)$ per iteration $\Rightarrow$ pseudo-polynomial $\square$

Theorem 57. *If $u : E(G) \to \mathbb{Z}_+, b : V(G) \to \mathbb{Z}$ and c is conservative, the successive shortest path algorithm can be implemented in $\mathcal{O}(nm + B(m + n \log n))$.*

Proof. Assume wlog. there exists exactly one $v \in V(G) : b(v) > 0$. Otherwise consider $G' = (V(G) \cup \{s\}, E(G) \cup \{(s, v) : v \in V(G), b(v) > 0\})$.

$$\overline{b}(s) = \sum_{\substack{v \in V(G) \\ b(v) > 0}} b(v)$$

$$\overline{b}(v) = \sum_{\substack{\forall v \in V(G) \\ b(v) > 0}} 0$$

$\overline{b}(v) = b(v)$ for all other capacities u of new edges.

Assumption wlog: All vertices t with $b(t) < 0$ are only reachable from the single s satisfying $b(s) > 0$. All vertices not reachable from s and its incident edges can be removed. $\square$

Theorem 58. *In every i-th iteration of the algorithm a potential function π exists:*

$$\pi : V(G) \to \mathbb{R} \text{ in } G_{f_i} (c_{f_i}(u, v) + \pi(u) - \pi(v) \geq 0)\ \forall\, e \in E(G_{f_i})$$

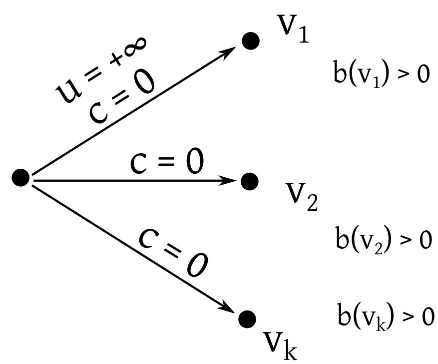


Figure 35: Proof of theorem 57

Proof. Proof by induction.

Induction base. $f = 0, c$ conservative, $G_{f_0} = G \Rightarrow \exists$ potential function π_0.

Induction step. Let f_{i-1} be a flow before the i-th iteration. From the induction hypothesis it follows that a potential function π_{i-1} exists. The shortest path computation shall happen in regards of $c_{f_{i-1}}(u,v)+\pi_{i-1}(u)-\pi_{i-1}(v)$. Let $l_i(v)$ be the length of the shortest s-v-path in $G_{f_{i-1}}$.

Let $\pi_i(v) = \pi_{i-1}(v) + l_i(v)\ \forall\, v \in V(G_{f_{i-1}})$. Show π_i is a potential function in G_{f_i}.

Consider $e = (x,y) \in E(G) \setminus E(P)$ ("old edges")

$$l_i(y) \leq l_i(x) + c_{\pi_{i-1}}(e) = l_i(x) + l(e) + \pi_{i-1}(x) - \pi_{i-1}(y)$$

$$\Rightarrow 0 \leq c(e) + \pi_{i-1}(x) + l_i(x) - (\pi_{i-1}(y) + l_i(y)) = c(e) + \pi_i(x) - \pi_i(y)$$

"new edges" or "augmenting edges": A new edge $\overleftarrow{e} = (y,x)$ is introduced on a path P from s to t in a graph $G_{f_{i-1}}$ for some edge (x,y). This new edge constitutes a new graph G_{f_i}.

$\forall\, e = (x,y) \in P$ it holds that $l_i(y) = l_i(x)+c_{\pi_{i-1}}(e) = l_i(x)+c(e)+\pi_{i-1}(x)-\pi_{i-1}(y)$

$$\Rightarrow 0 = c(e) + \pi_i(x) - \pi_i(y) \qquad \text{(analogously to "old" edges)}$$

$$\Rightarrow c_{\pi_i}(\overleftarrow{e}) = -c_{\pi_i}(e) = 0 \qquad \checkmark$$

$\mathcal{O}(mn)$ in first iteration (first potential function). for other $\mathcal{O}(B)$ iterations, we use Dijkstra's algorithm to compute the shortest paths in $\mathcal{O}(m + n \log n)$ per iteration and we compute the new potential function (as given in the induction step) in $\mathcal{O}(n)$

$$\Rightarrow O(mn + B(m + n\log n))$$

$\square$

This lecture took place on 25th of Nov 2014.

Algorithm 15 Capacity scaling algorithm

$$u(e) = +\infty \;\forall\, e \in E(G)$$

$$b(v) \in \mathbb{Z} \;\forall\, v \in V(G)$$

$$c : E(G) \to \mathbb{R} \text{ conservative}$$

Given. $(G, c, b), \sum_{v \in V(G)} b(v) = 0, c$ conservative
Find. b-flow with minimum costs or $\nexists$ b-flow in (G, c, b)

$b'(v) := b(v) \;\forall\, v \in V$
$f(e) := 0 \;\forall\, e \in E(G)$
if $b_{\max} > 0$ **then**
    $y = 2^{\lfloor \log b_{\max} \rfloor}$ with $(b_{\max} := \max\{b(v) : v \in V(G)\})$.
else
    return f
end if
if $b' = 0$ **then**
    return f
else
    select vertex s and t with $b'(s) \geq \gamma$
    select vertex $b'(t) \leq -\gamma$ such that t is reachable from s in G_f.
    if no such pair exists **then**
        go to 22
    end if
end if
Determine a s-t-path P in G_f with minimal weight (in regards of c_f).
Let $b'(s) := b'(s) - \gamma$
Let $b'(t) := b'(t) + \gamma$ (hence γ flow units are going through P)
$f := f \oplus P$
go to 8
if $y = 1$ **then return** "no b-flow exists"
else
    $y := \frac{\gamma}{2}$
    go to 8
end if

Theorem 59. *(Edmonds and Karp, 1972) The capacity scaling algorithm solves the MKFP with integers b, infinite capacities and conservative weights correctly.*

The algorithm can be implemented in $\mathcal{O}(n(m + n \log n) \log b_{max})$ runtime where $b_{max} := \max \{b(v) : v \in V(G)\}$.

Proof. **Correctness proof.**

Analogously as with successively shortest paths (for $\gamma = 1$ we use integer integrity). So we have to point out that in step 4, the augmentation is valid.

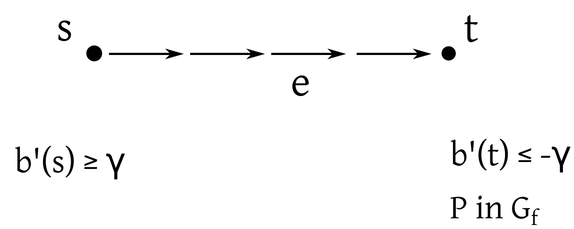


Figure 36: Capacity scaling algorithm proof

Runtime complexity proof.

A *phase* of the algorithm is a sequence of consecutive iterations which use the same value of γ. We prove: There are less than $4n$ augmentations in every phase.

Assumption. There exists a phase with $\geq 4n$ augmentations. Let f be a flow at the beginning of the phase. Let g be a flow at the end of the phase. $g - f$ is a flow. Let b'' be balance values of $g - f$:

$$b''(v) := \sum_{e \in \delta^+(v)} (g - f)(e) - \sum_{e \in \delta^-(v)} (g - f)(e)$$

- It holds that $\sum_{v \in V(G)} |b''(v)| \geq 8n\gamma$.

  Rationale. $f, f_2, f_3, \ldots, f_l = g$

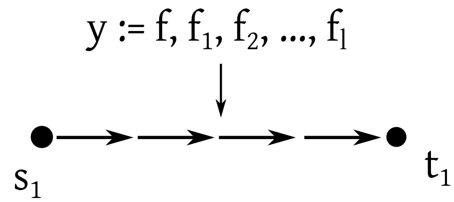


Figure 37: Capacity scaling algorithm proof rationale

$$
\begin{aligned}
b_{f_1}(s_1) &= b_f(s_1) + \gamma \\
b_{f_1}(v) &= b_f(v) \; \forall \, v \in V(G) \setminus \{t_1, t_2\} \\
b_{f_1}(t_1) &= b_f(t_1) - \gamma \\
\Rightarrow \sum_{v \in V(G)} |b_{f_1}(v)| &= \sum_{v \in V(G)} |b_f(v)| + 2\gamma
\end{aligned}
$$

After $\geq 4n$ iterations it holds that

$$\sum_{v \in V(G)} |b_g(v)| \geq \sum_{v \in V(G)} |b_f(v)| + 8n\gamma$$

$$\Rightarrow \sum_{v \in V(G)} |b''(v)| \geq 8n\gamma$$

- $$
\begin{aligned}
S &:= \{x \in V(G) : b''(x) > 0\} \\
S^+ &:= \{x \in V(G) : b''(x) \geq 2\gamma\} \\
T &:= \{x \in V(G) : b''(x) < 0\} \\
T^+ &:= \{x \in V(G) : b''(x) \leq -2\gamma\}
\end{aligned}
$$

  Is there a path from S^+ to T^+ in G_f? No, otherwise the 2γ phase would not be finished yet.

  Let $\mathcal{X}$ be the set of reachable vertices from S^+ in G_f. $\Rightarrow$ the total value of all sinks reachable from S^+ is $\geq n(-2\gamma) = -2n\gamma$.

  In figure 39, if the upper edge does not exist, it is saturated, but $u(0) = \infty$ $\Rightarrow$ edge does not exist; analogously the other direction.

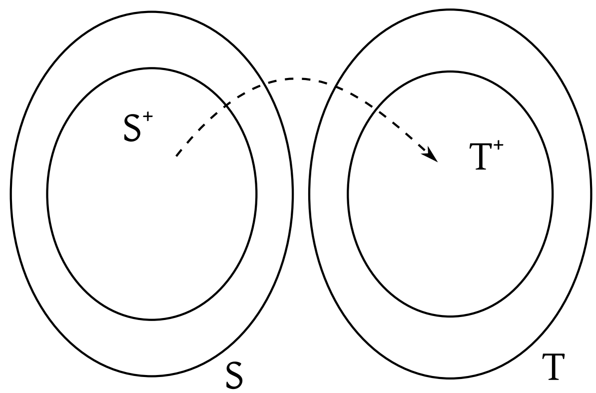


Figure 38: Capacity scaling algorithm: sets S, S^+, T and T^+

Because g-f is a b''-flow, it holds that

$$\sum_{x \in S^+} b''(x) < 2n\gamma$$

because in general it holds that

$$\sum_{x \in \mathcal{X}} b''(x) = 0$$

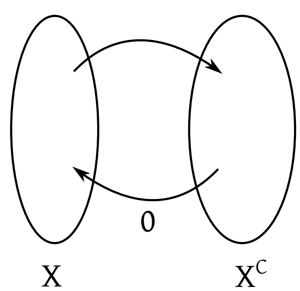


Figure 39: Capacity scaling algorithm: sets X and X^C

because

$$0 = \sum_{v \in V(G)} b''(v) = \sum_{v \in \mathcal{X}} b''(v) + \sum_{v \in \mathcal{X}^C} b''(v) = \sum_{x \in \mathcal{X}} b''(v)$$

$$= \sum_{v \in \mathcal{X}^C} \left(\sum_{e \in \delta^+(v)} \overline{f}(e) - \sum_{e \in \delta^-(v)} \overline{f}(e) \right) = \text{inner edges get cancelled} =$$

$$= \sum_{e \in \delta^+(x^c)} \underbrace{f(e)}_{=0} + \sum_{e \in (\mathcal{X}' \times \mathcal{X}^C) \cap E(G)} f(e) - \sum_{e \in (\mathcal{X}^C \times \mathcal{X}^C) \cap E(G)} f(e) = 0$$

$$\Rightarrow \sum_{v \in V(G)} |b''(v)| = 2 \sum_{x \in S} b''(v) = 2 \left(\sum_{x \in \delta^+} b''(v) + \sum_{x \in S \setminus S^+} b''(v) \right)$$

$$< 2(2n\gamma + 2n\gamma) = 8n\gamma$$

This is a contradiction to our assumption.

Hence there are $< 4n$ augmentations per phase. The number of phases equals to the number of halvings of $2^{\lfloor \log b_{\max} \rfloor}$ until $\gamma < 1$ is reached.

$$\Rightarrow \text{number of phases} \leq \log b_{\max} + 1$$

$$\Rightarrow \mathcal{O}(n \log b_{\max}) \text{ iterations}$$

with reduced costs from Theorem 55

$$\mathcal{O}(mn + (m + n \log n) n \log b_{\max})$$

$$= \mathcal{O}((m + n \log n) n \log b_{\max})$$

□

16 Time-dependent dynamic flow

This lecture took place on 1st of Dec 2014.

- Transit time per edge
- Value of flow over edge e can vary in time process

$$f_e(t) \qquad \text{flow values at timestamp } t \text{ at edge } e$$

Definition. Let (G, u, s, t) be a network with capacities $u : E(G) \to \mathbb{R}_+$, source s, sink t and transit times $l : E(G) \to \mathbb{R}_+$ $(e \mapsto l(e))$ and a time horizon $[0, T]$ with $T \in \mathbb{R}_+$. Then a timedependent s-t-flow s is a Lebesgue-measurable function $f_e : [0, T] \to \mathbb{R}_+ \; \forall\, e \in E(G)$ with all properties:

- $f_e(\zeta) \leq u(e) \; \forall\, e \in E(G) \; \forall\, \zeta \in [0, T]$
- $\mathrm{ex}_f(v, a) := \sum_{e \in \delta^-(v)} \int_0^{\max\{0, a - l(e)\}} f_e(\zeta)\, d\zeta - \sum_{e \in \delta^+(v)} \int_0^a f_e(\zeta)\, d\zeta \geq 0 \; \forall\, v \in V(G) \setminus \{s\} \; \forall\, a \in [0, T]$

$\text{value}(f) := \mathrm{ex}_f(t, T)$ is called value of flow at timestamp T.

16.1 Max-Flow-over-time problem (MFoTP)

Given. (G, u, s, t, l), $T \in \mathbb{R}_+$

Find. Determine a timedependent flow $f_e : [0, T] \to \mathbb{R}_+ \ \forall e \in E(\zeta)$ with maximum value f

Theorem 60. *(Ford, Fulkerson, 1958) The MFoTP can be solved with the same time complexity like MKFP.*

Proof. MFoTP can be reduced to (static) MKFP. Without loss of generality: There are no edges ending in s in G (flow delay!). Construct a new network $(G', u, 0, c)$.

$$G' = (V(G), E(G) \cup \{(t, s)\})$$

$$u : E(G') \to \mathbb{R}_+$$

with $u(e)$ as the same like in (G, s, t, u, l) for $e \in E(G)$ and $u(t, s) = \sum_{e \in E(G)} u(e)$.

$$c : E(G') \to \mathbb{R}$$

$$c(e) = \begin{cases} l(e) & e \in E(G) \\ -T & e = (t, s) \end{cases}$$

Determine a circulation f' with minimum costs in $(G', u, 0, c)$.

Theorem of Gallai: There exists a family $\mathcal{C}$ of cycles in g' with $h : \mathcal{C} \to \mathbb{R}_{+*}$, $K \mapsto h(K) > 0$ with $|\mathcal{C}| = \mathcal{O}(|E(G')|)$ and $f(e) = \sum_{K \in \mathcal{C}, e \in K} h(K) \ \forall e \in E(G')$.

f' has minimum costs $\to c(K) \leq 0 \ \forall K \in \mathcal{C}$.

In the other case if K_* with $c(K_*) > 0$ exists, consider f'' with $f''(e) = \sum_{K \in \mathcal{C} \setminus \{K_*\}, e \in K} h(K) \ \forall e \in E(G')$. f'' is circulation.

$$c(f'') = \sum_{e \in E(G')} f''(e) \cdot c(e) = \sum_{K \in \mathcal{C} \setminus \{K_*\}} c(K) < \sum_{K \in \mathcal{C}} c(K) = c(f')$$

This would be a contradiction.

$$(t, s) \in K \ \forall K \in \mathcal{C} \text{ because } (t, s) \text{ is the only edge with negative costs}$$

Let $e = (v, w) \in K$. Denote d_e^K the length of the path of s to v in K in regards of costs c. Define

$$f_e^*(\zeta) = \sum_{\substack{k \in \mathcal{C}, c(K) < 0 \\ e \in K, d_e^K \leq \zeta \leq d_e^K - c(K) \\ \zeta \leq T - l(e) - d_{w,t}^K \\ d_{w,t}^K + \zeta + l(e) \leq T}} h(K) \ \forall e \in E(G) \ \forall 0 \leq \zeta \leq T$$

Define

$$c(k) = d_e^K + l(e) + d_{w,t}^K - T$$

$$d_e^K - c(K) = T - l(e) - d_{w,t}^K$$

When is s_0 a $f_e^*(\zeta)$ dynamic flow?

$$f_e^*(\zeta) = \sum h(K) \leq \sum_{\substack{K \in \mathcal{C} \\ c(K)<0 \\ e \in K}} h(K) = f'(e) \leq n(e) \;\forall\, e \;\forall\, \zeta \in [0,T] \tag{5}$$

$$\mathrm{ex}_f(v,a) = \sum_{e \in \delta^-} \int_0^{\max\{0, a-l(e)\}} \cdots - \sum_{e \in \delta \ldots} \int_0^a \cdots \overset{!}{\geq} 0 \tag{6}$$

$$\sum_{e \in \delta^-(v)} \sum_{\substack{K \in \mathcal{C} \\ c(K)<0 \\ e \in K}} h(K) - \sum_{e \in \delta^+(v)} \sum_{\substack{K \in \mathcal{C} \\ c(K)<0 \\ e \in K}} h(K) = \sum_{e \in \delta^-(v)} f'(e) - \sum_{e \in \delta^+(v)} f'(e) = 0$$

because f' is circulation

$$\mathrm{value}(f^*) = \mathrm{ex}_{f^*}(t,T) = \underbrace{\sum_{e \in \delta^-(t)} \int_0^{\max\{0, T-l(e)\}} f_e^K(\zeta)\, d\zeta}_{A} \overset{!}{=} - \sum_{e \in E(G')} c(e) f'(e)$$

$$A = \sum_{e \in \delta^-(t)} \int_0^{\max\{0, T-l(e)\}} \sum_{\substack{K \in \mathcal{C} \\ c(K)<0 \\ e \in K \\ d_e^K \leq \zeta \leq d_e^K - c(K)}} h(K)\, d\zeta$$

$$= \sum_{e \in \delta^-(t)} \sum_{\substack{K \in \mathcal{C} \\ c(K)<0 \\ e \in K}} h(K) \int_{d_e^K}^{d_e^K - c(K)} 1\, d\zeta = \sum_{e \in \delta^-(t)} \sum_{\substack{K \in \mathcal{C} \\ c(K)<0 \\ e \in K}} h(K)(-c(K))$$

$$= \sum_{e \in \delta^-(t)} \sum_{\substack{K \in \mathcal{C} \\ c(K)<0 \\ e \in K}} h(K) \left(- \sum_{e \in K} l(e) + T \right)$$

$$= \sum_{e \in E(G)} -l(e) \underbrace{\sum_{\substack{K \in \mathcal{C} \\ c(K)<0 \\ e \in K}} h(K)}_{f'(e)} + T \left(\sum_{e \in E(G)} f'(e) - \sum_{e \in E(G)} f'(e) \right)$$

$$= \sum_{e \in E(G)} -l(e) f'(e)$$

$\square$

17 Matchings

17.1 Definitions and optimality criterion

Definition. $G = (V, E)$ is undirected. $M \subseteq E$ is called *matching* if $\forall e_1, e_2 \in M$ and $e_1 \cap e_2 = \varnothing$. A v is called "matched by M" if $\exists e \in M$ with $v \in e$. A matching M is called *perfect* if every vertex of M is matched by M. A matching M is called *maximal* if for every matching M', $|M'| \leq |M|$. The *matching number of* G is defined as

$$V(G) = \max\{|M| : M \ldots \text{matching}\}$$

A *vertex cover* is a subset $C \subseteq V(G)$ such that $e \cap C \neq \varnothing\ \forall e \in E(G)$. A *minimal vertex cover* is a vertex cover of minimal cardinality.

The *vertex cover number of* G is defined as

$$\mathcal{C}(G) = \min\{|C| : C \text{ is vertex cover}\}$$

Observation. For a triangle (3 vertices, 3 edges):

$$\mathcal{C}(K_3) = 2 \qquad V(K_3) = 1$$

Let M is a matching. Let C is a vertex cover.

$$V(G) = \max_{M \text{ matching}} |M| \leq \min_{C \text{ vertex cover}} |C| = \mathcal{C}(G)$$

$$\Rightarrow V(G) \leq \mathcal{C}(G)$$

For bipartite graphs, equality holds (König, 1938).

Definition. Let G be an undirected graph and M a matching. A path in G is called m-alternating if its edges are alternatingly assigned to the matching and not assigned. Hence

$$P = (v_0, v_1, v_2, \ldots, v_k)$$

$$v_{i-1}, v_i \in M \to v_i, v_{i+1} \notin M \text{ and } v_{i-1}, v_i \notin M \to v_i, v_{i+1} \in M\ \forall 1 \leq i \leq k$$

If a M-alternating path starts and ends with a non-matched vertex, it is called M-augmenting.

Example.

$$P = (3, 4, 1, 5, 2, 6, 8, 7)$$

Let $M' = M \triangle P$ where M is a matching and P is an M-augmenting path.

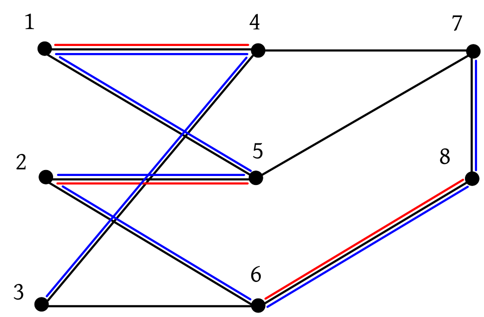


Figure 40: M augmentation example

- We show: M' is a matching.

$$M' = (M \setminus P) \cup (P \setminus M)$$

  Let $e_1, e_2 \in M'$

  - $e_1, e_2 \in M \setminus P \Rightarrow e_1 \cap e_2 = \varnothing$
  - $e_1, e_2 \in P \setminus M \Rightarrow e_1 \cap e_2 = \varnothing$
  - $e_1 \in M \setminus P, e_2 \in P \setminus M \Rightarrow e_1 \cap e_2 = \varnothing$

- We show: $|M'| = |M| + 1$

$$M' = (M \setminus P) \cap (P \setminus M)$$

$$|M'| = |M \setminus P| + |P \setminus M|$$

$$M = (M \cap P) \cap (M \setminus P)$$

$$|M| = |M \cap P| \cap |M \setminus P|$$

We conclude:

$$\Rightarrow \underbrace{|M \cap P|}_{\text{red edges}} = \underbrace{|P \setminus M| - 1}_{\text{white edges}}$$

P is M-augmented.

This lecture took place on 2nd of Dec 2014.

Theorem 61. *(Berge, 1957) Let M be a matching in (G, E). M is maximal if and only if there is no M-augmenting path in G.*

Proof. Direction ,,$\Rightarrow$": There exists a M-augmenting path $\Rightarrow$ M is not maximal (see previous section).

Direction ,,$\Leftarrow$": There does not exist a M-augmenting path $\Rightarrow$ M is maximal.

Assumption: M is not maximal $\Rightarrow \exists M'$ matching with $|M'| > |M|$.

Consider $M \triangle M'$ $(= M \setminus M' \cup M' \setminus M)$.

$$\deg_{M \triangle M}(\gamma) \leq 2 \; \forall \, v \in V(G)$$

$$\Rightarrow (V(G), M \triangle M') \text{ can be decomposed in cycles and paths}$$

$$|V_{\text{matched}}(M')| \geq |V_{\text{matched}}(M + 2)|$$

$$\Rightarrow \exists \text{ at least 2 vertices } v_1, v_2 \text{ which are matched in } M' \text{ but not in } M$$

$v_1(V_2)$ is end point in one of the paths from the decomposition

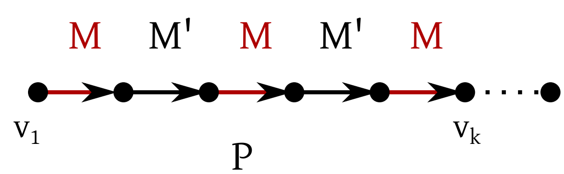


Figure 41: Theorem 61

- v_k is not matched by M'. Number of matched vertices by M' in P is the number of matched vertices in P by M.
- From the number of matched vertices it follows that a path exists. v_k matched by M' but not by M. P is M-augmented $\Rightarrow M \triangle P$ is a matching larger than M

This is a contradiction to our assumption. $\square$

Algorithmic idea. Start with arbitrary matching $M = M_0$ (eg. $M_0 = \varnothing$). While there exists an M-augmented path P, repeat

$$M := M \triangle P$$

Output is our maximal M.

17.2 Matchings in bipartite graphs

The $M\triangle P$ (Maximal matching problem) can be written as Max-Flow problem.
Let $G = (A \mathbin{\dot\cup} B, E)$.

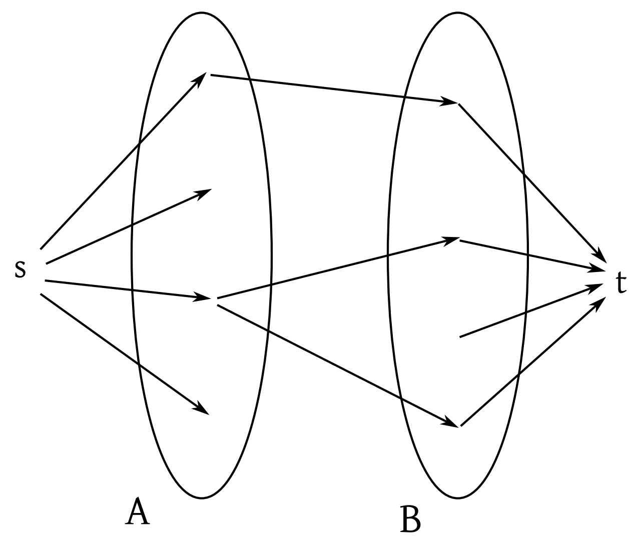


Figure 42: Bipartite matching

Let (G', u, s, t) be a network where G' is a digraph.

$$s, t \notin A \mathbin{\dot\cup} B \qquad s \neq t$$

$$V(E') = A \mathbin{\dot\cup} B \mathbin{\dot\cup} \{s, t\}$$

$$E(G') = \{(s, a) : a \in A\} \cup \{(b, t) : b \in B\} \mathbin{\dot\cup} \{(a, b) \in E(G), a \in A, b \in B\}$$

Observation.

- Every matching M in G corresponds to a complete flow F_M in G'.
- Every flow f in (G', s, t, u) corresponds to one matching M_f in G.

- $$f_M(e) = \begin{cases} 1 & e = (a,b) \in A \times B, e \in M \\ 0 & e = (a,b) \in A \times B, e \notin M \\ 1 & e = (s,a), a \in A, \text{matched} \\ 0 & e = (s,a), a \in A, \text{unmatched} \\ 1 & e = (b,t), b \in B, \text{matched} \\ 0 & e = (b,t), b \in B, \text{unmatched} \end{cases}$$

$$\text{value}(f_M) = \sum_{a \in A} f(s,a)$$

$$= |\{a : a \in A, f(s,a) = 1\}| = |\{a : a \in A, \text{matched}\}| = |M|$$

- Let f be a s-t-flow in (G', u, s, t) with integer values.

$$\Rightarrow f(e) \in \{0,1\} \ \forall e \in E(G')$$

Define $M_f = \{e \in A \times B : f(e) = 1\}$. M_f is matching, because $e_1 = (a, b_1) \in M$ and $e_2 = (a, b_2) \in M$

$$\rightarrow \sum_{l \in \delta^-(a)} f(e) = f(s,a) < f(e_1) + f(e_2) = 2 \leq \sum_{e \in \delta^+(a)} f(e)$$

This is a contradiction. f is not a flow.

Analogously as in the first bullet item: value$(f) = |M_f|$. MMP in bipartite graphs is solvable with $\mathcal{O}(mn)$ runtime, eg. with Ford-Fulkerson in n iterations (flow extension by 1 unit per iteration) and $\mathcal{O}(m)$ runtime for determination of (s,t) path in G_f.

$$v(G) \leq \zeta(G)$$

In bipartite graphs *equivalence* holds (König, 1937).

Theorem 62. *Let $G = (v_1 \cup v_2, E)$ be a bipartite graph. Then it holds $v(G) = \zeta(G)$.*

Proof. $v(G) \subseteq \zeta(G)$ is trivial to show. Show that $\zeta(G) \geq v(G)$.

Consider MMP as MFP (as in Observation). Let f be a maximal s-t-flow of integer values. Let $\delta(s)$ be the corresponding minimal cut.

$$u(\delta(s)) = \text{value}(f)$$

There does not exist $e \in (S \cap V_1) \times (S' \cap V_2)$.

Assumption. There exists $e = (a,b)$. Case distinction:

1. $f(e) = 0 \Rightarrow e \in E(G_f) \Rightarrow b \in S$. This is a contradiction.
2. $f(e) = 1 \Rightarrow \sum_{e\in\delta^-(a)} f(e) - \sum_{e\in\delta^+(a)} f(e) = 0$

$$f(s,a) = \sum_{e\in\delta^+(a)} f(e) \geq 1 \Rightarrow f(s,a) = 1$$

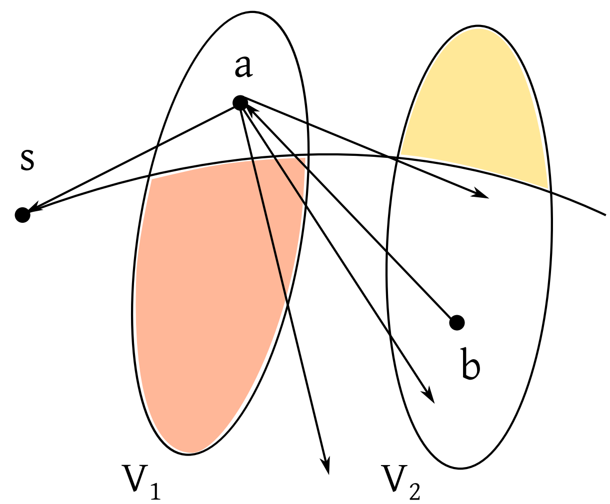


Figure 43: Bipartite matching – Residual graph

Hence a is only reachable from b. $b \notin S \Rightarrow a$ is not reachable. Contradiction.

$$\Rightarrow \nexists e \in (S \cap V_1) \times (S^C \times V_2)$$

Construct cover: $C = (V_1 \cap S^C) \cup (V_2 \cap S)$.
Compare value(f) and

$$|c| = \left|V_1 \cap S^C\right| + |V_2 \cap S| = \left|V_1 \cap S^C\right| + \{(a,t) : a \in S\}$$

$$\text{value}(f) = u(\delta(s)) = \sum_{e\in\delta^+(\mathcal{S})} u(e) = \left|\delta^+(\mathcal{S})\right| \overset{!}{=} \left|S^C \cap V_1\right| + |S \cap V_2|$$

$\square$

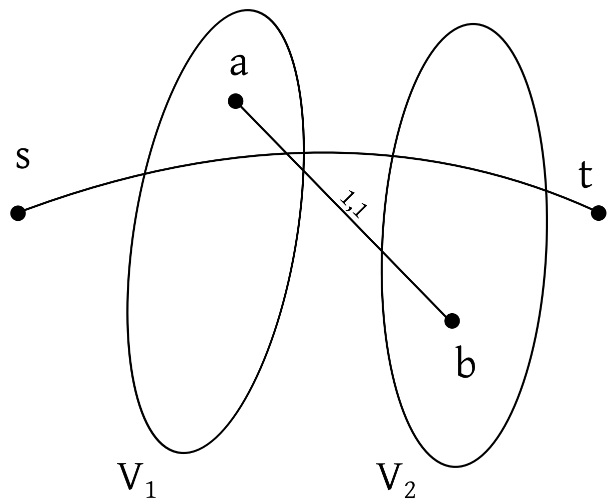


Figure 44: Bipartite graph

Theorem 63. *(Hall's marriage condition.) Let G be a bipartite graph $(A \cup B, E)$ then G has a covering matching for A if and only if $|\Gamma(X)| \geq |X| \ \forall X \subseteq A$ where $\Gamma(X) = \{b \in B : \exists a \in X, (a, b) \in E(G)\}$.*

Theorem 64. *(Marriage corollary.) Let G be a bipartite graph with $V(e) = A \cup B$ and $|A| = |B|$. G has a perfect matching if and only if $\forall X \subseteq A$ with $|\Gamma(X)| \geq |X|$ holds.*

17.3 Theorem of Tutte (in general graphs)

$$X \subseteq V(G)$$

$G \setminus X$ results from G if X and all incident edges with X are removed.

Let $q_G(X)$ be the number of even (in terms of number of vertices) connected components in $G \setminus X$. Then we can observe: $q_G(X) > |X| \Rightarrow \nexists$ perfect matching in G.

$$\Rightarrow \underbrace{q_G(X) \leq |X| \ \forall X \subseteq V(G)}_{\text{Tutte condition}}$$

$\ldots$ is necessary and sufficient for existence of a perfect matching in G

Definition. A graph G is called *factor-critical*, if $G \setminus \{v\}$ (also denoted $G \setminus v$) contains a perfect matching $\forall v \in V(G)$. A matching M is called *almost perfect* if G has exactly one unmatched vertex.

Theorem 65. *Let G be a graph, then*

$$q_G(X) - |X| \equiv |V(G)| \mod 2 \;\forall X \subseteq V(G)$$

Proof.

$$V(G) = |X| + \sum_{i=1}^{k} |X_i|$$

- Let X_i for $1 \leq i \leq k$ be the connected components of $G \setminus X$. Without loss of generality the even connected components are the $q_G(X)$ first ones.

$$|V(G)| \equiv |X| + \sum_{i=1}^{d_G(X)} |X_i| \mod 2$$

- Because

$$|X_i| \equiv 1 \mod 2 \;\forall 1 \leq i \leq q_G(X)$$

it holds that

$$\sum_{i=1}^{q_G(X)} |X_i| \equiv q_G(X) \mod 2$$

From both statements it follows that

$$\begin{aligned}
|V(G)| &\equiv |X| + q_G(X) \mod 2 \\
V(G) - 2|X| &\equiv |X| + q_G(X) - 2|X| \mod 2 \\
V(G) - 2|X| &\equiv q_G(X) - |X| \mod 2 \\
\Rightarrow V(G) &\equiv q_G(X) - |X| \mod 2
\end{aligned}$$

□

Theorem 66. *Let G be a graph. G contains a perfect matching if and only if the Tutte condition is satisfied, hence $q_G(X) \leq |X| \;\forall X \subseteq V(G)$.*

Proof. **Necessary?** By observation a necessary condition.

Sufficient? Assume Tutte condition is satisfied. Show that a perfect matching exists in G. Proof by induction over $|V(G)|$.

$$|V(G)| \leq 2 \text{ statement holds}$$

For $V(G) = 1$:

$$|X| = 0 \qquad |X| = 1 \qquad q_G(\varnothing) = 1 \overset{!}{<} 0$$

For $V(G) = 2$:

- No edge between two vertices.

$$X = \varnothing \qquad q_G(X) = 2 > 0 \quad \text{(no perfect matching)}$$

- Edge between both vertices.

$$\begin{aligned} X = \varnothing &\qquad q_G(X) = 0 \leq 0 \qquad \checkmark \\ |X| = 1 &\qquad q_G(X) = 1 \leq 1 \qquad \checkmark \\ |X| = 2 &\qquad q_G(X) = 0 \leq 2 \qquad \checkmark \end{aligned}$$

$\square$

This lecture took place on 9th of Dec 2014.

Theorem 67. *(Theorem by Tutte.) Let G be a graph with a perfect matching $\Leftrightarrow q_G(x) \leq |X| \; \forall X \subseteq V(G)$ (tutte condition).*

Less formally: A graph $G = (V, E)$ has a perfect matching if and only if every subgraph G' of any $U \subseteq V(G)$ has at most $|U|$ connected components with an odd number of vertices.

The Theorem by Tutte is a generalization of Theorem 63 (Hall's marriage condition).

Proof. Induction over $|V(G)|$ for non-trivial direction $\Leftarrow$. Induction base $|V(G)| = 1$ (done).

Induction base. Condition holds $\forall G$ with $|V(G)| < n$.

Induction step. Condition holds for G with $|V(G)| = n$.

Let G with $|V(G)| = n$ such that tutte condition is satisfied.

Observation 1. ($|V(G)|$ is even) If $|V(G)|$ is odd: Let $X = \varnothing$ and $q_G(\varnothing) =$ number of connected, even components in G.

$$|V(G)| = \sum_{i=1}^{q_G(\varnothing)} |V(K_i)| + \sum_{i=q_G(\varnothing)+1}^{l} |V(K_i)| \Rightarrow |V(G)| \equiv q_G(\varnothing) \mod 2$$

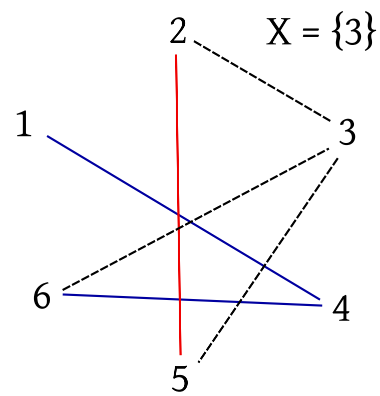
$$|V(K_i)| \equiv 1 \mod 2$$

$$\Rightarrow q_G(\varnothing) \geq 1 > 0 = |\varnothing|$$

The Tutte condition is not satisfied. Contradiction.

Observation 2. Every $X = \{v\}$ with $v \in V(G)$ is barrier. From the last proposition it follows

$$q_G(V) - |X| \equiv |V(G)| \mod 2 \; \forall X \subseteq V(G)$$



Especially:

$$
\left.\begin{array}{r} q_G(\{v\}) - |\{v\}| \text{ is even} \Rightarrow q_G(\{v\}) \text{ even} \\ \Rightarrow q_G(\{v\}) \geq 1 \\ \text{from tutte condition: } q_G(\{v\}) \leq |\{v\}| = 1 \end{array}\right\} \Rightarrow q_G(\{v\}) = 1 = |\{v\}| \Rightarrow \{v\} \text{ is barrier}
$$

Let X is a cardinality-maximal barrier; hence $q_G(x) = |X|$.

Observation 3. $G - X$ has no even connected component. Assume there exists an even connected component K. Let $v \in V(K)$. Consider $X \cup \{v\}$.

$$q_K(\{v\}) - |\{v\}| \equiv |V(K)| \mod 2 \Rightarrow q_K(\{v\}) \text{ is odd}$$

$$q_G(X \cup \{v\}) = q_K(\{v\}) + q_G(X) \geq 1 + |X| = |X \cup \{v\}|$$

with tutte condition it follows that $q_G(X') = |X'| \Rightarrow |X'| > |X|$ where X' is barrier.

$$q_G(X) := \text{number of odd connected components in } G - X$$

$$q_G(\{3\}) = 1$$

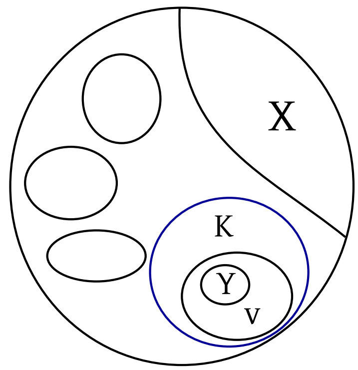
Definition. A set $X \subseteq V(G)$ which satisfies the tutte condition with $=$ is called *barrier.*

Observation 4. Every even connected component of $G - X$ is factor-critical.

Assume there exists a connected component K of $G - X$, which is not factor-critical. $\Rightarrow \exists v \in V(K)$ such that $K - \{v\}$ does not have a perfect matching. Because $|V(K - \{v\})| < |V(G)|$ the induction hypothesis holds for $K - \{v\}$. The tutte condition is *not* satisfied in $K - \{v\}$.

$$\Rightarrow \exists Y \subseteq V(K) \setminus \{v\} \text{ with } q_{K-\{v\}}(Y) > |Y|$$

Consider $X \cup Y \cup \{v\}$ ($X, Y, \{v\}$ pairwise disjoint)



$$
\begin{aligned}
q_G(X \cup Y \cup \{v\}) &= q_G(X) - 1 + q_K(Y \cup \{v\}) \\
&= |X| - 1 + q_{K-\{v\}}(Y) \geq |X| - 1 + |Y| + 2 \\
&= |X| + |Y| + 1 = |X \cup Y \cup \{v\}| \\
&\Rightarrow X \cup Y \cup \{v\} \text{ is barrier}
\end{aligned}
$$

$|X|$ is largest barrier.

End of proof for observation 4.

Let G' be a bipartite graph with $v(G') = X \cup Z$ where

$$Z = \{z_i : z_i \text{ corresponds to contracted } K_i,\ 1 \leq i \leq q_E(X)\}$$

Inner vertices in X get deleted. Edges between X and Z are created by contraction.

We show G' satisfies the Hall condition, hence

$$\forall\, A \subset Z \text{ satisfies } |\Gamma(A)| \geq |A|$$

Assume that Hall condition is unsatisfied $\Rightarrow \exists A \subseteq Z$ with $|\Gamma_{G'}(A)| < |A|$.

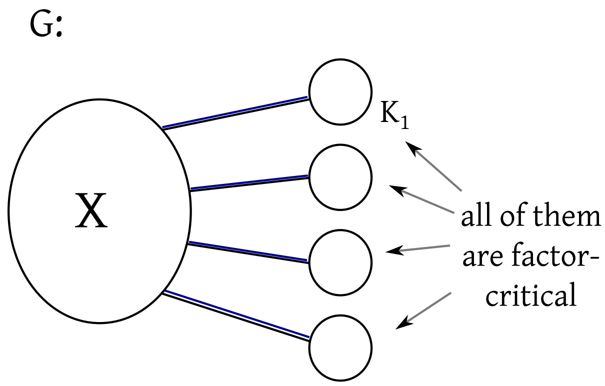
$$q_G(\Gamma_G, (A)) \underbrace{\geq}_{?} |A| > |\Gamma_G(A)|$$

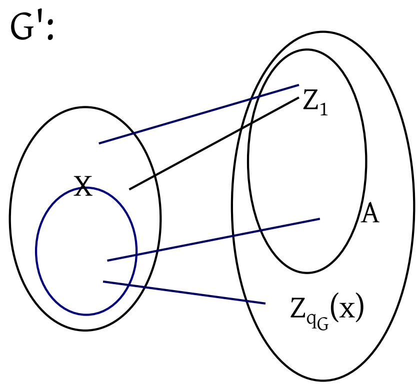
Tutte condition is unsatisfied. Contradiction.

$\exists$ matching in G' that matches all vertices in Z hence all vertices in X because $|X| = q_G(X)$ can be fully matched inside of every K_i, because K_i is factor-critical.

$$\forall\, 1 \leq i \leq q_G(X)$$

Hence a perfect matching in G is created. $\square$





18 Blossom algorithm

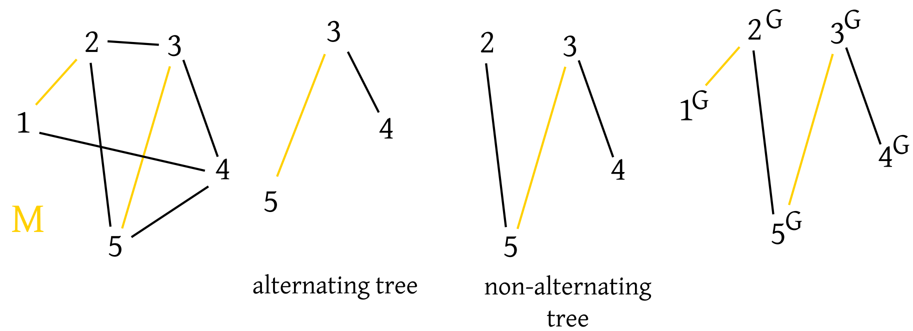
This algorithm by Edmonds determines a perfect (or cardinality-maximal) matching in any arbitrary graph.

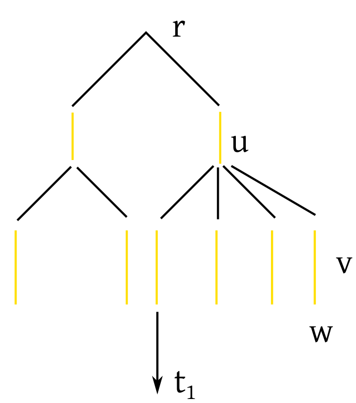
Definition. Let G be a graph and M be a matching. A tree with root r in T is called *alternating*, if

- the root r is unmatched.
- every path of r to any vertex v in T is an alternating path.
- all vertices $v \in V(T) \setminus \{r\}$ are matched with edges in T.

The root is per definition an *even vertex*. Every vertex in T with even distance from r is called *even vertex*. All remaining vertices are *odd vertices*.

An alternating forest F where





- Every connected component is an alternating tree
- Every exposed vertex is a root of one alternating tree

Observation 1. Let $A(T)$ be the set of odd vertices. Let $B(T)$ be the set of even vertices. $|B(T)| = |A(T)| = 1$.

Let $v \notin V(T)$. Let $u \in B(T)$. Case distinction:

- v is matched by $(r, w) \notin E(T)$ with $T' = T + (u, v, w)$. T' is alternating and greater than T.
- v_1 is not matched. $(u_1, v_1) \notin E(T)$ with $(u_1, v_1) \notin M$. $P = (r, \ldots, u_1, v_1)$ is an augmenting path. Hence $M \oplus P$ is a larger match than M.

Definition. Let G be a graph and M be a matching of G and T be an alternating tree in terms of M in G. T is called *degenerated* (dt. verkümmert) if

$\forall\, u \in B(T)$, $(u, v) \in E(G) \Rightarrow v \in A(T)$ is implied.

Theorem 68. *Let M be a matching in M in G and T be an alternating degenerated tree. Then G has no perfect matching.*

Proof. We show that the tutte condition is violated. Let $X := A(T)$.

$$q_G(X) \geq |B(T)| > |A(T)| = |X|$$

Last case: $\exists$ BB-edges hence edges $(v_1, v_2) \in E(G)$ with $v_1, v_2 \in B(T)$. Such an edge closes an odd cycle with the tree. Such a cycle will be contracted. $\square$

Algorithm 16 Edmonds blossom algorithm

Given. Graph G and matching M in G
Find. Perfect matching in G or report "$\nexists$ perfect matching in G"
Recall. $A(T)$ is the set of odd vertices. $B(T)$ is the set of even vertices.

$M' := M$, $G' := G$
if all vertices in G' are matched **then**
    return perfect matching M
else
    r is unmatched
    $T := (\{r\}, \varnothing)$, $B(T) := \{r\}$, $A(T) := \varnothing$
end if
while $\exists (v, w) \in E(G')$ with $v \in B(T), w \notin A(T)$ **do**
    if $w \notin V(T)$ and w is unmatched **then**
        use (v, w) to augment M'
        if $\exists$ no odd vertex in G' **then**
            return perfect matching M'
        else
            replace T by $(\{r\}, \varnothing)$ where r is a (new) unmatched vertex in G'
        end if
    else if $w \notin V(T)$ and w is matched in regards of M' **then**
        use (v, w) and the matching-edge incident with w to extend T.
    else if $w \in B(T)$ **then**
        use (v, w) to contract
        update M', T' and G'
    end if
end while
return "G has no perfect matching"

This lecture took place on 18th of Dec 2014.

Theorem 69. *Let C be an odd cycle in G and let G' be a graph which results by contraction of C. Let M' be a matching in G'. Then there exists a matching M in G with*

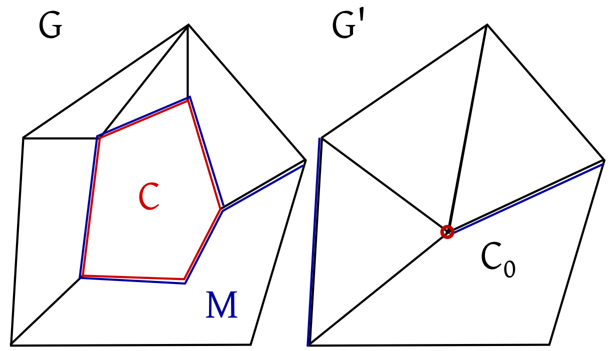


Figure 45: Edmonds blossing contraction: example for contraction (M' has 2 off vertices in G')

- *$M \subset M' \cup E(C)$*
- *the number of non-matched vertices of M in G equals the number of non-matched vertices of M' in G'*

Proof. Build M such that $M' \subseteq M$ and

- C in E' is matched by M'.
  Let $e_0 \in M'$ be incident with C (super-vertex in G') and let $v_0 \in V(G)$ such that $e_0 \in M'$ is built by contraction of corresponding vertices starting from v_0. Extend M as far as possible by cycle edges starting from v_0 and initially with the second edge.
- Let $c \in G'$ not be matched. Extend M as far as possible with cycle edges (arbitrary).

As exercise: Show that in both points it holds that the number of odd vertices of $M \in G$ equals the number of odd vertices of M' in G'. $\square$

Corollary. If Edmonds Blossom Algorithm terminates with a perfect matching (in a contracted graph), then G contains a perfect perfect matching. This perfect matching can be reconstructed as explained in Theorem 69 with an iterative approach.

Theorem 70. *Let G' be a graph constructed by iterative contraction of odd cycles as in Edmonds Blossom Algorithm. Let M' be a matching in G' and T be a M'-alternating tree in G, such that $\forall\, w \in A(T)$ is w a contracted vertex.*

It follows if T becomes atrophied (no edges left), then G has no perfect matching.

Proof. Assign one $S(v) \subseteq V(G)$ for every $v \in V(G')$ such that

$$S(v) = \begin{cases} \{v\} & v \text{ is not contracted} \\ \bigcup_{w \in C} S(w) & v \text{ is contracted, created by contraction of cycle } C \end{cases}$$

Try to find a set $X \subseteq V(G)$ which does not satisfy the Tutte-Berge condition: $q(G - x) \leq |X| \ \forall X \subseteq V(G)$.

$$X = A(T) \subseteq V(G)$$

$$q(G - X) \geq |B(T)|$$

$$\Rightarrow \ \forall v \in B(T), S(v) \text{ is a connected component of } G - X \text{ and}$$

$$|S(v)| = \left| \bigcup_{w \in C} S(w) \right| = \sum_{w \in C} |S(w)| \equiv 1 \mod 2$$

$$\text{if } |S(w)| \text{ are all odd (induction of number of iterations)}$$

$$\Rightarrow q(G - X) \geq |B(T)| > A(T)$$

Contradiction. □

Theorem 71. *Edmonds Blossom Algorithm terminates after $\mathcal{O}(n)$ matching augmentations, $\mathcal{O}(n^2)$ contractions and $\mathcal{O}(n^2)$ extensions of the tree. It decides correct whether a perfect matching exists.*

Proof. Correctness is given by Theorems 69 and 70.

The number of iterations is the number of matching augmentations $\leq \lfloor \frac{n}{2} \rfloor = \mathcal{O}(n)$, because the matching will never be shrunken. Count the number of contractions (equals number of tree extensions) between every two possible consecutive M augmentations.

Number of contractions lose at least 1 vertex per contraction

Number of tree augmentations Number of vertices outside of tree decrements by at least one vertex per tree extension.

It follows that there are $\mathcal{O}(n)$ contractions and $\mathcal{O}(n)$ tree extensions. □

Theorem 72. *Edmonds Blossom Algorithm can be implemented with runtime $\mathcal{O}(nm \log n)$.*

This lecture took place on 8th of January 2015: Welcome to 2015!.

18.1 Using Edmonds Blossom Algorithm to determine a matching with maximal cardinality

Given. Unweighted instance $G = (V, E)$

Find. Matching M^* in G with $|M^*| = \max_{\text{M is matching in G}} |M|$

Apply Edmonds Blossom Algorithm:

- Output is a perfect matching: $M^* \implies M^*$ is a solution to P
- Output claims "no perfect matching exists in G"

Let T_1 be an atrophied tree, which results in output of the second kind. Remove $T_1(E(T_1))$ from G (hence the edges of T_1), let resulting graph be G_1. Let M_1 be the matching evaluated by Edmonds Blossom Algorithm. Let $M_2 := M_1 \setminus E(T_1)$. Apply Edmonds Blossom Algorithm in G_1 with initial matching M_2 (probably $M_2 = \varnothing$). At termination either

- M_2 (transformed) is perfect matching in G_1
- there does not exist a perfect matching in G_1. Let T_2 be an atrophied tree regarding M_2 in G.

Repeat this procedure as long as either a perfect matching in the current graph can be found or no edges survive.

Let k be the number of repetitions (must be finite, because every repetition removed at least one edge).

Case distinction:

1. k repetitions terminate with atrophied tree. Let $T_1, T_2, \dots, T_k$ be atrophied trees and $M_1, M_2, \dots, M_k$ the matchings at termination at the corresponding iterations of Edmonds Blossom Algorithm.
2. k repetitions terminate with a perfect matching in G_{k-1}. Let $T_1, \dots, T_{k-1}$ be atrophied trees and $M_1, \dots, M_k$ matchings as in the first case.

Let $M := \bigcup_{i=1}^{k} M_i$. We show M is solution to the given problem. Removal from atrophied tree isolates all vertices in tree because if not all incidenting edges are in the tree, the tree is atrophied.

Definition. $\operatorname{def}(M) := |V(G)| - 2|M| = |\textit{unmatched vertices}|$ *(called* deficiency*).*

Let $B(T_i), A(T_i)$ be the set of even and odd vertices in T_i with $1 \leq i \leq k$ respectively.

$$X := \bigcup_{i=1}^{k(k-1)} A(T_i)$$

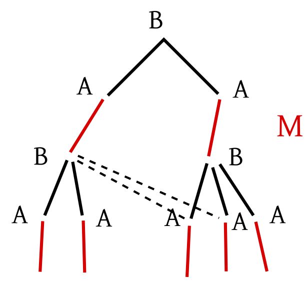


Figure 46: Basic structure when computing X

$$q_G(X) \geq \sum_{i=1}^{k} |B(T_i)|$$

because in every even vertex create a singleton in $G \setminus X$.

For the first case:

$$\sum_{i=1}^{k} (|A(T_i)| + 1) = k + \sum_{i=1}^{k} |A(T_i)| = k + |X|$$

$$q_G(X) - |X| \geq k = \operatorname{def}(M)$$

Analogously for the second case: It always holds that $q_G(X) - |X| \geq \operatorname{def}(M)$.

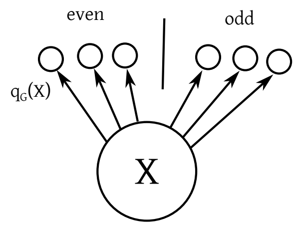
$$q_X(G) - |X| \leq \operatorname{def}(M)$$

holds for all matching M and for all $X \subseteq V(G)$ because At least $q_G(X) - |X|$ stay unmatched.

Hence matching satisfies $\operatorname{def}(M) = q_G(X) - |X|$. Hence M has minimal deficiency. Hence M has maximal cardinality. So a solution to our given problem is given.

Remark. Implicitly this is an algorithmical proof for Beige-Tutte.

$$\min_{M \text{ matching in } G} \operatorname{def}(M) = \max_{X \subseteq V(G)} (q_G(X) - |X|)$$



19 Weighted matching problems and complete unimodular matrices

19.1 Weighted matching problems

Max-Weight-Matching problem (Max-WMP)

Given. $G = (V, E)$ is unweighted, $c : E(G) \to \mathbb{R}$

Find. Determine a matching M^* such that $c(M^*) = \sum_{e \in M^*} c(e) = \max_{M \text{ matching in } G} c(M)$.

Min-Weight Perfect Matching Problem (MinWPMP)

Given. $G = (V, E)$ is unweighted, $c : E(G) \to \mathbb{R}$

Find. Determine a perfect matching M^* with $c(M^*) = \min M$ perfect matching in $Gc(M)$ or "no perfect matching exists in G".

Observation. MaxWMP and MinWPMP are equivalent.

Proof. First, we prove MaxWMP is polynomially reducible to MinWPMP. MaxWMP $\Rightarrow$ MinWPMP.

Let (G, c) be an instance of MaxWMP. Construct an instance (G', c') of Min-WPMP with G' created from G by introduction of $|V(G)|$. Additional vertices with full degree:

$$c'(e) = \begin{cases} -c(e) & e \in E(G) \\ 0 & e \in E(G') \setminus E(G) \end{cases}$$

Let M' be a perfect matching in G'. Let $M = M' \cap E(G)$.

$$c'(M') = -c(M)$$

$$\min_{M' \text{ is perfect matching in } G'} c'(M') = - \max_{M \text{ matching in } G} c(M)$$

because every matching in G in a perfect matching in G' can be completed.

Now we prove the other direction: MinWPMP is polynomially reducible to MaxWMP.

Let (G, c) be an instance of MinWPMP. Construct (G', c') instance of MaxWMP with $G' = G$ and $c'(e) = K - c(e)$ where $K = 1 + \sum_{e \in E(G)} |c(e)|$.

Let M' be a matching with maximal weight in (G', c').

Observation. M' is matching with maximal cardinality in G'.

So every $|M_1| > |M|$ and $c'(M_1) \leq c'(M')$.

$$K |M_1| - c(M_1) \leq K |M'| - c(M')$$

$$\Rightarrow K (|M_1| - |M'|) \leq c(M_1) - c(M')$$

$$K = 1 + \sum_{e \in E(G)} |c(e)| \leq c(M_1) - c(M')$$

This is a contradiction.

M' is the solution for the MinWPMP problem if it is a perfect matching, because $c'(M) = |K| \cdot |M| - c(M)$. Otherwise no perfect matching exists (see Observation). $\square$

19.2 The MinWPMP in bipartite graphs

This is an alternative description of the assignment problem.

Theorem 73. *The assignment problem can be solved with $\mathcal{O}(nm + n^2 \log n)$ runtime.*

Proof. Instance (G, c) with $G = (A \cup B, E)$ and $c : E(G) \to \mathbb{R}$. Transform it to minimum-cost flow problem.

$$\forall\, a \in A : (s, a) \in E(G')$$

$$\forall\, b \in B : (b, t) \in E(G')$$

$$V(G') = V(G) \cup \{s\} \cup \{t\}$$

$$u : E(G') \to \mathbb{R}_+$$

$$u(e) \equiv 1$$

$$b : V(G') \to \mathbb{R}$$

$$b(s) = n \qquad b(t) = -n$$

$$b(v) = 0 \; \forall\, v \in A \cup B$$

Weights are 0 for new edges. Weights are unmodified for the old ones.

We observe:

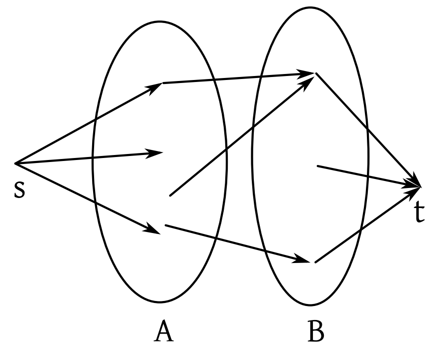


Figure 47: Structure of instance (G, c)

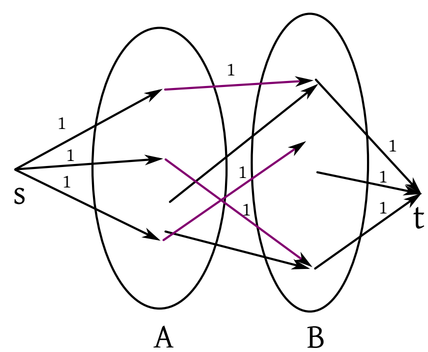


Figure 48: (G, c) with flow f_M

- For all perfect matchings M in G, there is a flow f_M.

$$f_M(e) = \begin{cases} 1 & e \text{ (with loss of direction)} \in M \\ 0 & e \text{ (with loss of direction)} \in E(G) \setminus M \\ 1 & e = (s, a) \vee e = (b, t) \end{cases}$$

$$c(f_M) = c(M)$$

- Every valid integer flow F corresponds to one perfect matching M_f with $c(M_f) = c(f)$ (with s as source, the flow must be saturated $\Rightarrow$ only valid flows are perfect matchings).

For the solution of MinWPMP in (G, c) solve the minimum-cost flow problem in (G', u, b, c') and determine an optimal integer solution with successive-shortest-path algorithm.

From both observations, we conclude the provided solution is an optimal solution for MinWPMP. The runtime is $\mathcal{O}(mn + B(m + n\log n))$ for successive-shortest-path where

$$B = \frac{1}{2}\sum_{v\in V(G')} |b(v)| = n$$

This is in general similar to the hungarian method by Harold Kuhn (1955). $\square$

19.3 Definition of the assignment problem as (MI)LP

(MI)LP = *mixed integer linear program.*

A *generic MILP* looks as follows:
Minimize $c^t x$ subject to $Ax \leq b$ with $x \geq 0$ and

$$x_i \in \mathbb{Z}_+ \quad i \in I \subset \{1, 2, \dots, n\}$$

A *generic LP-relaxation* looks as follows:
Minimize $c^t x$ subject to $Ax \leq b$ with $x > 0$.

Introduce some

$$\{x_e\}_{e\in E(G)} \in \{0,1\}^{|E(G)|}$$

encoding some edge set

$$M := \{e \in E(G) : x_e = 1\}$$

M shall be a perfect matching:

$$\sum_{e\in\delta(v)} x_e = \underbrace{1}_{\text{p.m.}} \ \forall\, v \in V(G)$$

where the perfect matching has constraint $c(M) = \sum_{e\in E(G)} c_e x_e$. Equality is given, because $\leq$ would only show *some* matching.

$$\min \sum_{e\in E(G)} c_e x_e$$

$$\sum_{e\in\delta(v)} x_e = 1 \quad \forall\, v \in V(G)$$

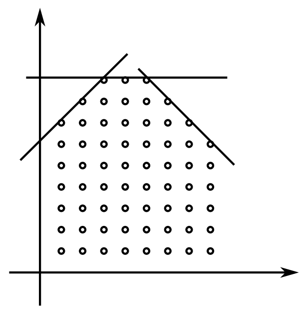


Figure 49: General linear programming - recognize the linear constraints

$$x_e \in \{0,1\} \quad \forall\, e \in E(G)$$

LP-relaxation: $\min \sum c_e x_e$

$$\sum_{e \in \delta v} x_e = 1 \quad \forall\, v \in V(G)$$
$$x_e \geq 0 \quad \forall\, e \in E(G)$$

Question: When is optimal solution of LP-relaxation also an optimal solution for MILP?

$$\text{valid-solution (MILP)} \subseteq \text{valid-solution(LP-relaxation)}$$
$$\min(\text{valid-solution (MILP)}) \geq \min(\text{valid-solution(LP-relaxation)})$$
$$(=?) \quad \text{what about equality?}$$

Sufficient A total unimodular $\Rightarrow$ LP-relaxation "equivalent" MILP.

This lecture took place on 13th of January 2014.

Definition. A *polyeder* $P(B) = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$ with $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$ is called *integral* (dt. ,,ganzahlig"), if all its corners are integral.

Remark. You can optimize linearly over integral polyeders in polynomial time ("Interior point method").

Question. Which sufficient conditions are provided by integral polyeders?

Definition. $A \in \mathbb{R}^{n \times n}$ is called *unimodular* if $\det(A) \in \{\pm 1\}$.

Observation.
$$\left.\begin{array}{c} A \in \mathbb{Z}^{n \times n} \\ A \text{ is unimodular} \end{array}\right\} \Rightarrow A^{-1} \in \mathbb{Z}^{n \times n}$$

where A^{-1} is unimodular. $\det(A^{-1}) = \frac{1}{\det(A)} \in \{\pm 1\}$.

$$A^{-1} = \frac{\text{adjacency}(A)}{\det(A)} = \frac{((-1)^{i+j}\det(A_{ji}))^t}{\det(A)}$$

$A \in \mathbb{R}^{m \times n}$ is called *total unimodular* (abbr. tum) if for every quadratic submatrix $B \in \mathbb{R}^{k \times k}$ ($k \leq \min m, n$) it holds that

$$\det(B) \in \{0, \pm 1\}$$

Considerations:

(1) $(a_{ij}) = A$ total unimodular $\Rightarrow a_{ij} \in \{0, \pm 1\}\ \forall i\, \forall j$

(2) A total unimodular $\Rightarrow A^t$ total unimodular

(3) A total unimodular $\Rightarrow -A$ total unimodular

(4) A total unimodular $\Rightarrow (A|A)$ total unimodular (where — denotes vertical concatenation)

(4, remark) Let B be a quadratic submatrix of $(A|A)$. 2 cases:

- if B contains two identical columns $\Rightarrow \det(B) = 0$
- if B contains no pair of identical columns $\Rightarrow B$ can be considered as submatrix of $A \Rightarrow \det(B) \in \{0, \pm 1\}$

(5) A total unimodular $\Rightarrow (A|E)$ is total unimodular where $E \in \mathbb{R}^{m \times m}$ is an identity matrix.

(5, remark) Let B be a quadratic total unimodular matrix of $A|E$:

- is submatrix of $A \Rightarrow \det(B) \in \{0, \pm 1\}$
- With appropriate rows and S_n permutations π.

$$k_1 + \underbrace{k_2}_{|\text{columns in } E|} = k$$

See figure 50 with

$$P_\pi = (B_j^{(\pi)}) = \begin{cases} 1 & \pi(i) = j\ \forall i, j \\ 0 & \text{else}\ \ \forall i, j \end{cases}$$

$$|\det B| = \left|\det \hat{B}\right|$$

$$\det \hat{B} = \det(A_1) \cdot \det(E_1) = \det(A_1)$$

$$\left(P^t = P_{\pi^{-1}}\right)$$

because A is a permuted submatrix of A.

$$
\begin{array}{c|c}
A & E \\
\hline
\left(\begin{matrix} \end{matrix}\right. & \left.\begin{matrix} 1 & & 0 \\ & \ddots 1 & \\ 0 & & \ddots 1 \end{matrix}\right)
\end{array}
$$

$$
P^{E}{}_{\pi} \cdot B \cdot P_{\pi} - \hat{B} \cdot \left(\begin{array}{c|c} A_1 & 0 \\ \hline A_2 & E_1 \end{array}\right)
$$

Figure 50: Matrix structure for consideration 5

6 A is total unimodular $\Rightarrow$ $(A| - A)$ is total unimodular (analogously to item 4)

Attention!

$$
\left.\begin{matrix} A \text{ tum} \\ B \text{ tum} \end{matrix}\right\} \not\Rightarrow (A|B) \text{ tum}
$$

As exercise, find a counterexample with $\det(C) \notin \{0, \pm 1\}$ but A and B are

$$
C = \left(\begin{array}{cc:c} & 1 & 0 \\ C & 0 & 1 \\ 1 & 0 & 1 \end{array}\right)
$$
$$
\underbrace{}_{A}\ \underbrace{}_{B}
$$

Figure 51: Counterexample structure

total unimodular.

Theorem 74. *(Hoffman & Kruskal, 1956) Let $A \in \mathbb{Z}^{m \times n}$. The following statements are equivalent:*

1. *A is total unimodular.*
2. *Polyeder $P(b) := \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$ is integral $\forall\, b \in \mathbb{Z}^m$*

3. *Every quadratic regular submatrix of A has an integral inverse*

Proof. (Arthur Veinott, 1968) We use the proof structure $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$

$1 \Rightarrow 2$ Assume A is total unimodular. Show that $P(b)$ is integral for $b \in \mathbb{Z}_n^m$ hence every base solution is integral. Let X_b be base solution with corresponding base A_B $m \times m$ submatrix of $(A|E)$ with $\det(A_B) \neq 0$. $X_b = A_B^{-1}b$ and $X_N = 0$. Because $A_B^{-1} \in \mathbb{Z}^{m\times m}$ (as inverse of the unimodular matrix A_B) and $b \in \mathbb{Z}^m$ it holds that

$$X_B \in \mathbb{Z}^m \Rightarrow (X_B, X_N) \in \mathbb{Z}^{n+m}$$

$2 \Rightarrow 3$ Assume $P(b)$ is integral for all $b \in \mathbb{Z}^m$. Show that for all quadratic submatrices $B \in \mathbb{Z}^{k\times k}$ of A, $\det(B) \neq 0 \Rightarrow B^{-1} \in \mathbb{Z}^{k\times k}$.

- $B \in \mathbb{Z}^{m\times m} \Rightarrow B$ is base of $(A|E) \Rightarrow X_B = A_B^{-1}b$, $X_N = 0$ is base solution (= corner) because $P(b)$ is integral: so $A_B^{-1}b$ must be integral.
  We show that A_B^{-1} is integral hence every column of A_B^{-1} is integral. Let $\overline{b_i}$ be the i-th column vector of A_B^{-1}. Let $t \in \mathbb{Z}^m$ such that $\overline{b_i} + t \geq 0$. Let $b(t) := A_B \cdot t + e$ with e_i as identity vector.

$$\left.\begin{array}{ll} \overline{X}_B & = A_B^{-1}b(t) = A_B^{-1}(A_Bt + e_i) = t + A_B^{-1} \cdot e_i = t + \overline{b}_i \geq 0 \\ \overline{X}_N & = 0 \end{array}\right\}$$

$$\Rightarrow (\overline{X}_B, \overline{X}_N) \text{ is valid base solution (corners) of } P(b(t))$$
$$\Rightarrow (\overline{X}_B, \overline{X}_N) \text{ is integral}$$
$$\Rightarrow t + \overline{b}_i = \overline{X}_B \text{ is integral}$$
$$\Rightarrow \overline{b}_i \text{ is integral}$$

  Because i is arbitrary, A_B^{-1} is integral.
- $B \in \mathbb{Z}^{k\times k}$ with $(k < m)$ (Figure 52). Complete columns of B with

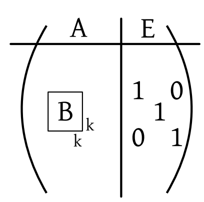


Figure 52: Base matrix

basis A_B of $(A|E)$ with column of E. With fitting permutation of

$$\left(\begin{array}{c|c} \widehat{B} & 0 \\ \hline \widehat{B}_1 & \begin{matrix} 1 & & 0 \\ & 1 & \\ 0 & & 1 \end{matrix} \end{array}\right)$$

Figure 53: Matrix structure

$$\left(\begin{array}{c|c} \widehat{B} & 0 \\ \hline \widehat{B}^1 & E_1 \end{array}\right)^{-1}$$

Figure 54: Matrix structure, part 2

$$\left(\begin{array}{c|c} \widehat{B}^{-1} & 0 \\ \hline -\widehat{B}_1 \cdot \widehat{B}^{-1} & E_1 \end{array}\right)$$

Figure 55: Matrix structure, part 3

rows and columns, A_B can be represented as (Figure 53) where $\hat{B}$ is created by column and row permutations of B. See Figure 54 is integral of bullet item 1. Figure 56 (equivalent to Figure 54) is integral $\Rightarrow \hat{B}^{-1}$ is integral.

$$\hat{B} = P_\pi^t B P_\pi \Rightarrow \hat{B}^{-1} = (P_\pi^t B P_\pi)^{-1} = P_\pi^{-1} B^{-1} P\pi$$
$$B^{-1} = P_\pi B^{-1} p_\pi^t \text{ is integral}$$

$3 \Rightarrow 1$ Let B be a quadratic submatrix of A. If $\det(B) = 0$, we are done. Else $\det(B) \neq 0$ and B^{-1} is integral, then $\det(B^{-1}) \in \mathbb{Z}$. $\det(B^{-1}) = \frac{1}{\det(B)}$ is integral $\Rightarrow \det(B) \in \{\pm 1\}$.

□

Corollary. Let $A \in \mathbb{Z}^{m \times n}$ be total unimodular. Then it holds $\forall\, b \in \mathbb{Z}^m\ \forall\, c \in \mathbb{Z}^n$ that

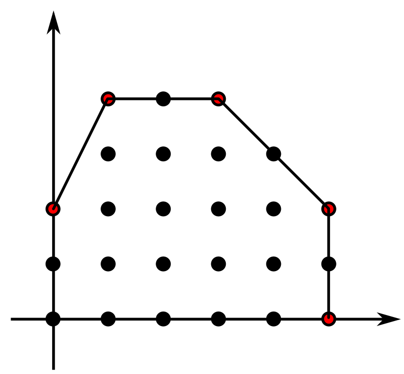


Figure 56: Linear programming problem

$$\max\left\{c^t x : Ax \leq b, x \geq 0, x \in \mathbb{Z}\right\} = \min\left\{b^t y : A^t y \leq c, y \geq 0, y \in \mathbb{Z}^m\right\}$$

This is equivalent (because of Theorem 74)

$$\max\left\{c^t x : Ax \leq b, x \geq 0\right\} = \min\left\{b^t y : A^t y \leq c, y \geq 0\right\}$$

This is the *duality of linear programming.*

Theorem 75. *(Heller & Tompkins, 1959) Let $A \in \{0, \pm 1\}^{m \times n}$ with at most two non-zero eintries per column. A is total unimodular if there exists a partition (R, T) of the rows in A ($R \cup T = \{1, 2, \dots, m\}$) such that*

- *if column j contains two ± 1 entries, then the corresponding rows belong to different parts of the partition.*
- *if column j contains one $+1$ and one -1 entry, then the corresponding rows belong to the same part of the partition.*

Proof. Let B be a quadratic submatrix of A with $B = (b_{ij})$. Show that $\det(B) \in \{0, \pm 1\}$.

Induction over number of rows of B. $B \in \{0, \pm 1\}^{k \times k}$.

Induction base $k = 1$ is fine.

Induction step Assume that the hypothesis holds for $B \in \{0, \pm 1\}^{(k-1)\times(k-1)}$
We want to show that it holds for $B \in \{0, \pm 1\}^{k \times k}$.

- If B contains zero-column, then $\det(B) = 0$
- If column j has exactly one non-zero entry, the structure of Figure 57 occurs. Compute $\det(B)$ with development along column j and apply

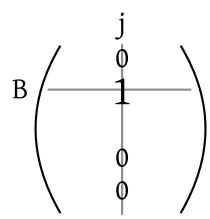


Figure 57: Matrix structure for induction step, case 2

induction assumption.

- Every column has two non-zero entries. Let j be an arbitrary column index.
$$\sum_{i \in R} b_{ij} = \sum_{i \in T} b_{ij} \ \forall\, j$$
It follows that rows of B are linearly dependent. Hence $\det(B) = 0$.

$\square$

Remarks:

- Determination of total unimodular matrices is solvable in polynomial time with $\mathcal{O}((n+m)^4 \min(m, n))$ algorithm of Seymour.
- Conditions by Heller and Tompkins are also required for $A \in \{0, \pm 1\}^{m \times n}$ with at most two non-zero entries per column.

19.3.1 Examples

- Vertex-edge incidence matrices of digraphs are total unimodular according to Theorem 75. Let G be digraph $(a_{ve}) : A \in \{0, \pm 1\}^{|V(G)| \times |E(G)|}$.

$$a_{ve} = \begin{cases} 0 & v \cap e = \varnothing \\ 1 & e = (v, ?) \\ -1 & e = (?, v) \end{cases}$$

Partition is $R = V(G)$ and $T = \varnothing$.

- Vertex-edge incidence matrices of complete bipartite graphs:

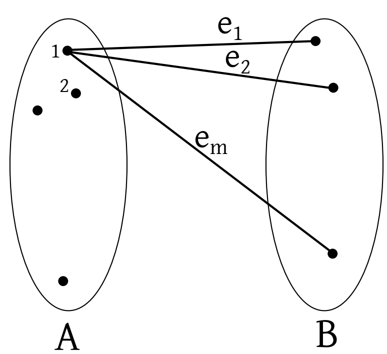


Figure 58: Example edges and vertices

$$|A| = n \qquad B = m \quad (|A| + |B|) \times |E(G)|$$
$$(a_{v,e}) = A = \{0, +1\}$$
$$a_{v,e} = \begin{cases} 1 & e \cap v \neq \{\} \\ 0 & \text{else} \end{cases}$$

This lecture took place on 19th of January 2015.

Theorem 76. *(Corollary by Hoffman and Kruskal) Let A be total unimodular with $A \in \{0, \pm 1\}^{m \times n}$.*

1. *Then it holds that*

$$\forall\, c \in \mathbb{Z}^n,\ \forall\, b \in \mathbb{Z}^m : \begin{array}{ll} P_p & = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\} \\ P_d & = \{y \in \mathbb{R}^m : A^t y \geq c, y \geq 0\} \end{array}$$

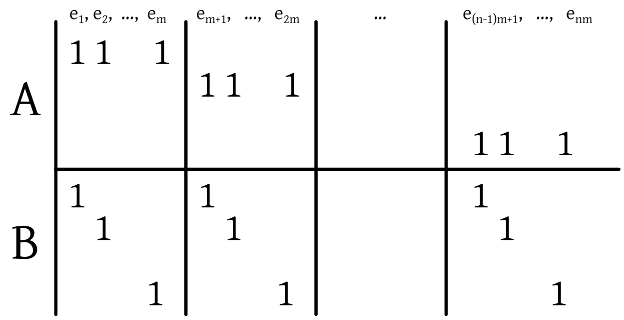


Figure 59: Example edges and vertices matrix

P_p and P_d are integral.

2. *Polyeder* $S = \left\{x \in \mathbb{R}^n : \underline{b} \leq Ax \leq \overline{b}, 0 \leq x \leq d\right\}$ *is integral if* $\underline{b}, \overline{b} \in \mathbb{Z}^m$ *and* $d \in \mathbb{Z}^n_+$.

Proof. Define restrictions:

$$Ax \leq \overline{b}$$

$$-Ax \leq -\underline{b}$$

Identity matrix $n \times n$:

$$Ix \leq d$$

$$S = \left\{ x \in \mathbb{R}^n : \begin{pmatrix} A \\ -A \\ I \end{pmatrix} x \leq \begin{pmatrix} b \\ -b \\ d \end{pmatrix}, x \geq 0 \right\}$$

$$\left.\begin{array}{ll} A \text{ total unimodular} & \Rightarrow A| - A \text{ t.um.} \\ A \text{ total unimodular} & \Rightarrow A|I \text{ t.um.} \end{array}\right\} \text{also holds of transposed matrices}$$

□

Definition. A matrix $X = (x_{ij}) \in \mathbb{R}^{n \times n}$ is called *double-stochastic* if

- $\sum_{j=1}^n x_{ij} = 1 \;\forall\, i \in \{1, \ldots, n\}$
- $\sum_{i=1}^k x_{ij} = 1 \;\forall\, j \in \{1, \ldots, n\}$
- $x_{ij} \geq 1 \;\forall\, i, j \in \{1, \ldots, n\} \times \{1, \ldots, n\}$

Definition. The polyeder (polytop)

$$P_A := \left\{ (x_{ij}) \in \mathbb{R}^{n \times n} : \sum_i x_{ij} = 1, \sum_j x_{ij} = 1, x_{ij} \geq 0 \right\}$$

$$\begin{array}{r} n\left\{\vphantom{\begin{matrix}1\\1\\1\\1\end{matrix}}\right. \\ n\left\{\vphantom{\begin{matrix}1\\1\\1\\1\end{matrix}}\right. \end{array} \underbrace{\left(\begin{array}{c|c|c|c|c} 1111 & & & & \\ & 1111 & & & \\ & & \cdots & & \\ & & & 1111 & \\ \hline \begin{matrix}1&&&0\\&1&&\\&&1&\\0&&&1\end{matrix} & \begin{matrix}1&&&0\\&1&&\\&&1&\\0&&&1\end{matrix} & & & \end{array}\right)}_{n^2}$$

is called *alignment polyeder* (dt. ,,Zuordnungspolyeder", set of all double-stochastic matrices). Matrix of restrictions of P_A is total unimodular (Heller and Tompkins $\Rightarrow P_A$ is integral).

$$\Leftrightarrow \text{ corners of } P_A \text{ are integral}$$

$$\Leftrightarrow \text{ corners of } P_A \text{ are permutation matrices}$$

$$\forall\, x \in P_A : \exists \text{ vector } (\alpha_e)_{e \in \text{corner}(P_A)} \text{ such that } x = \sum_{e \in \text{corner}(P_A)} \alpha_e \underbrace{Y_e}_{\text{corner}}$$

...which is the convex comination of the corners.

$$\sum_{e \in \text{corner}(P_A)} \alpha_e = 1 \qquad \alpha_e \geq 0 \;\forall\, e \in \text{corner}(P_A)$$

Proof. We now what to prove corners of $P_A \Leftrightarrow P_A$ are permutation matrices.

$\Rightarrow$ trivial

$\Leftarrow$ Let $X^{\pi_0} = (x_{ij}^{\pi_0})$ the permutation matrix for the permutation π_0. Hence,

$$x_{ij}^{\pi_0} = \begin{cases} 1 & \pi_0(i) = j \\ 0 & \text{else} \end{cases} \quad \forall\, i, j \in \{1, \dots, n\}$$

X^{π_0} is double-stochastic $\Rightarrow X^{\pi_0} \in P_A \Rightarrow X^{\pi_0}$ is convex combination of the corners of P_A.

$$\Rightarrow \exists \alpha_\pi \geq 0 \;\forall\, \pi \in S_n \text{ with } \sum \alpha_\pi = 1, \quad X^{\pi_0} = \sum_{\pi \in S_n} \alpha_\pi X^\pi$$

where $\pi \in S_n$ is (by convention) the set of permutations over $\{1, \dots, n\}$.

Goal. Show that $\alpha_{\pi_0} = 1$ (all others are in $\alpha = 0$) $\Rightarrow \pi_0$ is a corner.

$$x_{ij}^{\pi_0} = \sum_{\pi \in S_n} \alpha_\pi x_{ij}^\pi$$

Let $j \neq \pi_0(i)$ then it holds that

$$0 = \sum_{\pi \in S_n} \alpha_\pi x_{ij} \Rightarrow \alpha_\pi x_{ij}^\pi = 0 \, \forall \, \pi, \; \forall \, i, j \Rightarrow \alpha_\pi = 0$$

If $\pi(i) = j$, then

$$x_{ij}^\pi = 1 \Rightarrow \alpha_\pi = 0$$

$\forall \, i, j \in \{1, \ldots, n\}$ it holds that

$$\left.\begin{array}{c} \pi_0(i) \neq j \\ \pi(i) = j \end{array}\right\} \Rightarrow \alpha_\pi = 0$$

Followingly $\forall \, \pi \neq \pi_0$, it holds that $\alpha_\pi = 0$ because $\exists i \in \{1, 2, \ldots, n\}$ with $\pi_0(i) \neq \pi(i)$.

Let $j = \pi(i)$. Then $j \neq \pi_0(i)$. Apply

$$\left.\begin{array}{c} \pi_0(i) \neq j \\ \pi(i) = j \end{array}\right\} \Rightarrow \alpha_\pi = 0$$

with these i and j. $\Rightarrow \alpha_{\pi_0} = 1$. π_0 muss be contained in sum $\Rightarrow \pi_0$ must be corner.

□

Theorem 77. *(Theorem by Birckhoff) The permutation matrices correspond to the corners of an assignment polytop and every double-stochastic matrix can be represented as convex combination of permutation matrices.*

matching number, vertex cover number, edge cover number, edge incidence number, stability number

matching number

$\vartheta(G) := \max \{|M| : M \text{ matching in } G\}$

vertex/node cover number

$\gamma(G) := \min \{|Y| : Y \subseteq E(G), Y \text{ is vertex cover in } G\}$

edge cover number

$\zeta(G) := \min\{|Y| : Y \subseteq V(G), Y \text{ is vertex cover in } G,$
$\forall \, e \in E(G) \, \exists v \in Y : e \cap V \neq \varnothing\}$

stability number

$\alpha(G) := \max \{|S| : S \text{ is stable in } G\}$

Definition. $S \subseteq V(G)$ is called a *stable set* or *independent set* if $\forall \, u, v \in S : (u, v) \neq E(G)$.

Let A be total unimodular with $A \in \mathbb{R}^{m \times n}$. Consider

$$\begin{aligned} P_1 &:= \max\left\{1^t x : Ax \leq 1, x \in \mathbb{Z}_+^n\right\} = \min\left\{1^t y : A^t y \geq 1, y \in \mathbb{Z}_+^m\right\} =: D_1 \\ P_2 &:= \min\left\{1^t x : Ax \geq 1, x \in \mathbb{Z}_+^n\right\} = \max\left\{1^t y : A^t y \leq 1, y \in \mathbb{Z}_+^m\right\} =: D_2 \end{aligned}$$

... where 1^t represents a vector $(1, 1, \dots)$.

Let G be a bipartite graph and A be a vertex incidence matrix of G.

$$A \in \{0,1\}^{|V(G)| \times |E(G)|}$$

$$\begin{aligned} &\max\left\{\sum_{e \in E(G)} x_e : \sum_{e \in E(G)} a_{ve} x_e \leq 1, \underbrace{x_e \in \mathbb{Z}_+}_{x_e \in \{0,1\}} \underbrace{\forall\, e \in E(G)}_{\forall\, v \in V(G)}\right\} \\ = &\min\left\{\sum_{v \in V(G)} y_v : \sum_{v \in V(G)} a_{ve} y_v \geq 1, \underbrace{y_v \in \mathbb{Z}_+}_{y_v \in \{0,1\}}, \underbrace{\forall\, v \in V(G)}_{\forall\, e \in E(G)}\right\} \end{aligned}$$

x_e and y_v will be 1 or 0, because we are minimizing.

Construct $\forall\, x_e$ matching M with $e \in M \Leftrightarrow x_e = 1$. P_1 corresponds to max-cardinality matching problem.

Observation. The optimal solutions y^* of D_1 satisfy $y_v^* \in \{0,1\}\ \forall\, v \in V(G)$.

For D_1, $\forall\, e \in E(G)$ with $e = (v_1, v_2)$:

$$\sum_{v \in V(G)} a_{ve} y_v \geq 1 \Leftrightarrow a_{v_1 e} y_{v_1} + a_{v_2 e} y_{v_2} = y_{v_1} + y_{v_2} \geq 1$$

So every valid solution of $\overline{D_1} = \{\dots y_v \in \{0,1\}\ \forall\, v \in V(G) \dots\}$ corresponds to an edge cover.

Optimal value of $P_1 = \vartheta(G)$ is the optimal value of $D_1(\overline{D_1}) = \zeta(G)$.

$$\overline{P_2} := \min\left\{1^t x : Ax \geq 1, x \in \{0,1\}^n\right\} = \min\left\{1^t x : Ax \geq 1, x \in \mathbb{Z}_+^n\right\}$$

as in $D_1 = \overline{D_1}$ where $n = |E(G)|$.

$$\forall\, v \in V(G) : \sum_{\forall\, e \in E(G)} a_{ve} x_e \geq 1$$

$$\forall\, v \in V(G) : \sum_{e \in \delta(v)} a_{ve} x_e \geq 1$$

so at last 1 edge per vertex is incident with v.

Valid solutions of $\overline{P_2}$ are vertex covers.

Target function value corresponds to the cardinality of the vertex cover.

Optimal target function value of $\overline{P_2} = \delta(G) =$ cardinality of small vertex cover.

$$D_2 : \max \left\{ 1^t y : A^t y \leq 1, y \underbrace{\in \mathbb{Z}_+^n}_{\in\{0,1\}^n} \right\}$$

where $n = |V(G)|$

$$\forall_{e=(v_1,v_2)} e \in E(G) \sum_{v \in V(G)} a_{ve} y_v \leq 1$$

$$\leq a_{v_1 e} y_{v_1} + a_{v_2 e} y_{v_2} = y_{v_1} + y_{v_2} \leq 1$$

Every valid solution of D_2 ($\overline{D_2}$) corresponds to one stable set and the target function value corresponds to its cardinality.

$$\mathrm{opt}(D_2) = \alpha(G) = \zeta(G) = \mathrm{opt}(P_2)$$

20 Matroids

A family of sets $(E, \mathcal{F})$ consists of a finite set E and a family $\mathcal{F}$ of subsets of E. $(E, \mathcal{F})$ is called independence system (IDS, dt. UAS) if

M_1 $\varnothing \in \mathcal{F}$

M_2 $Y \in \mathcal{F}, X \subseteq Y \Rightarrow X \in \mathcal{F}$

The elements of $\mathcal{F}$ are called independent, $A \subseteq E$, $A \notin \mathcal{F}$ are called *dependent*.

Inclusion minimal dependent sets are called *cycles*. Inclusion maximal dependent sets are called *bases*.

$\forall X \subseteq E$ we define a rank of X:

$$\mathrm{rank}(X) := \max \{|Y| : Y \subseteq X, Y \in \mathcal{F}\}$$

$\Rightarrow$ only independent subsets of X.

$\forall X \subseteq E$ define a closure of X, $\sigma(X)$ with

$$\sigma(X) := \{y \in E : \mathrm{rank}(X \cup \{y\}) = \mathrm{rank}(X)\}$$

Example. Let $G = (V, E)$ be a graph. $E = V(G)$.

$$\mathcal{F} := \{F \subset V(G) : F \text{ is stable}\}$$

This lecture took place on 20th of January 2015.

20.1 Maximization problem for IDS

Given. IDS $(E, \mathcal{F})$ and $c : E \to \mathbb{R}$

Find. Determine $F^* \in \mathcal{F}$ with $c(F^*) = \max_{F \in \mathcal{F}}(F)$

$\mathcal{F}$ is specified by a (polynomial) IDS-oracle, which is an algorithm solving the question "Is a given $F \subseteq E$ independent?".

20.2 Minimization problem for IDS

Given. IDS $(E, \mathcal{F})$ and $c : E \to \mathbb{R}$

Find. Determine base B^* of IDS with $c(B^*) = \min_{\text{base } B \in (E,\mathcal{F})} c(B)$

This problem is NP-hard.

The (polynomial) base-superset-oracle is used to solve the question "For $F \subseteq E$, is there some $B \subseteq F$ the base in $(E, \mathcal{F})$?".

21 Examples

21.1 Max-Weight Stable Set Problem

Given. $G = (V, E)$ graph

Find. Determine a stable set $S \subseteq V(G)$ with maximal $|S|$

This problem is NP-hard.

$$E = V(G) \qquad \mathcal{F} = \{F \subseteq E = V(G) : F \text{ is stable}\} \qquad c \equiv 1$$

21.2 Travelling Salesman Problem

Given. $G = (V, E)$ graph, $c : E(G) \to \mathbb{R}_+$

Find. Hamiltonian cycle with minimal weight

$$E = E(G) \qquad \mathcal{F} = \{F \subseteq E(G) : F \text{ is subset of hamiltonian cycle}\}$$

c as above.

Hamiltonian cycles are bases of $(E, \mathcal{F})$. NP-hard problem.

21.3 Shortest-Path Problem

Given. Digraph $G = (V, E), c : E(G) \to \mathcal{R}, \{s, t\} \in V(G)^2$

Find. Determine a shortest s-t-path

$$E = E(G) \qquad \mathcal{F} := \{F \subseteq E : F \text{ is subset of } s\text{-}t\text{-path}\}$$

Can be solved in polynomial time.

21.4 Knapsack Problem

Given. $n \in \mathbb{N}, c_i, w_i \geq 0, \forall i \in \{1, 2, \ldots, n\}, W \geq 0$

Find. Determine $T \subset \{1, 2, \ldots, n\}$ with $\sum_{i \in T} w_i \leq W$, $\sum_{i \in T} c_i \to \max$

Type. NP-hard problem.

$$E = \{1, 2, \ldots, n\} \qquad \mathcal{F} := \left\{ F \subseteq E : \sum_{i \in F} w_i \leq W \right\}$$

c as above.

21.5 Minimum Spanning Tree Problem

Given. $G = (V, E)$ is valid graph $c : E(G) \to \mathbb{R}$, graph is connected

Find. Spanning tree T with minimal weight

Type. Minimization problem. Polynomial time solvable.

$$E = E(G) \qquad \mathcal{F} := \{F \subset E(G) : F \text{ is forest}\}$$

The bases of $(E, \mathcal{F}) \equiv$ spanning trees.

21.6 Maximum Weighted Forest Problem

Given. $G = (V, E)$, $c : E(G) \to \mathbb{R}$

Find. Forest F in G with maximum weight $c(F) = \sum_{e \in E(F)} c(e)$

Type. Maximization problem. Polynomial time solution.

c as above.

$$E = E(G) \qquad \mathcal{F} := \{F \subseteq E : F \text{ is edge set of a forest}\}$$

21.7 Maximum Weight Matching Problem

Given. $G = (V, E)$, $c : E(G) \to \mathbb{R}$

Find. Matching M with maximum weight $c(M) = \sum_{e \in M} c(e)$

$$E = E(G) \quad \mathcal{F} := \{F \subseteq E : F \text{ is matching}\}$$

c as above.

Definition. A IDS is a matroid if

$$\forall x, y \in \mathcal{F} : |X| > |Y| \Rightarrow \exists x \in \mathcal{X} \setminus \mathcal{Y} \text{ such that } Y \cup \{x\} \in \mathcal{F}$$

as condition M3.

Theorem 78. *The following IDS are matroids*

1. *E is set of column vectors of a matrix A over an arbitrary field K.*

   $\mathcal{F} := \{F \subseteq E : \textit{vectors of } F \textit{ are linearly independent in } K\}$ *"vector matroid"*

   $$Y = \{col_1, col_2, \ldots col_k\} \ \forall \in \mathcal{F}$$

   $$X = \left\{ \underbrace{\overline{col_1}, \overline{col_2}, \ldots, \overline{col_l}}_{\textit{linear indep.}} \right\} \in \mathcal{F} \qquad l > k$$

   Consider $X \cup Y$: $\operatorname{rank}(X \cup Y) \geq l$ *and* $\operatorname{rank}(Y) = k < \operatorname{rank}(X \cup Y)$*. Then it follows that*

   $$\exists \textit{vector } v \in X \cup Y \textit{ with } Y \cup \{u\} \textit{ linearly independent } v \in X \setminus Y$$

2. *IDS of exercise* 6*. "*Graphical matroids*". X, Y forests in $G : |X| > |Y|$ with (M3) condition. Show that $\exists x \in \mathcal{X} : Y \cup \{x\}$ is forest.*

   Assumption: $\forall x \in X : Y \cup \{x\}$ is not a forest $\Leftrightarrow$ x is in a connected component of Y $\forall x \in X$.

   $\Rightarrow$ every connected component of forest X is a subset of a connected component of forest Y.

   For any $G = (V, E)$ if G is cycle-free it holds that

   $$|\textit{connected components}| = |V(G)| - |E(G)|$$

   $$p := |\textit{connected components of } X|$$
   $$q := |\textit{connected components of } Y|$$

   $$p \geq q$$
   $$p = |V(G)| - |X| \geq |V(G)| - |Y|$$

   As far as $|X| \leq |Y|$, this is a contradiction.

> **Tree** *number of connected components* $= n - (n-1)$.
>
> **Forest** *number of connected components* $= |V(G)| - |E(G)|$ *if* G *is cycle-free.*

3. "Uniform matroid".

$$E = \{e_1, \ldots, e_n\} \quad \mathcal{F} := \{F \subseteq E : |F| \leq k\}$$

with $k \in \mathbb{N}$. $(M3)$ *is trivial to show.*

4. $G = (V, E)$ *is graph.* $S \subseteq V(G)$ *stable.* $\forall s \in S : k_s \in \mathbb{N}$.

$$E = E(G) \quad \mathcal{F} := \{F \subseteq E(G) : \delta_F(s) \leq k_s \; \forall \, s \in S\}$$

$$F = \{(1,2), (1,3), (4,5), \cancel{(4,2)}\}$$
$$F = \{(1,2), (1,3), (4,5), (4,3)\}$$

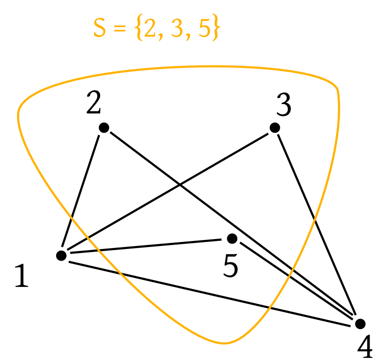


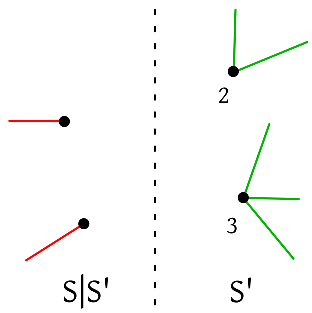
Figure 60: Example for Theorem 78 bullet point 4. $k_2 = 1, k_3 = 2, k_5 = 1$

See figure 60.
(M3) $X, Y \in \mathcal{F} : |X| > |Y|$.

$$S' = \{s \in S : \delta_Y(s) = k_s\}$$

$|X| > |Y|$ *and* $\delta_X(s) \leq k_s \; \forall \, s \in S$

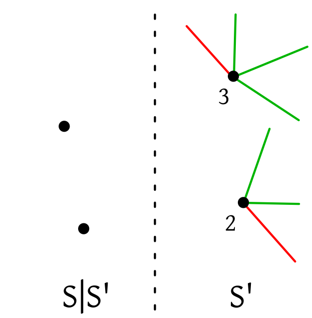
$$\xLongrightarrow{\text{to show}} \exists e \in S \setminus Y e \notin \delta(s) \; \forall \, s \in S'$$



If such an edge exists, we can append it.

$$\Rightarrow Y \cup \{e\} \in \mathcal{F}$$

Assumption: $\xRightarrow{\text{to show}}$ *does* not *hold:* $\forall\, e \in X \setminus Y : \exists s \in S' : e \in \delta(s)$



$$\Rightarrow |X| = \sum_{s \in S'} \delta_X(s) \leq \sum_{s \in S'} ks = \sum_{s \in S'} \delta y(s) = |Y|$$

$$|X| \leq |Y|$$

Contradiction to the assumption.

5. *Let* $G = (V, E)$ *be a digraph.* $S \subseteq V(E)$. $k_s \in \mathbb{N}\, \forall\, s \in S$. $E = E(G)$.

$$\mathcal{F} := \left\{ F \subseteq E : \delta_k^-(s) \leq k_s \right\}$$

(M3) analogous as in the previous item #4, but replace δ *with* δ^-. *Stability is relevant for the rational in item #4, but because a direction is given here, it is not required.*

Theorem 79. *Let* $(E, \mathcal{F})$ *be a IDS. Then the following statements are equivalent:*

M3: *Let* $X, Y \in \mathcal{F}, |X| > |Y| \Rightarrow \exists x \in X \setminus Y \quad Y \cup \{x\} \in \mathcal{F}$

M3': *Let* $X, Y \in \mathcal{F}, |X| = |Y| + 1 \Rightarrow \exists x \in X \setminus Y \quad Y \cup \{x\} \in \mathcal{F}$

M3": *For every* $X \subseteq E$ *the bases of* X *have the same cardinality.*

Proof. We show M3 $\Leftrightarrow$ M3', M3 $\Rightarrow$ M3" and M3' $\Rightarrow$ M3.

M3 $\Leftrightarrow$ M3' trivial.

M3 $\Rightarrow$ M3" Let B_1, B_2 be bases of $X \Rightarrow$

- $B_1, B_2 \in \mathcal{F}$
- $B_1, B_2 \subseteq X$
- B_1 and B_2 are inclusion maximal independent

Assumption. $|B_1| > |B_2|$. From M3, it follows that $\exists b \in B_1 \setminus B_2$: $B_2 \cup \{b\} \in \mathcal{F}$. So it follows that B_2 is not a tree (contradiction).

M3' $\Rightarrow$ M3 Let $X, Y \in \mathcal{F} : |X| > |Y|$. Show that $\exists x \in X \setminus Y : Y \cup \{x\} \in \mathcal{F}$.

Consider $X \cup Y$. Is Y a base of $X \cup Y$? No. Assume that Y is base of $X \cup Y \xRightarrow{(M3'')}$ all bases of $X \cup Y$ have cardinality $= |Y|$.

Consider $X \subseteq X \cup Y, X \in \mathcal{F}$. X is not inclusion maximal because otherwise it would be a base and hence $|X| = |Y|$ which is a contradiction.

$$\Rightarrow \exists \overline{X} \supsetneq X \text{ with } \overline{X} \in \mathcal{F}, \overline{X} \subseteq X \cup Y$$

$\overline{X}$ is analogously not a base. Can (again) be extended until a contradiction occurs, because E is finite.

Y is not a base

$\Rightarrow$ Y is not inclusion maximal independent in $X \cup Y$
$\Rightarrow$ Y is independent extensible in $X \cup Y$
$\Rightarrow$ $\exists e \in (X \cup Y) \setminus Y = X \setminus Y \quad Y \cup \{e\} \in \mathcal{F}$.

Let $X = e$.

$\square$

Definition. Let $(E, \mathcal{F})$ be a IDS. $\forall\, X \subseteq E$ the *lower rank* is defined as $\rho(X)$ where

$$\begin{aligned}\rho(X) &:= \min\{|Y| : Y \subseteq X, Y \in \mathcal{F}, Y \cup \{x\} \notin \mathcal{F}\ \forall\, x \in X \setminus Y\} \\ &:= \min\{|B| : B \text{ is base of } X\}\end{aligned}$$

The *rank* of X (rank(X)) is defined as $\max\{|Y| : Y \subseteq X, Y \in \mathcal{F}\}$.

The lower rank function $f : 2^E \to \mathbb{Z}_+$ with $f : x \mapsto \rho(x)$.

The rank quotient of $(E, \mathcal{F})$ is defined as $q(E, \mathcal{F}) = \min_{X \subseteq E} \frac{\rho(x)}{r(x)}$.

Example.

$$G = (V, E), E = E(G), \mathcal{F} = \{M \subseteq E : M \text{ matching}\}$$

$$\rho(E) = |\text{smallest inclusion maximal matching}|$$

$$r(E) = |\text{largest matching}|$$

$$\rho(E) \leq r(E)$$

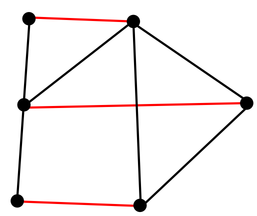


Figure 61: Rank example

This lecture took place on 26th of January 2015.

Theorem 80. *Let $(E, \mathcal{F})$ be an IDS. Then it holds that $q(E, \mathcal{F}) \leq 1$. Furthermore iff $q(E, \mathcal{F}) = 1$ then $(E, \mathcal{F})$ is a matroid.*

Proof.

$$q(E, \mathcal{F}) = \min_{A \subseteq E} \frac{\rho(A)}{r(A)} \leq 1$$

because $\rho(A) \leq r(A) \, \forall \, A \subseteq E$.

$$q(E, \mathcal{F}) = 1 \Leftrightarrow \frac{\rho(A)}{r(A)} = 1 \, \forall \, A \subseteq E$$

$$\Leftrightarrow \rho(A) = r(A) \, \forall \, A \subseteq E \Leftrightarrow M3''$$

all bases of X have the same cardinality. □

Theorem 81. *(Hausmann, Jenkyns, Korte, 1980) Let $(E, \mathcal{F})$ be an IDS. If $\forall A \in \mathcal{F}\, \forall e \in E$, $A \cup \{e\}$ contains at most ρ cycles, then it holds that*

$$q(E, \mathcal{F}) \geq \frac{1}{\rho}$$

A proof for Theorem 81 is not provided.

21.8 Additional matroid axioms

Theorem 82. *(bases) Let E be a finite set and $\mathcal{B} \subseteq 2^E$. Family $\mathcal{B}$ is the set of bases of a matroid if and only if the following base axioms are satisfied*

(B1) $B \neq \varnothing$

(B2) $\forall B_1, B_2 \in \mathcal{B}$ *and* $x \in B_1 \setminus B_2 : \exists y \in B_2 \setminus B_1$ *with* $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$.

If (B_1) satisfies (B_2), then $(E, \mathcal{F})$ is the matroid with base set $\mathcal{B}$ where

$$\mathcal{F} = \{F \subseteq E : \exists B \in \mathcal{B} \text{ with } F \subseteq B\}$$

Example. Let $G = (V, E)$ and is connected. Then the bases are its spanning trees. The graphical matroid $M(G)$ is given.

(B1)

(B2) Let T_1 and T_2 be spanning trees. $x \in T_1 \setminus T_2$. $\{v\} \cup (T_1 \setminus \{x\})$ is a spanning tree again.

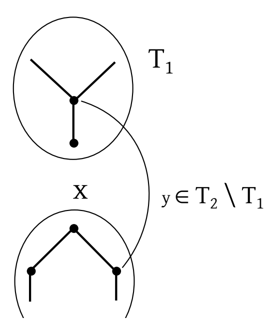


Figure 62: Example. y connects both connected components

Theorem 83. *Let E be a finite set and $r : 2^E \to \mathbb{Z}_+$. Then the following 3 statements are equivalent:*

- *r is the rank function of a matroid $(E, \mathcal{F})$ (with $\mathcal{F} = \{F \subseteq E : r(F) = |F|\}$).*
- *$\forall X, Y \subseteq E$ it holds that*

  (R1) $r(X) \leq |X|$

  (R2) $X \subseteq Y \Rightarrow r(X) \leq r(Y)$

  (R3) $r(X \cup Y) + r(X \cap Y) \leq r(X) + r(Y)$ *(submodular)*
- *$\forall X \subseteq E$ and $x, y \in E$ it holds that*

  (R1') $r(\varnothing) = 0$

  (R2') $r(X) \leq r(X \cup \{y\}) \leq r(X) + 1$

  (R3') $r(X \cup \{x\}) = r(X \cup \{y\}) = r(X) \Rightarrow r(X \cup \{x, y\}) = r(X)$

Example. $M(G)$. G is connected.

(R1) $X \subseteq E(G)$. $r(X) = \sum_{i=1}^{q}(n_i - 1)$ where q is the number of connected components of $(V(G),\ X)$ and n_i is the number of vertices in the i-th connected component of $(V(G), X)$.

$$|X| \geq r(X) \qquad \text{where} \qquad |X| = \sum_{i=1}^{q} h_i$$

(R2) $X \subseteq Y \Rightarrow r(X) \leq r(Y)$.

$F \subseteq X_i$ is cycle-free in $(V(G), X)$ implies F is cycle-free in $(V(G), Y)$.

(R1') done

(R2') $r(X + \{x\}) \leq r(X) + 1$

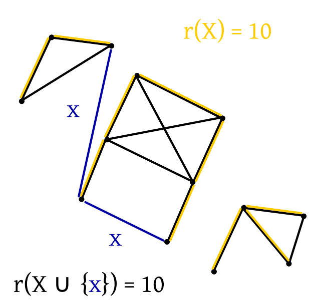


Figure 63: Example for (R2')

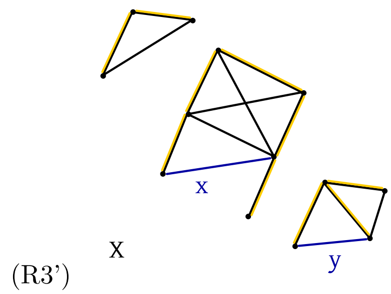


Figure 64: Example for (R3')

Theorem 84. *(Closure) Let E be a finite set with $r : 2^E \to 2^E$. σ is the closure function of a matroid if $\forall X, Y \subseteq E$ and $\forall x, y \in E$ it holds that*

(S1) $X \subseteq \sigma(X)$

(S2) $X \subseteq Y \Rightarrow \sigma(X) \subseteq \sigma(Y)$

(S3) $\sigma(\sigma(x)) = \sigma(x)$

(S4) $[y \notin \sigma(X) \wedge y \in \sigma(X \cup \{x\})] \Rightarrow x \in \sigma(X \cup \{y\})$

Example. $M(G)$ with connected G. $G = K_{13}$

$$\sigma(X) = \bigvee_{i=1}^{q} E(C_i)$$

where q is the number of connected components of $(V(G), X)$ and c_1 to c_q are the connected components of $(V(G), X)$.

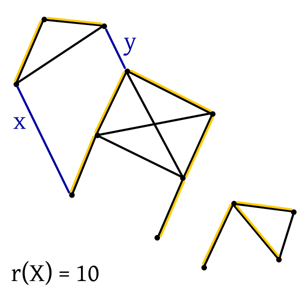


Figure 65: Example for S* statements

Theorem 85. *(Cycles) Let E be a finite set and $\mathcal{C} \subseteq 2^E$. $\mathcal{C}$ is the set of cycles of an IDS $(E, \mathcal{F})$ with $\mathcal{F} := \{F \subseteq E : \nexists C \in \mathcal{C} \text{ with } C \subseteq F\}$ if and only if the following conditions are satisfied:*

(C1) $\varnothing \notin \mathcal{C}$

(C2) $\forall C_1, C_2 \in \mathcal{C} : C_1 \subseteq C_2 \Rightarrow C_1 = C_2$

Furthermore for the set $\mathcal{C}$ of cycles of an IDS it holds that:

a) $(E, \mathcal{F})$ *is a matroid*

b) $\forall X \in \mathcal{F}\ \forall e \in E : X \cup \{e\}$ *contains at most one cycle. Denote this number of cycles as $C(X, e)$. If no cycle exists, let $C(X, e) = \varnothing$.*

where $a \Leftrightarrow b$.

Furthermore this statement is equivalent b)

(C3) $\forall C_1, C_2 \in \mathcal{C}$ *with* $C_1 \neq C_2\ \forall e \in C_1 \cap C_2, \exists C_3 \in \mathcal{C}$ *with* $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$

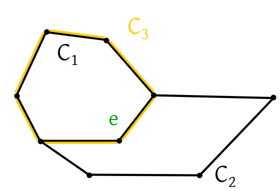
(C4) $\forall C_1, C_2 \in \mathcal{C},\ \forall e \in C_1 \cap C_2,\ \forall f \in C_1 \setminus C_2$ *exists* $C_3 \in \mathcal{C}$ *with* $f \in C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$.

Example. $M(G)$ and G is connected. Cycles in $M(G)$ are cycles in a graph-theoretical sense

(C1) trivial

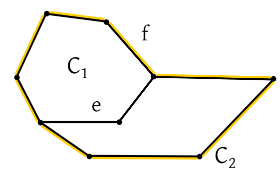
(C2) Cycle does not have proper subset which is cycle again.

b) X is cycle-free because $X \in \mathcal{F}$. One additional edge gives at most one cycle.



(C3)

Figure 66: Example for (C3)



(C4)

Figure 67: Example for (C4)

21.9 Duality of matroids

Let $(E, \mathcal{F})$ be an IDS. The dual of $(E, \mathcal{F})$ is the family of sets $(E, \mathcal{F}^*)$ with

$$\mathcal{F}^* := \{F \subseteq E : \exists \text{ base of } (E, \mathcal{F}) \text{ such that } F \cap B = \varnothing\}$$

Question. Is $(E, \mathcal{F}^*)$ an IDS? Yes.

(M1) $\varnothing \in \mathcal{F}^*$ $\forall$ base $\mathcal{B}$ of $(E, \mathcal{F})$ it holds that $\varnothing \cap B = \varnothing$

(M2) $Y \in \mathcal{F}^* : X \subseteq Y \overset{!}{\Rightarrow} X \subseteq \mathcal{F}^*$
From Y it follows that $\exists$ a base B of $(E, \mathcal{F}) : Y \cap B = \varnothing \Rightarrow X \cap B = \varnothing \Rightarrow X \in \mathcal{F}^*$

Hence a dual of an IDS is an IDS.

Theorem 86. *It holds that* $(E, \mathcal{F}^{**}) = (E, \mathcal{F})$

Proof. $F \in \mathcal{F}^{**} \overset{\text{def}}{\Leftrightarrow} \exists$ base B^* in $(E, \mathcal{F}^*)$ with $F \cap B^* = \varnothing$. $\Leftrightarrow \exists$ base B in $(E, \mathcal{F})$ with $B^* = B^C$ and $F \cap B^C = \varnothing$ hence $F \subseteq B \Leftrightarrow F \in \mathcal{F}$

Claim. B^* base of $(E, \mathcal{F}^*) \Leftrightarrow \exists$ base B of $(E, \mathcal{F}) : B^* = B^C$

Proof. B^* base of $(E, \mathcal{F}^*) \Rightarrow B^* \in \mathcal{F}^* \overset{\text{def}}{\Rightarrow} \exists$ base B of $(E, \mathcal{F})$

$\Rightarrow$ $B^* \subseteq B^C$. Assume that $\exists x \in B^C \setminus B^*$. Then $B^* \cup \{x\} \subseteq B^C$

$\Rightarrow$ $(B^* \cup \{x\}) \cap B = \varnothing \overset{\text{def}}{\Rightarrow} B^* \cap \{x\} \in \mathcal{F}^*$. This contradicts with B^* is a base of $(E, \mathcal{F}^*)$.

Hence $B^* = B^C$. □

Theorem 87. *B^* base of $(E, \mathcal{F}^*) \Leftrightarrow \exists$ base B of $(E, \mathcal{F})$ with $B^* = B^C$ and $(E, \mathcal{F}^*)$ its dual. Let r and r^* be the corresponding rank functions. Then it holds that* *a)* $(E, \mathcal{F})$ *is a matroid* $\Leftrightarrow$ $(E, \mathcal{F}^*)$ *is matroid*

b) If $(E, \mathcal{F})$ *is a matroid, then it holds that* $r^*(F) = |F| + r(E \setminus F) - r(E) \; \forall F \subseteq E$

Example. $(M(G))^* = M(G^*) \; \forall G$ planar. A dual graph contains one vertex per area and 1 infinite area. An edge between 2 new vertices is created, if they are neighbors (share one edge). $(M(G))^* = M(G^*) \; \forall G$ planar.

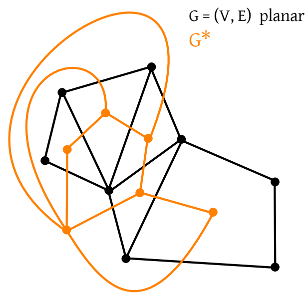


Figure 68: Duality for Theorem 87

This lecture took place on 27th of January 2015.

Theorem 88. *Let G be a connected planar graph with an arbitrary planar embedding. Let G^* be the planar duality of G. Let $M(G)$ be the graphical matroid of G. It holds that*

$$M^*(G) = (M(G))^* = M(G^*)$$

Furthermore G is planar if and only if $(M(G))^*$ is graphical; hence if a graph G' with $M(G') = (M(G))^*$ exists.

If G is planar, then G' is isomorphic to a planar embedding of G^*.

21.10 The greedy algorithm

Let $(E, \mathcal{F})$ be a IDS and $c : E \to \mathbb{R}_+$ (without loss of generality). A maximization problem is given by $\max \left\{ c(F) := \sum_{c \in F} c(e) : F \in \mathcal{F} \right\}$.

Definition. 2 oracles O_1 and O_2 are called equivalent if $O_1(O_2)$ can be simulated by a polynomial algorithm using an Oracel $O_2(O_1)$ (in both directions!).

Let O_1 be an independent oracle. Let O_2 be a base superset oracle. For general IDS O_1 and O_2 are not equivalent.

Algorithm 17 BEST-IN greedy algorithm

Given. IDS $(E, \mathcal{F})$, independent oracle, $c : E \to \mathbb{R}_+$
Find. An independent set $F \in \mathcal{F}$

Sort edges $E = \{e_1, e_2, \dots, e_n\}$ such that $c(e_1) \geq c(e_2) \geq \dots \geq c(e_n)$
$F := \varnothing$
for i from 1 to n **do**
  if $F \cup \{e_i\} \in \mathcal{F}$ **then**
    $F := F \cup \{e_i\}$
  end if
end for

Algorithm 18 WORST-OUT greedy algorithm

Given. An IDS $(E, \mathcal{F})$ is given by base supersets oracle (does given set contain a base?)
Find. B base of $(E, \mathcal{F})$

Sort edges $E = \{e_1, e_2, \dots, e_n\}$ such that $c(e_1) \geq c(e_2) \geq \dots \geq c(e_n)$
$F := E$
for i from 1 to n **do**
  if $F|\{e_i\}$ contains base **then**
    $F := F|\{e_i\}$
  end if
end for

21.11 IDS for TSP on K_n (complete graph)

E are edges and $\mathcal{F}$ is the subset of Hamiltonian cycles.

O_1 is polynomial? $F \in \mathcal{F}$ (TM Hamiltonian cycle) $\Leftrightarrow \deg_F(v) \leq 2$ and no subcycles can be computed in polynomial time

O_2 is polynomial? (Base OM oracle) $F \leq E(K_n)$. $\exists$ base B of $(E, \mathcal{F})$ with $B \subseteq F$? Bases are Hamiltonian cycle. Hence not polynomial. Contradiction. Is actually NP-hard.

One direction O_1 is simple. O_2 difficult…other way also possible?

IDS for shortest s-t-path system (F are edges in s-t-path (parts of the path; base is complete path)). O_2 is polynomial? $F \subseteq E(G)$ is there a base B with $B \subseteq F$? Hence "Does some s-t-path exist in $(V(G), F)$?". Polynomial behavior follows.

O_1 is polynomial? $F \subseteq \mathcal{F}$: Is F independent $\Leftrightarrow$ "F is subset of a s-t-path in G?" This is NP-complete problem and hence likely not polynomial (Korte and Monna, 1979)

O_1 and O_2 are for general IDS in complexity independent of each other.

Remark. If $(E, \mathcal{F})$ is a matroid, then O_1 (rank oracle) and O_2 (closure oracle) equivalent under each other. *But* don't have an equivalent base oracle (oracle for determination of an dependent TM with minimal cardinality)

Theorem 89. *(Jenkyns, Korte, Hausmann, 1978) Let $(E, \mathcal{F})$ be an IDS and $c : E \to \mathbb{R}_+$. Denote $G(E, \mathcal{F}, c)$ as the costs of a solution determined by the BEST-IN-GREEDY algorithm. Denote $\mathrm{OPT}(E, \mathcal{F}, c)$ as the costs of an optimal solution (both for the maximization problem the GREEDY-IN algorithm is tackling).*

Then it holds that

$$q(E, \mathcal{F}) \leq \frac{G(E, \mathcal{F}, c)}{\mathrm{OPT}(E, \mathcal{F}, c)} \underbrace{\leq}_{trivial} 1 \; \forall \, c : E \to \mathbb{R}_+$$

Furthermore the best lower bound is $\exists c : E \to \mathbb{R}_+$ with $q(E, \mathcal{F}) = \frac{G(E, \mathcal{F}, c)}{\mathrm{OPT}(E, \mathcal{F}, c)}$.

Proof. $E = \{e_1, e_2, \ldots, e_n\}$ with $c(e_1) \geq c(e_2) \geq \ldots \geq c(e_n)$. Let G_n be the solution determined by BEST-IN-GREEDY and O_n be an optimal solution. Let $E_j = \{e_1, e_2, \ldots, e_j\}$, $G_j = G_n \cap E_j$, $O_j = O_n \cap E_j \; \forall \, j = 0, 1, \ldots, n$. Let $d_j = c(e_j) - c(e_{j+1}) \geq 0 \; \forall \, j = 1, \ldots, n-1$ and $d_n = c(e_n)$.

- $\forall \, j : Oj \in \mathcal{F} \Rightarrow |O_j| \leq r(E_j)$
- G_j is base of E_j. Hence G_j is inclusive maximal independent submatrix of E_j, otherwise $\exists \overline{G_j} \underset{\neq}{\supseteq} G_j$ with $\overline{G_j} \subseteq E_j$ (where $E_j = \{e_1, \ldots, e_j\}$) and $\overline{G_j} \in \mathcal{F}$. This contradicts with the way BEST-IN-GREEDY works.

□

$|G_j| \geq \rho(E_j)$

$$q(E, \mathcal{F}) = \min_{X \subseteq E} \frac{\rho(X)}{r(X)} \leq \frac{\rho(x)}{r(x)}$$

$$\rho(X) \geq q(E, \mathcal{F}) r(X) \; \forall \, X \subseteq E$$

$$
\begin{aligned}
G(E,\mathcal{F},c) = c(G_n) &= \sum_{j=1}^{n} \underbrace{(|G_j| - |G_{j-1}|)}_{1 \text{ or } 0 \text{ depending on } e_j} \; c(e_j) = \sum_{e_j \in G_n} c(e_j) \\
&= (|G_1| - \underbrace{|G_0|}_{=0})c(e_1) + (|G_2| - |G_1|)c(e_2) + \ldots + (|G_n| - |G_{n-1}|)c(e_n) \\
&= |G_1| \underbrace{(c(e_1) - c(e_2))}_{d_1} + |G_2| \underbrace{(c(e_2) + c(e_3))}_{d_2} + \ldots + |G_n| \underbrace{c(e_n)}_{d_n} \\
&= \sum_{j=1}^{n} |G_j| \, d_j \geq \sum_{j=1}^{n} \rho(E_j) d_j \geq q(E,\mathcal{F}) \sum_{j=1}^{n} r(E_j) d_j \\
&\geq |O_1| \, (c(1) - c(2)) + \ldots + |O_n| \, c(e_n) = |O_1| \, c(e_1) + (|O_2| - |O_1|)c(e_2) + \ldots + (|O_n| - |O_{n-1}|)c(e_n) \\
&\geq q(E,\mathcal{F}) \sum_{j=1}^{n} (|O_j| - |O_{j-1}|) \, c(e_j)
\end{aligned}
$$

$$
\Rightarrow \frac{G(E,\mathcal{F},c)}{\mathrm{OPT}(E,\mathcal{F},c)} = \frac{c(G_n)}{c(O_n)} \geq q(E,\mathcal{F})
$$

$|O_j| - |O_{j+1}|$ is again either 0 or 1 depending on existence of j in $O_{j+1} \ldots O_n$.

Furthermore we also show now that the bound $q(E,\mathcal{F})$ is optimal: Let $F \subseteq E$ with $q(E,\mathcal{F}) = \frac{\rho(F)}{r(F)} = \min_{X \subseteq E} \frac{\rho(X)}{r(X)}$. Let B_1 and B_2 bases of F such that $|B_2| = r(F)$ where r is the cardinality of the greatest independent submatrix and $|B_1| = \rho(F)$. Let $\bar{c} : E \to \mathbb{R}_+$ with

$$
\bar{c}(e) = \begin{cases} 1 & e \in F \\ 0 & \text{else} \end{cases}
$$

BEST-IN-GREEDY: Sort $E = \{e_1, \ldots, e_n\}$ with $\bar{c}(e_1) \geq c(e_2) \geq \ldots \geq \bar{c}(e_n)$ where $B_1 = \left\{e_1, \ldots, e_{|B_1|}\right\}$. This returns B_1 (because $e_1, \ldots, e_{|B_1|}$ must be connected first and B_1 is base and therefore has maximal cardinality). Then $G(E,\mathcal{F},\bar{c}) = |B_1| = \rho(F)$.

$$
\mathrm{OPT}(E,\mathcal{F},\bar{c}) = |B_2| = r(F)
$$

$r(F)$ is the cardinality of a maximal independent set. The weights are 1. $r(F)$ is optimal.

$$
q(E,\mathcal{F}) = \frac{\rho(F)}{r(F)} = \frac{G(E,\mathcal{F},\bar{c})}{\mathrm{OPT}(E,\mathcal{F},\bar{c})}
$$

$\Rightarrow$ minimal $q(E,\mathcal{F})$ is always achieved by at least one c.

Theorem 90. *(Edmonds, Rado, 1971) An IDS $(E,\mathcal{F})$ is a matroix if and only if the BEST-IN-GREEDY algorithm provides an optimal solution for the maximization problem $\forall\, c : E \to R_+$.*

Proof. From Theorem 89 it follows that

$$q(E,\mathcal{F}) \leq \frac{G(E,\mathcal{F},c)}{\mathrm{OPT}(E,\mathcal{F},c)} \leq 1 \;\forall\, c : E \to \mathbb{R}_+$$

From Theorem 80 it follows that $(E,\mathcal{F})$ is matroid $\Leftrightarrow q(E,\mathcal{F}) = 1$. Because the lower bound is always reached, it follows

$$(E,\mathcal{F}) \text{ is matroid } \Leftrightarrow \frac{G(E,\mathcal{F},c)}{\mathrm{OPT}(E,\mathcal{F},c)} = 1 \;\forall\, c : E \to \mathbb{R}_+$$

$\square$

Remark. For matroids, minimization and maximization problems are equivalent. Hence the maximization problem handled by the GREEDY-IN algorithm is equivalent to $\min\left\{\overline{c}(B) = \sum_{e\in B}\overline{c}(e) : B \text{ is base of } (E,\mathcal{F})\right\}$ where $\overline{c}(e) = M - c(e)$ with $M = \max_{e\in E}|c(e)| + 1$.

Theorem 91. *(Edmonds 1971, polyedric representation) Let $(E,\mathcal{F})$ be a matroid and $r : E \to \mathbb{Z}_+$ be a rank function. Then the matroid polytop $P(E,\mathcal{F})$ (convex hull of incidence vectors of all independent sets) is given by:*

$$P(E,\mathcal{F}) = \left\{ x \in \mathbb{R}^{|E|} : x \geq 0, \sum_{e\in A} x_e \leq r(A) \;\forall\, A \subseteq E \right\}$$

$$F \in \mathcal{F} \qquad \underbrace{x^F(e)}_{\textit{incidence vectors}} = \begin{cases} 1 & e \in F \\ 0 & \textit{else} \end{cases} \qquad \sum_{e\in A} x_e^F = |A \cap F| \leq r(A)$$

We conclude: $x^F \in P(E,\mathcal{F}) \;\forall\, F \in \mathcal{F}$.

Example. $P(M(G))$ is the graphical matroid of the graph connected components.

$$P(M(G)) = \left\{ x \in \mathbb{R}^{\{E(G)\}} : x \geq 0, \qquad \underbrace{\sum_{e\in A} x_e \leq n-1}_{\text{independent of components of } M(G) \text{ are forests}} \qquad \forall\, A \subsetneq E(G), \sum_{e\in A} x_e = n-1, A = E(\right.$$

Index

$q_G(X)$, 101

alignment polyeder, 125
almost perfect matching, 102
alternating forest, 106
alternating tree, 106

barrier, 104
Bottleneck edge, 47

deficiency, 111
double-stochastic matrix, 125

edge cover number, 127
even vertices, 106

factor-critical graphs, 102

graphical matroid, 132

independent set, 127
integral polyeder, 117

MA-order, 65
matching number, 127
matroid, 129
    bases, 129
    cycles, 129
    lower rank, 135

odd vertices, 106

Polyeder, 117
polynomial runtime, 9

stability number, 127
stable set, 127
strongly polynomial, 9

Theorem by Tutte, 103
total unimodular matrix, 118
Tutte theorem, 103

Uniform matroid, 133
unimodular matrix, 117

vertex cover number, 127