



Fakultät für Mathematik
Lehrstuhl für Angewandte Geometrie und Diskrete Mathematik

Visualisierung von Flussalgorithmen

Ford-Fulkerson Algorithmus

zur Lösung des Maximum Flow Problems

Cycle Canceling Algorithmus

zur Lösung des Minimum-Cost Flow Problems

Interdisziplinäres Projekt (IDP) von Quirin Fischer

Themensteller: Prof. Dr. Peter Gritzmann

Betreuer: M.Sc. Wolfgang F. Riedl

Abgabedatum: 30.09.2016

Hiermit erkläre ich, dass ich diese Arbeit selbstständig angefertigt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 30.09.2016

Quirin Fischer

Abstract

This interdisciplinary project encompasses the development of interactive applications to visualise two flow algorithms: The Ford-Fulkerson algorithm for solving the maximum-flow problem, and the cycle canceling algorithm for solving the minimum-cost flow problem. This report presents the algorithms, describes the layout of the applications, and explains the technical realisation.

The aim of the applications is to facilitate the comprehension of the algorithms by allowing the step-by-step execution on any desired instance of the problem, displaying the computational phases transparently, and graphically visualising the data structures.

The implementation is done as an application for web browsers by using an existing framework which uses modern web standards like HTML5, SVG, CSS, Javascript and D3.js. In particular, multiple important components of the implementation and their interactions are described in order to give a clear overview of the code base.

Zusammenfassung

Dieses interdisziplinäre Projekt umfasst die Entwicklung von interaktiven Anwendungen für die Visualisierung zweier Flussalgorithmen: Des Ford-Fulkerson Algorithmus zur Lösung des Maximum-Flow Problems, und des Cycle-Canceling Algorithmus zur Lösung des Minimum-Cost Flow Problems. Dieser Bericht stellt die Algorithmen vor, beschreibt den Aufbau der Applikationen, und erläutert die technische Umsetzung.

Das Ziel der Applikationen ist das Verständnis der Algorithmen zu erleichtern, indem eine schrittweise Ausführung an beliebigen Instanzen des Problems ermöglicht, die Berechnungsschritte sichtbar gemacht und Datenstrukturen anschaulich visualisiert werden.

Die Implementierung erfolgt als Webapplikation für den Browser mithilfe eines bestehenden Frameworks, das moderne Webstandards wie HTML5, SVG, CSS, Javascript und D3.js verwendet. Insbesondere werden verschiedene wichtigen Komponenten der Implementierung und deren Zusammenspiel behandelt, um einen klaren Überblick über den Code zu geben.

Inhaltsverzeichnis

Abstract / Zusammenfassung	v
Inhaltsverzeichnis	vii
Einleitung	1
1 Behandelte Flussalgorithmen	3
1.1 Flüsse in Netzwerken	3
1.2 Ford-Fulkerson Algorithmus	5
1.3 Cycle Canceling Algorithmus	7
2 Aufbau der Webapplikationen	9
2.1 Framework für Graphalgorithmen	9
2.2 Visualisierung des Ford-Fulkerson Algorithmus	12
2.3 Visualisierung des Cycle Canceling Algorithmus	12
3 Implementierung	13
3.1 Technische Konzepte	13
3.1.1 Statische Elemente und Formatierung	13
3.1.2 Dynamische Elemente und Kontrollfluss	13
3.2 Code-Komponenten und deren Interaktion	13
3.2.1 Struktur der statischen Seite	13
3.2.2 Initialisierung der Applikation	13
3.2.3 Verwaltung der Graphendaten	13
3.2.4 Schrittweise Ausführung des Algorithmus	13
3.2.5 Darstellung der Graph-Visualisierung	13
3.2.6 Synchronisierung von Anzeige und Berechnungszustand	13
Fazit	14
Abbildungsverzeichnis	17
Listings	19
Literatur	21

Einleitung

Die Graphentheorie ist ein Teilgebiet der (diskreten) Mathematik mit einigen herausstellenden Merkmalen: Sie ist sehr anwendungsbezogen, da viele praktische Entscheidungsprobleme mithilfe von Graphen modelliert werden können. Sie hat weiterhin eine sehr enge Verbindung zur Algorithmik, wobei sowohl algorithmische Probleme mit bestimmten Graphen zusammenhängen, als auch Probleme der Graphentheorie mithilfe von Algorithmen gelöst werden, und daraus folgend auch zur Informatik. Aufgrund ihrer Anschaulichkeit und der Nützlichkeit der Konzepte ist die Graphentheorie einer der ersten Bereiche der Mathematik mit dem Studenten mathematischer, informatischer oder technischer Fachrichtungen in Berührung kommen.

Eine besonders charakteristischer Teil hierbei ist die Beschäftigung mit verschiedenen Graphalgorithmen. Das Verständnis der Abläufe, der Eigenschaften und Strukturen, die der einzelne Algorithmus ausnutzt, und den entsprechenden Eigenschaften der resultierenden Lösungen gibt eine vielseitige Perspektive auf die behandelten Problemstellungen. Ein wichtiges Hilfsmittel für die Einarbeitung in neue Algorithmen sind dabei die ausführliche Durchführung des Algorithmus auf verschiedenen Beispielen und Betrachtung der Berechnungsschritte. Insbesondere die Visualisierung des Graphen mit verschiedenen Annotationen ist oftmals hilfreich um ein Gefühl für die Vorgänge zu bekommen.

Eine Klasse von Graphalgorithmen mit vielseitigen Anwendungen sind Flussprobleme. Abgeleitet von praktischen Fragestellungen zum Transport in verschiedenartigen Netzwerken, man denke an Rohrnetze, Straßennetze oder Datennetzwerke, behandeln sie Optimierungsprobleme wie eine Menge an Ressourcen von einem Start- zu einem Zielknoten gelangen kann. Je nach speziellen Anforderungen wird der Graph, der die Netzwerktopologie modelliert, mit verschiedenen Informationen annotiert. Typischerweise sind dies Kapazitäten (verschiedene Teile des Netzwerkes können nur bestimmte Mengen an Ressource transportieren) oder Kosten (die Benutzung mancher Teile des Netzwerkes ist anderen vorzuziehen).

Dieses Dokument beschreibt die Entwicklung einer interaktiven Anwendung zur Visualisierung zweier Flussalgorithmen - des Ford-Fulkerson Algorithmus für das Maximum-Flow Problem, und des Cycle Canceling Algorithmus für das Min-Cost Flow Problem. Die Anwendung hat das Ziel das Verständnis der Algorithmen zu erleichtern, indem sie einfach Schritt für Schritt auf verschiedene Instanzen des Problems ausgeführt werden können, mit transparenter Anzeige des Berechnungszustandes und anschaulicher Darstellung der Datenstrukturen.

Im erste Kapitel werden die zwei behandelten Probleme und Lösungsalgorithmen behandelt. Im zweiten Kapitel wird der Aufbau der Anwendung und ihre verschiedenen Elemente beschrieben. Das dritte Kapitel beschreibt die technische Umsetzung, mit zunächst einem Überblick über die verwendeten Basistechnologien und schließlich einer Behandlung der Implementierung interessanter Komponenten und deren Zusammenspiel.

Kapitel 1

Behandelte Flussalgorithmen

1.1 Flüsse in Netzwerken

Anschaulich entsprechen Flüsse in Netzwerken der Modellierung von Transportsystemen - Anwendungsfälle sind beispielsweise Wassernetze (wobei Kapazitäten den Rohrdurchmessern entsprechen, und die Menge an geflossenem Wasser dem Fluss), Straßensysteme (Kapazität: Spurbreite, Fluss: beförderte Fahrzeuge) oder Datennetze (Kapazität: Übertragungsgeschwindigkeit, Fluss: übermittelte Datenmenge).

Als Netzwerk bezeichnet man einen gerichteten Graphen $G = (V, E)$ annotiert mit einer Kapazitätsfunktion für Kanten $u : E \rightarrow \mathbb{R}^+$ (dem *upper limit*), Startknoten s (der *source*) und Zielknoten t (*target*). Insgesamt notiert man folglich ein Netzwerk als 4-Tupel $N = (G, u, s, t)$. [Wik16]

Ein Fluss in einem Netz ist definiert als eine Funktion $f : E \rightarrow \mathbb{R}^+$ (auch *flow*), die für jede Kante des Graphen eine Menge an Fluss bestimmt. Ein gültiger s-t-Fluss f in einem Netzwerk muss weiterhin zwei Bedingungen erfüllen:

Kapazitätskonformität Der Fluss muss auf allen Kanten unter deren Kapazitätsgrenze liegen:

$$\forall e \in E : f(e) \leq u(e)$$

Flusserhalt Der Fluss ist in allen Knoten abgesehen von s und t konstant, es gelangt gleich viel Fluss zum Knoten, wie ihn verlässt. Bezeichnet man die Menge an eingehenden Kanten eines Knotens als $\delta^-(v) \stackrel{\text{def}}{=} \{e = (x, v) \in E \mid x \in V\}$ und die Menge an ausgehenden Kanten als $\delta^+(v) \stackrel{\text{def}}{=} \{e = (v, x) \in E \mid x \in V\}$, so entspricht dies der Bedingung

$$\forall v \in V \setminus \{s, t\} : \sum_{e \in \delta^-(v)} f(e) = \sum_{e \in \delta^+(v)} f(e)$$

Ein wichtiges Konzept für Flussalgorithmen ist die Konstruktion des zu einem Fluss gehörigen Residualnetzwerkes. Dieses verwendet die gleiche Knotenmenge, enthält allerdings eine neu konstruierte Kantenmenge und andere Kapazitäten. Intuitiv modelliert

das Residualnetzwerk welche Änderungen des Flusses möglich sind - es enthält jede Kanten der ursprünglichen Netzwerks, falls zusätzlicher Fluss auf der Kante möglich ist, mit angepasster Kapazität. Für Kanten, die bereits Fluss tragen, werden zusätzlich Rückkanten mit Kapazität entsprechend des Flusses auf der Kante eingefügt, da es möglich wäre, diesen Fluss wieder zurückzuleiten.

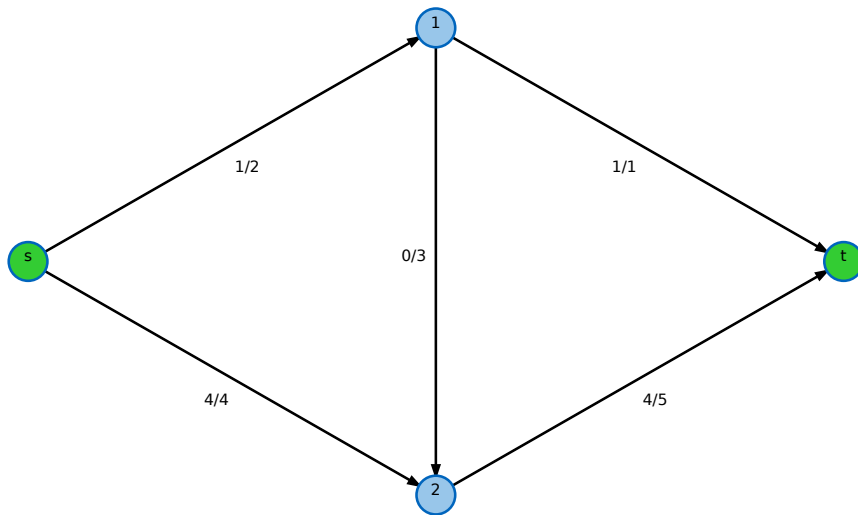


Abbildung 1.1: Ein Netzwerk mit Fluss.

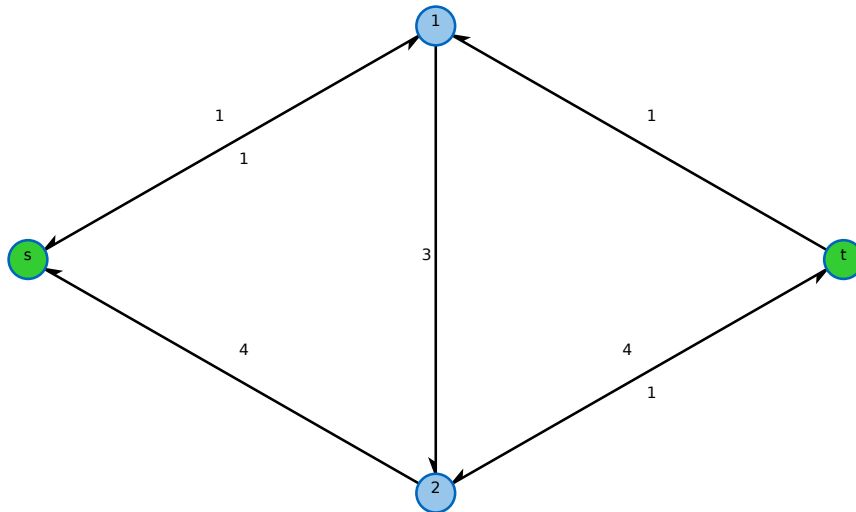


Abbildung 1.2: Das zugehörige Residualnetzwerk.

1.2 Ford-Fulkerson Algorithmus

Eine der intuitivsten Fragestellungen für ein gegebenes Netzwerk ist, wie viel Fluss gleichzeitig transportiert werden kann. Dies entspricht der Suche nach einem maximalen Fluss, das *maximum flow problem*. Zur Berechnung maximaler Flüsse gibt es eine Reihe verschiedener Methoden, die sich in der Komplexität der Berechnung und Effizienz unterscheiden. Ein einfacher Algorithmus, der gleichzeitig Ausgangspunkt für eine Reihe an fortgeschritteneren Methoden bietet, ist der *Ford-Fulkerson Algorithmus*. [FF56]

Der Ford-Fulkerson Algorithmus berechnet den maximal möglichen Fluss, indem ausgehend von einem gültigen Fluss (dem Null-Fluss) wiederholt zusätzlicher Fluss durch das Netzwerk geschickt wird, ohne die Bedingungen zu verletzen. Sobald keine weiteren Änderungen mehr möglich sind ist der maximale Fluss gefunden.

Um eine gültige Änderung zu finden wird zunächst das Residualnetz des aktuellen Flusses konstruiert. In diesem wird dann ein Pfad gesucht, der Start- und Zielknoten verbindet. Entlang dieses *augmentierenden Pfades* wird dann der Fluss um so viel erhöht, bis eine Kante saturiert und aus dem Residualnetz entfernt wird. Dies ist immer die Kante des Pfades mit der minimalen Residual-Kapazität. Diese Prozedur wird dann mit dem neuen Fluss wiederholt, bis kein Pfad mehr gefunden werden kann.

Die genaue Vorgehensweise für die Suche nach dem augmentierenden Pfad ist für den Ford-Fulkerson Algorithmus nicht exakt spezifiziert - sie hat allerdings starke Auswirkungen auf die Effizienz des Algorithmus. Naive Tiefensuche führt zu nicht-polynomieller Laufzeit $\mathcal{O}(|V| \cdot |E| \cdot u_{\max})$, und auch das nur falls alle Kapazitäten ganzzahlig sind. Ein Breitensuche-Verfahren läuft dagegen schon in $\mathcal{O}(|V| \cdot |E|^2)$, diese spezielle Anwendung wird als *Edmonds-Karp-Algorithmus* bezeichnet. [EK72] Eine weitere Verbesserung stellt mit einer Laufzeit von $\mathcal{O}(|V|^2 \cdot |E|)$ der *Algorithmus von Dinic* dar, der nur kürzeste augmentierende Pfade verwendet. [Din70][Din06]

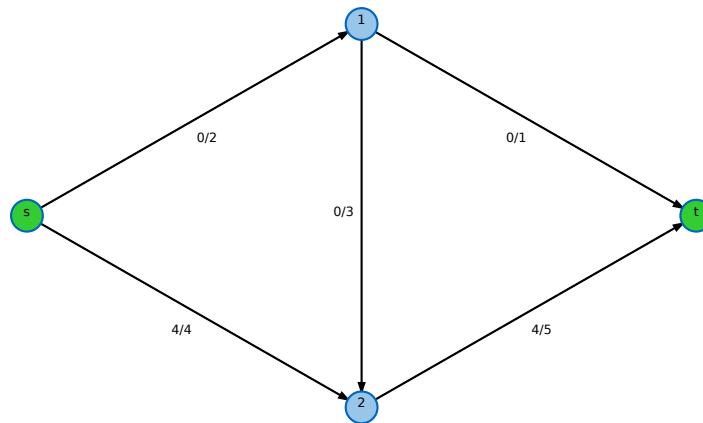


Abbildung 1.3: Ein initialer Fluss.

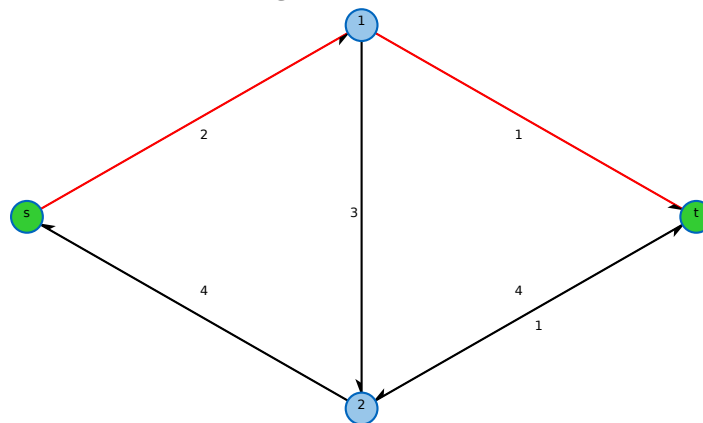


Abbildung 1.4: Ein augmentierender Pfad im Residualnetz.

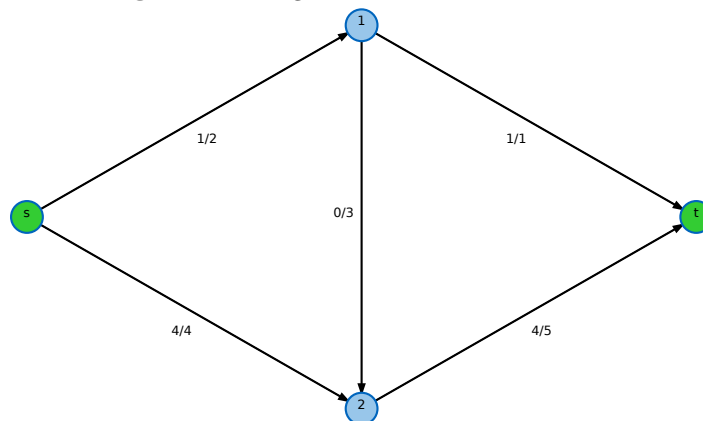


Abbildung 1.5: Der neue Fluss nach Modifikation.

1.3 Cycle Canceling Algorithmus

Eine neue Klasse von Problemen stellt sich, sobald das Netzwerk zusätzlich mit einer Kostenfunktion auf den Kanten annotiert wird. Dies ist eine Funktion $c : E \rightarrow \mathbb{R}$ die jeder Kante zuordnet, wie teuer es ist, eine Einheit Fluss über die Kante zu transportieren. Für einen Fluss ergeben sich dann die Gesamtkosten als

$$C = \sum_{e \in E} c(e) \cdot f(e)$$

Es ergibt sich die Frage nach der günstigsten Möglichkeit, eine bestimmte Menge an Fluss durch das Netzwerk zu leiten - das *min-cost flow problem*. Eine übliche Variation ist zusätzlich den größtmöglichen Fluss (nach wie vor zu niedrigst möglichen Kosten) zu suchen, was man als *min-cost max flow problem* bezeichnet.

Ein relativ einfacher Algorithmus zur Berechnung von Flüssen mit minimalen Kosten ist der *Cycle Cancelling Algorithmus*. [Kle67] Der Ansatz ist dem Ford-Fulkerson Algorithmus ähnlich, da auch hier eine vorläufige Lösung, ein gültiger Fluss der gesuchten Gesamtmenge, iterativ verbessert wird. Dazu werden Kreise im Residualnetz gesucht, die negative Gesamtkosten besitzen. Die Umleitung von Fluss entlang eines solchen Kreises verändert den Gesamtfluss nicht, senkt allerdings die Kosten. Es kann so viel Fluss umgeleitet werden, bis die Kante des Kreises mit geringster Residualkapazität aus dem Residualnetz entfernt werden muss. Dieser Ablauf kann wiederholt werden, bis keine negativen Kreise mehr im Residualnetz gefunden werden.

Zur Suche nach negativen Kreisen eignet sich der *Bellman-Ford Algorithmus*, ausgeführt auf dem Residualgraphen mit den Kantenkosten als Gewichtungen und ausgehend vom Zielknoten.

Der allgemeine Cycle Canceling Algorithmus führt zu einer nichtpolynomiellen Laufzeit in $\mathcal{O}(|V| \cdot |E| \cdot C_{max})$, da jede Iteration eine Ausführung des Bellman-Ford Algorithmus mit Laufzeit $\mathcal{O}(|V| \cdot |E|)$ erfordert und die Kosten um mindestens 1 verringert. Eine Abwandlung mit polynomieller Laufzeit stellt das *Minimum Mean Cycle Canceling Verfahren* dar. [GT89]

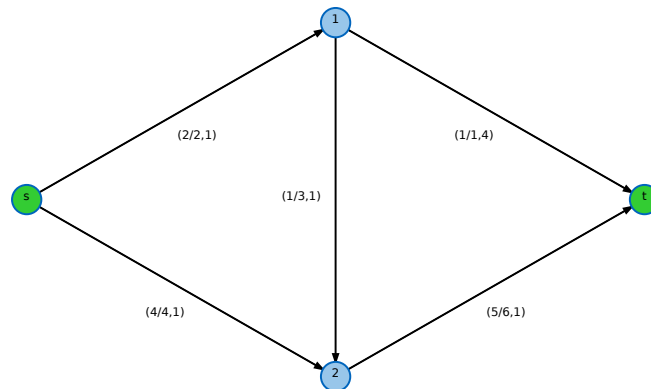


Abbildung 1.6: Ein initialer maximaler Fluss.

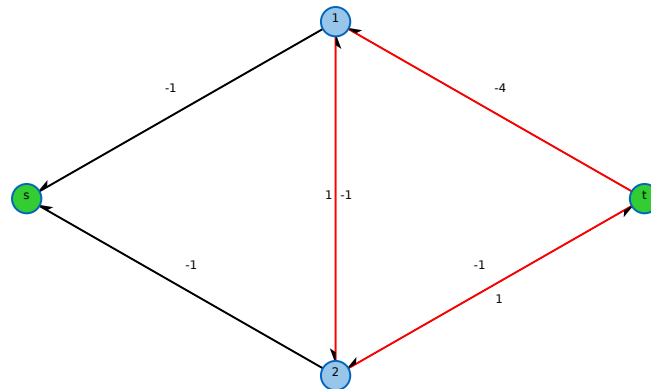


Abbildung 1.7: Das Residualnetz enthält einen negativen Kreis.

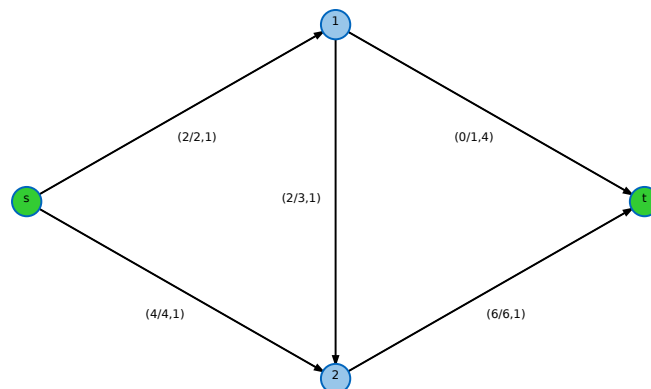


Abbildung 1.8: Fluss nach Verschiebung entlang des negativen Kreises.

Kapitel 2

Aufbau der Webapplikationen

2.1 Framework für Graphalgorithmen

Die Webapplikationen wurden in Analogie zu anderen, bereits bestehenden Applets für die Visualisierung verschiedener Graphalgorithmen konzeptioniert. Diese sind frei zugänglich auf der Webpräsenz des Lehrstuhls M5 für Mathematik an der TUM, erreichbar unter <https://www-m9.ma.tum.de/Allgemeines/GraphAlgorithmen>.

Die bestehende Sammlung an Applets enthält bereits verschiedene andere Algorithmen der Graphentheorie, von Lösungsmethoden für des kürzeste Wege, Matchings, Spannbäume und mehr. Alle Anwendungen sind mit den gleichen Zielen entwickelt, die Algorithmen möglichst anschaulich und verständlich zu präsentieren, und verwenden das gleiche Seitenlayout. Auch die hier vorgestellten Applets zur Darstellung des Ford-Fulkerson Algorithmus und des Cycle Canceling Algorithmus stellen die Informationen in diesem bewährten Layout dar. Die Inhalte sind auf mehrere Tabs verteilt, die verschiedene Materialien bieten.

Tab: Einführung Die erste Einführung für den Benutzer gibt einen kurzen Überblick über die Problemstellung, die mithilfe des Algorithmus gelöst wird.

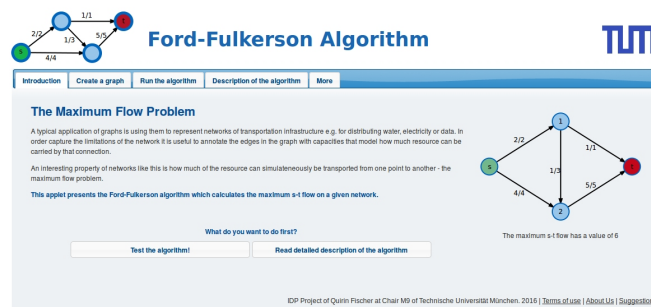


Abbildung 2.1: Der Einführungs-Tab.

Tab: Grapherstellung Ein Grapheditor, um die Instanz des Problems auszuwählen, auf der der Algorithmus ausgeführt werden soll. Der Editor bietet sowohl die Möglichkeit, vorgefertigte Beispiele zu laden, als auch diese zu verändern oder eigene Instanzen anzulegen. Knoten und Kanten des Graphen können einfach angelegt und gelöscht werden, und auf Klick ist auch die Modifikation zusätzlicher Eigenschaften wie Kantengewichte oder Kapazitäten möglich.

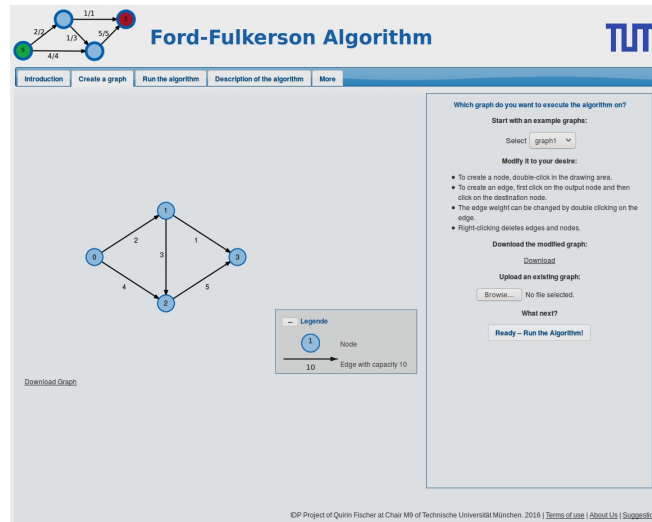


Abbildung 2.2: Der Tab des Grapheditors.

Tab: Ausführung Algorithmus Der wichtigste Tab erlaubt die schrittweise Ausführung des Algorithmus und zeigt die internen Datenstrukturen. Einige Informationen werden grafisch in die Darstellung des Graphen integriert, indem beispielsweise Knoten oder Kanten farblich hervorgehoben werden. Eine seitliche Anzeige liefert wahlweise eine textuelle Beschreibung des aktuellen Arbeitsschrittes, die aktuelle Ausführungsposition in einer Pseudocode-Diagramm, oder die Werte von Variablen die der Algorithmus verwendet.

Die darzustellenden Daten unterscheiden sich zwischen den Algorithmen, so dass die Applikationen sich in diesem Tab am deutlichsten unterscheiden.

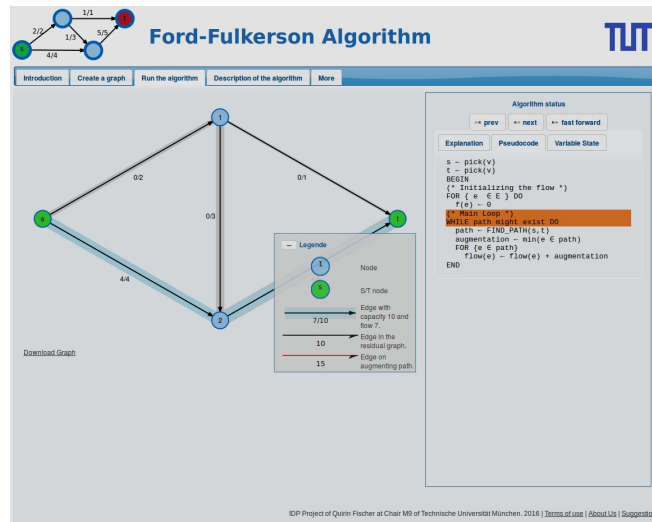


Abbildung 2.3: Der Algorithmus-Tab erlaubt schrittweise Ausführung und visualisiert den Status.

Tab: Beschreibung Algorithmus Eine ausführlichere Beschreibung erläutert die Problemstellung und den genauen Ablauf des Algorithmus.

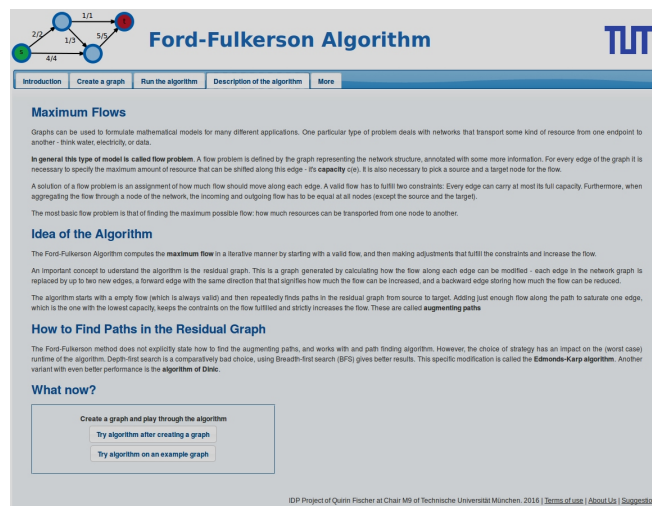


Abbildung 2.4: Eine längere Beschreibung des Algorithmus befindet sich im vierten Tab.

Tab: Weitere Informationen Eine Sammlung nützlicher Informationen wie beispielsweise einer Pseudocode-Darstellung des Algorithmus, Informationen zur Effizienz des Algorithmus, Hinweise auf abgewandelte Varianten oder eine Auswahl an Literatur und Referenzen.



Abbildung 2.5: Der letzte Tab enthält weiteres Hintergrundmaterial.

2.2 Visualisierung des Ford-Fulkerson Algorithmus

2.3 Visualisierung des Cycle Canceling Algorithmus

Kapitel 3

Implementierung

3.1 Technische Konzepte

3.1.1 Statische Elemente und Formatierung

HTML5

CSS

SVG

3.1.2 Dynamische Elemente und Kontrollfluss

Javascript

D3.js

3.2 Code-Komponenten und deren Interaktion

3.2.1 Struktur der statischen Seite

3.2.2 Initialisierung der Applikation

3.2.3 Verwaltung der Graphendaten

3.2.4 Schrittweise Ausführung des Algorithmus

3.2.5 Darstellung der Graph-Visualisierung

3.2.6 Synchronisierung von Anzeige und Berechnungszustand

Fazit

Abbildungsverzeichnis

1.1	Ein Netzwerk mit Fluss.	4
1.2	Das zugehörige Residualnetzwerk.	4
1.3	Ein initialer Fluss.	6
1.4	Ein augmentierender Pfad im Residualnetz.	6
1.5	Der neue Fluss nach Modifikation.	6
1.6	Ein initialer maximaler Fluss.	8
1.7	Das Residualnetz enthält einen negativen Kreis.	8
1.8	Fluss nach Verschiebung entlang des negativen Kreises.	8
2.1	Der Einführungs-Tab.	9
2.2	Der Tab des Grapheditors.	10
2.3	Der Algorithmus-Tab erlaubt schrittweise Ausführung und visualisiert den Status.	11
2.4	Eine längere Beschreibung des Algorithmus befindet sich im vierten Tab.	11
2.5	Der letzte Tab enthält weiteres Hintergrundmaterial.	12

Listings

Literatur

- [Din06] Y. Dinitz. „Theoretical Computer Science“. In: Hrsg. von O. Goldreich, A. L. Rosenberg und A. L. Selman. Berlin, Heidelberg: Springer-Verlag, 2006. Kap. Dinitz’ Algorithm: The Original Version and Even’s Version, S. 218–240. ISBN: 3-540-32880-7, 978-3-540-32880-3.
- [Din70] E. A. Dinic. „Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation“. In: *Soviet Math Doklady* 11 (1970), S. 1277–1280.
- [EK72] J. Edmonds und R. M. Karp. „Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems“. In: *J. ACM* 19.2 (Apr. 1972), S. 248–264. ISSN: 0004-5411.
- [FF56] L. R. Ford und D. R. Fulkerson. „Maximal Flow through a Network.“ In: *Canadian Journal of Mathematics* 8 (1956), S. 399–404.
- [GT89] A. V. Goldberg und R. E. Tarjan. „Finding Minimum-cost Circulations by Canceling Negative Cycles“. In: *J. ACM* 36.4 (Okt. 1989), S. 873–886. ISSN: 0004-5411.
- [Kle67] M. Klein. „A Primal Method for Minimal Cost Flows with Applications to the Assignment and Transportation Problems“. In: *Management Science* 14.3 (1967), S. 205–220. eprint: <http://dx.doi.org/10.1287/mnsc.14.3.205>.
- [Wik16] Wikipedia. *Flüsse und Schnitte in Netzwerken*. Online; accessed 29-September-2016. 2016.