Fach 1216 – Kombinatorische Optimierung WS17/18

Prüfer: Prof. Hochstättler
Datum: 13.3.2018
Dauer: ~halbe Stunde
Studiengang: Master Praktische Informatik
Note: sehr gut

(Die Prüfung hatte einen etwas ungewöhnlichen Einstieg: Zum Warm-Up haben wir über dies und das und allerlei gesprochen, was zu einer angenehmen Stimmung beitrug und gar keine Idee von Nervosität aufkommen ließ.
Als wir bei Speisewagen der ungarischen Bahn angekommen waren meinte ich, dass „Ungarn“ ein ganz gutes Stichwort wäre, woraufhin Hr. Hochstättler zu unserem Beisitzer meinte, dass ich wohl mit der Prüfung beginnen wolle. Und damit sind wir wohl deutlich weiter „hinten“ eingestiegen, als das üblich ist.)

- Warum ist „Ungarn“ ein gutes Stichwort?

Er wollte auf die „ungarische Methode“ hinaus, welche für das gewichtete Matching in bipartiten Graphen verwendet wird. Er beschränkt sich dabei jedoch nicht auf die Koeffizienten 0 und 1 wie in seinem Buch auf p. 135 erwähnt, sondern versteht unter dem gesamten Dual-Primalen Verfahren zum gewichteten Matching die „ungarische Methode“.
Also, das Problem als primales Problem aufgeschrieben, das duale Problem dazu abgeleitet, aufgeschrieben und erklärt. Dabei erwähnt, dass es zu LPs ein duales LP gibt und die Optimallösungen beider bei Existenz übereinstimmen (Duality Theorem). Erwähnung, dass Gleichungen zu nicht vorzeichenbeschränkten Dualvariablen führen Ungleichungen zu VZB.
Das Knotengewicht soll maximiert werden, so dass die Gewichtsrestriktionen der Kanten eingehalten werden. Prinzip der Complementary Slackness am Algorithmus erklärt: Es werden nacheinander Kanten „dicht“ gezogen, bis im Graph der dichten Kanten ein perfektes (kardinalitätsmaximales) Matching existiert. Ansonsten nächste Kante dicht ziehen indem auf dem Vertex Cover des Matchings Gewichte verringert, und auf den anderen Knoten erhöht werden. Dabei ist die Lösung für das duale Problem jederzeit zulässig und wird so lange erweitert, bis eine zulässige Lösung für das primale Problem entstanden ist.

- Wie findet man denn das (kardinalitäts-)maximale Matching im Graph der dichten Kanten, bzw. in einem allgemeinen bipartiten Graphen?

Methode der augmentierenden Pfade erklärt.

- Welche Beweisideen fallen Ihnen ein, dass es sich um ein maximales Matching handelt?

Max-Matching-Min-Cover: Es handelt sich um einen zertifizierenden Algorithmus, der das minimale Cover mitliefert. (Satz von König) (Theorem27)
Max-Flow-Min-Cut: Kardinalitätsmaximales Matching lässt sich als Max-Flow Problem auffassen und liefert einen Min-Cut mit. (Theorem 28)
Augmenting-Path: Matching ist maximal genau dann, wenn es keinen augmentierenden Pfad mehr gibt (Theorem 31)

- Springen wird dahin zurück, wo ich normalerweise beginne: Was wissen Sie über minimale Pfade in Graphen?

Problemstellung erklärt inklusive der verschiedenen Varianten (Problem 4). Es gibt zwei Algorithmen: Dijkstra und Bellmann-Ford. Dijkstra geht nur mit nicht-negativen Kanten, Bellmann-Ford auch mit Negativen. Negative Kreise sind immer böse.

- Wie sind die Komplexitäten der Algorithmen?

O(n log n) bzw. O(n m).

- Wieso ist Bellmann-Ford langsamer? Wie kommt der Faktor log n bei Dijkstra und n bei BF zustande?

Laufzeitunterschiede wegen Label-Setting vs. Label-Correcting und unterschiedlichen Fähigkeiten hinsichtlich negativer Kanten. Faktor log n wegen Sortieren. Faktor n weil (Gestammel meinerseits)… (Dann hat Hr. Hochstättler ein niedriges, breites Rechteck auf seinen Zettel gemalt) …weil der BF in der FIFO Implementierung die möglichen Pfade in aufsteigender Pfad-Segmentanzahl in Runden untersucht und ein Pfad maximal n-1 Kanten haben kann.

- Wofür kann man das nutzen?

Um negative Kreise zu detektieren. Wenn es nach n Aufdatierungen noch Änderungen gibt, dann gibt es negative Kreise, die man per Backtracking identifizieren kann.

- Warum bekommt man bei den kürzesten Pfaden die kürzesten Pfade nicht nur zwischen s und t, sondern gleich zwischen s und allen anderen?

(Hier habe ich ziemlich rum geeiert. Im Endeffekt wollte er darauf hinaus (und dort sind wir schließlich auch angekommen), dass bei Interpretation als kostenminimaler Einheitsfluss das duale Problem gleich alle Distanzen als Dualvariablen mitliefert und damit alle notwendige Information gleich vorhanden ist.)

- Minimal-Kosten-Flüsse?

Problem kurz erklärt. Dann Successive Shortest Path kurz angerissen. Dann noch den negative-Cycle-Cancelling in Grundzügen erklärt: Bestimme Maximalfluss durch Einführung von Pseudoknoten s* und t* und eliminiere dann die negativen Kreise im Residualnetzwerk.

- Was ist interessant/anders als sonst bei den beiden Algorithmen?

Laufzeit ist nicht polynomiell sondern hängt von den Kapazitäten und Supply-/Demand ab. Negative-Cycle-Cancelling kann man aber umbauen ein strikt polynomielles Verfahren. Außerdem gibt es die Skalierungstechniken um schwach polynomielle Algorithmen abzuleiten.

- Wie löst man denn den Max-Flow, der ja beim negative-Cycle-Cancelling die primal zulässige Anfangslösung liefert?

Per Preflow-Push. Angefangen zu erklären. (Einwurf Hochstättler: Wir wollen ihn doch „Push-Relabel“ nennen. Da legt Goldberg wert drauf.) Ok, dann halt Tarjan-Goldberg (Einwurf Hochstättler:

Dann doch lieber Goldberg-Tarjan. Mathematiker sind eitel.)[1] Den Algorithmus selber aber nicht mehr besonders ausführlich erklären müssen.

Inzwischen war mir dann endlich auch der Ford-Fulkerson eingefallen. Diesen erwähnt und erklärt, dass die generische Implementierung exponentielle Laufzeiten haben kann und nicht zwangsläufig konvergiert. Das kann man aber mit der natürlichen Edmonds-Karp Implementierung auflösen.

- (jetzt kam dann offensichtlich die Kür) Was können sie mir denn über die Beschreibung des Minimalen Spannbaums als Polyeder sagen?

Kurz erwähnt, was ein minimaler Spannbaum ist. Dann die Beschreibung der Spannbaumpolyeder von p. 35 aufgeschrieben und erklärt: Alle möglichen Partitionen müssen untereinander verbunden sein und die Gesamtzahl der Kanten darf für einen Spannbaum im zusammenhängenden Graphen nur |V|-1 sein.

- Wie sieht das LP dazu aus?

Das LP für den minimalen Spannbaum aufgeschrieben (Gleichung 4.9) und mit etwas Mühe und Unterstützung die Dualisierung aufgeschrieben.

- Wie sieht das mit der Ganzzahligkeit aus? Müssen wir die nicht explizit fordern?

Nein, die bekommen wir „geschenkt“ und können uns auf das relaxierte Problem beschränken. Das hat seinen Grund darin, dass die Inzidenzmatrix total unimodular ist und damit nur ganzzahlige Lösungen rauskommen.

- Wie ist denn der Zusammenhang von Spannbäumen zu Matroiden?
  Hochstättler hat nun noch angefangen aus |V| und |P| auf seinem Zettel zu schreiben und wollte im Endeffekt wissen, dass man damit den Rang der Inzidenzmatrix herausbekommen kann.

(Es hat offensichtlich ausgereicht an dieser Stelle nicht gleich beim Wort „Matroid“ hinten über zu kippen und noch zu erwähnen, dass die Spannbäume gerade die Basen der Inzidenzmatrix bilden und dass es sich ja auch bei Matroiden um Mengenfamilien mit einer Basis handelt. Das wurde dann nicht mehr vertieft.)

Insgesamt eine Herausforderung (das ganze Semester und die Prüfungsvorbereitung), aber mit einem sehr versöhnlichen Ende (der Prüfung selber). Die Prüfung fand in einer extrem angenehmen Atmosphäre statt und war aus meiner Sicht sehr fair aufgebaut. Dass sie dann doch recht dicht war und viele Aspekte drin waren ist mir jetzt erst beim Schreiben des Protokolls aufgefallen.

Hr. Hochstättler legt offensichtlich hauptsächlich Wert darauf zu sehen, dass man den Stoff verstanden hat. Sobald er den Eindruck hat, dass man versteht, was man sagt geht es auch schnell weiter. So musste ich kaum Algorithmen in tiefen Details erklären. Die Skizze reichte aus und schon ging es zum nächsten Thema. Ich hatte den Eindruck, dass es nichts schadet bestimmte Aspekte, Nebenbedingungen, Voraussetzungen und Zusatzinfos von sich aus zu erwähnen, auch wenn sie

[1] Ihr seht schon, es hat Alles in Allem eine gute Atmosphäre während der Prüfung geherrscht. Kein Platz für Nervosität, auch wenn die Konzentration natürlich hoch gehalten werden muss.

nicht explizit gefragt wurden. Es war auch kein größeres Problem bei der Formulierung der linearen Programme beim Aufschreiben mal zwei Indizes zu vertauschen oder ein Summenzeichen zu vergessen. Man sollte halt erklären, was man da warum macht.

Interessantes Fach, schöner Abschluss, aber ziemlich arbeitsintensiv.

Ich kann nur jedem/jeder empfehlen, zusätzlich zu Hochstättlers Buch parallel noch den „Ahuja: Network Flows“ (Verlag: Pearson Education India (1. März 2017), Sprache: Englisch, ISBN-10: 9332535159, ISBN-13: 978-9332535152, https://www.amazon.de/Network-Flows-Theory-Algorithms-Applications/dp/9332535159/ref=sr_1_cc_1?s=aps&ie=UTF8&qid=1521198648&sr=1-1-catcorr&keywords=9789332535152) zu lesen. Dort ist mehr Platz für erklärenden Text, ohne die Mathematische Korrektheit zu opfern.

Viel Erfolg und viel Spaß,

der Felix

Gedächtnisprotokoll

Kombinatorische Optimierung

Kurs 1216

Prüfer:	Prof. Hochstättler
Datum:	13.06.2017
Dauer:	25 Minuten
Studiengang:	Master Praktische Informatik
Note:	2.0

Erläutern Sie das Kürzeste-Wege-Problem.
Wir haben einen gerichteten Graphen $D = (V, A)$ mit einer Gewichtsfunktion $w\colon A \to \mathbb{Z}$ auf den Kanten. Wir wollen von einem beliebigen Startknoten s aus den bezüglich w den kürzesten Pfad zu einem Knoten t oder zu allen anderen Knoten bestimmen. Ergenis ist ein Spannbaum mit Knotenmarkierungen $dist(v)$, welche die Länge des Pfades von s nach v angibt.

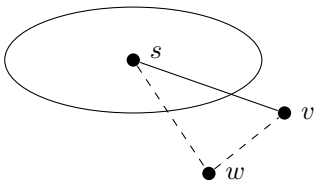
Welche Fälle sind dabei zu unterscheiden?
Beim Algorithmus von Dijsktra dürfen die Kantengewichte nicht negativ sein, weil er davon ausgeht, dass die Pfade monoton wachsen mit steigender Anzahl der enthaltenen Kanten. Bei Bellman-Ford dürfen die Kantengewichte auch negativ sein, aber es darf keine negativen Kreise geben.

Was passiert, wenn alle Kantengewichte -1 sind?
Das entspricht der Suche nach einem *längsten* Pfad im Graphen. Herr Hochstättler ergänzte, dass dies als Hamilton-Pfad-Problem bezeichnet wird und zur Komplexitätsklasse $\mathcal{NP}$ gehört. Alle Probleme im Kurs gehören zu $\mathcal{P}$, dies ist also ein Beispiel für ein „nicht gutartiges“ Problem.

Wie funktioniert nun der Dijkstra und wie ist seine Komplexität?
Ausgehend von s werden immer die kürzesten Kanten hinzugefügt. Die Komplexität wird vom Sortieren der Kanten nach Gewicht dominiert. Das geht in $O(|E| \log |E|)$, wenn man eine Priority Queue für die Implementierung verwendet.

Macht es einen Unterschied, ob ein Algorithmus $\log |E|$ oder $\log |V|$ hier als Faktor hat?
Nein, weil $\log |E| = \log |V|^2 = 2 \log |V|$ und der Koeffizient verschwindet im O.

Warum kann es keinen kürzeren Pfad geben, als der Dijkstra findet?
Herr Hochstättler zeichnete das Bild 5.1 von Seite 56:



Warum kann der Weg von s über w zu v nicht kürzer sein, als der Weg von s zu v ohne w? Induktionsbeweis über die Anzahl $|S|$ der besuchten Knoten: Da v und w die ersten Knoten außerhalb von S (die Ellipse im Bild) sind, muss $dist(w) \geq dist(v)$ gelten. Der Weg w–v hat aber eine Länge ≥ 0. Also hat der Weg über w entweder dieselbe Länge wie der s-v-Weg oder ist mindestens 1 länger.

Was passiert jetzt im Unterschied dazu beim Bellman-Ford?
Die Distanzlabels können öfter aktualisiert werden. Solange es noch Knoten gibt, bei denen $dist(w) > dist(v) + \ell(v, w)$ werden diese wieder in die Queue gestellt.

Was bedeutet das für die Komplexität?
Es gibt nur endlich viele Wege zu einem Knoten und bei jedem Schleifendurchlauf wird ein Distanzlabel

korrigiert. Deshalb terminiert der Algorithmus. Wir können also höchstens $O(|E|)$ viele Korrekturen machen und das jeweils für $O(|V|)$ viele Knoten. Also $O(|V||E|)$.

Beschreiben Sie das Problem der Maximalen Flüsse.
Wir haben wieder gerichteten Graphen $D = (V, A)$ mit einer Kapazitätsfunktion auf den Kanten. Wir suchen einen Fluss von einer Quelle s zu einer Senke t, der möglichst groß sein soll. Problem ist, dass wir Flüsse korrigieren müssen. Dafür führen wir Rückswärtskanten ein für jede Kante mit positivem Fluss, dadurch entsteht das Restnetzwerk. Beim Ford-Fulkeron suchen wir nun augmentierende s-t-Pfade, auf denen wir die Kanten um die minimale Restkapazität erhöhen können. Gibt es keine solchen Pfade, haben wir einen maximalen Fluss gefunden.

Warum liefert der Ford-Fulkerson den minimalen Schnitt gleich mit?
Wenn der Algorithmus terminiert, gibt es keinen Pfad mehr von s nach t. Die Knotenmenge kann also in S und W geteilt werden, wobei S alle Knoten sind, die noch von s aus erreichbar sind. Zwischen S und W gibt es im Restnetzwerk nur noch Rückwärtskanten, d. h. die zugehörigen Vorwärtskanten im Ausgangsgraphen sind „voll“. Das ist eine obere Schranke für den Fluss.

Wie ist die Komplexität und warum?
Wir können $O(|A||V|)$ viele Augmentierungen durchführen und das Suchen der kürzesten Pfade kann in $O(|A|)$ durchgeführt werden. Multipliziert ergibt sich $O(|A|^2|V|)$.

Sie haben damit die Variante von Edmonds-Karp beschrieben. Was ist das Besondere an dieser Implementierung im Vergleich zum ursprünglichen Ford-Fulkerson?
Die Suche der kürzesten s-t-Pfade erfolgt mittels Breitensuche.

Was ist der Unterschied zum Preflow-Relabel-Algorithmus von Goldberg und Tarjan?
Beim Ford-Fulkerson haben wir immer einen zulässigen Fluss, d. h. der Fluss erfüllt stets die Flusserhaltung für alle Knoten außer s und t. Er ist daher ein *primaler* Algorithmus. Der Preflow ist *dual*, weil er mit einem unzulässigen Fluss beginnt. Die Idee ist es, auf das Suchen der kürzesten s-t-Pfade zu verzichten. Wir schicken von s so viel Fluss wie möglich los. Dabei entsteht in den Knoten ein Überfluss. Diesen versuchen wir sukzessive *vorwärts* zu schicken. Was bedeutet also „vorwärts“? Ordnen die Knoten gedanklich in Höhen, wobei s an Höhe $|V|$ und t an Höhe 0 und die anderen Knoten dazwischen. Wenn ein Knoten Überfluss hat, betrachten wir die Nachbarn, die kleinere Höhe haben. Zu diesem können wir Fluss schicken.

Wie verändert sich dadurch die Komplexität im Vergleich zu Ford-Fulkeron?
Wir zählen die Anzahl der Relabel-Operationen und der saturierenden und nichtsaturierenden Pushes. Die Relabelings sind beschränkt duch $O(|V|^2)$ und die saturierenden Pushes analog zu Ford-Fulkerson durch $O(|A||V|)$. Für das Zählen der nichtsaturierenden Pushes bedienen wir uns einer Potenzialfunktion. Insgesamt haben wir damit $O(|V|^2|A|)$. Das Quadrat verschiebt sich also von der Anzahl der Kanten auf die Anzahl der Knoten.

Was für Algorithmen haben wir kennengelernt für das Minimale-Kosten-Fluss-Problem?
Wir haben unterschieden zwischen zwischen pseudo-polynomiellen und stark polynomiellen Algorithmen. Bei ersteren hängt die Laufzeit von den Werten der Kostenfunktion ab und nicht lediglich von der Länge der Codierung des Problems wie bei den stark polynomiellen Algorithmen.

Welchen stark polynomiellen Algorithmus gibt es?
Ich habe hier das Stichwort Min Mean Circuit Canceling genannt. Erklärt hat ihn dann Prof. Hochstättler selbst. Wahrscheinlich wollte er in Hinblick auf die Prüfungsdauer etwas Zeit herausholen. Ich habe nach seiner Erklärung dann doch noch das Prinzip erklärt, dass man negative Kreise ausnutzt, um die Kosten zu senken und man fertig ist, wenn es keine mehr gibt.

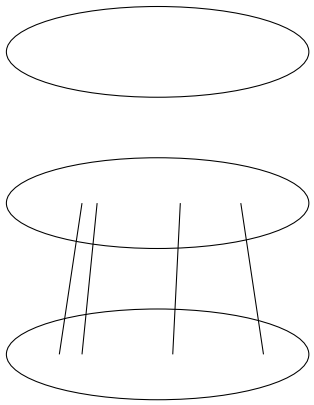
Kommen wir zum Thema Matchings. Was können Sie darüber sagen?
Ein Matching ist ein Teilgraph, in dem keine zwei Kanten gemeinsame Knoten haben. Wir suchen nun ein Matching, das möglichst groß ist. Dabei unterscheiden wir bipartite und nichtbipartite Graphen. Grundsätzlich suchen wir augmentierende Pfade; das sind alternierende Pfade, bei denen Anfangs- und Endpunkt jeweils ungematcht sind und die Kanten des Pfades zwischen gematchten und ungematchten Kanten alternieren. Eine Verbesserung des Matchings erhalten wir durch die symmetrische Differenz zwischen dem Pfad und dem bisherigen Matching. Der Fall bipartiter Graphen ist einfacher, weil hier ein alternierder Pfad auch zwischen den Farbklassen alterniert.

Wie ist die duale Interpretation des bipartiten Matchings?
Hier wusste ich nicht, worauf er hinaus wollte. Er half mir dann mit der Frage: Wissen Sie, wie Franz Beckenbauer auch genannt wird? Ja–der Kaiser. Was ist eine Stufe darunter? Der Satz von König! Die Größe eines maximalen Matchings ist gleich der Größe eines minimalen Vertex Covers. Herr Hochstättler fügte hinzu: Genau, das ist mit „dual“ gemeint. Es geht bei allen diesen Problemem immer um das Zusammenspiel von Maxima und Minima.

Warum geht das im nichtbipartiten Fall nicht so einfach?
Ein nichtbipartiter Graph kann ungerade Kreise enthalten, die wir auch als Blüten bezeichnet haben. Dadurch funktioniert das einfache Alternieren zwischen den Farbklassen nicht mehr.

Und was machen wir da? Was ist der Trick von Edmonds?
Wir kontrahieren die Blüte, suchen einen augmentierenden Pfad im kontrahierten Graphen und setzen diesen in der wieder expandierten Blüte fort. Das geht immer, weil wir die Matchingkanten in der Blüte so verschieben können, dass ein beliebiger Knoten in der Blüte exponiert ist. Dadurch können wir den augmentierenden Pfad in der Blüte fortsetzen, so wie wir es brauchen.

Beim bipartiten Fall gibt es den Zusammenhang zwischen Größe des Matchings und des Vertex Covers. Wie können wir eine solche Charakterisierung für den nichtbipartiten Fall bestimmen?
Mit der Tutte-Berge-Formel. Dazu verwenden wir die Gallai-Edmonds-Zerlegung als Vorüberlegung. Ich habe die Zerlegung an folgendem Bild erklärt:



C Menge der geraden Komponenten von $G \setminus A$

$A \subseteq V$, die „Barriere“

D Menge der ungeraden Komponenten von $G \setminus A$

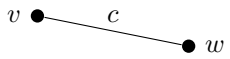
Die Knotenmenge V lässt sich zerlegen in eine beliebige Teilmenge $A \subseteq V$, eine Menge C von geraden Komponenten und eine Menge D von ungeraden Komponenten. Da jede ungerade Komponente $D_i \in D$ nur hypomatchable ist, d. h. ein Knoten muss exponiert sein, muss für ein maximales Matching je eine Kante von D_i nach A verlaufen. Nun zählen wir die Anzahl der Matchingkanten in A, C und D:

- Für A: $|A|$ wegen des Arguments mit den D_i.
- Für C: Jedes $C_i \in C$ ist perfekt matchbar, also $\frac{|C_i|}{2}$.
- Für D: Jedes $D_i \in D$ ist hypomatchbar, also $\frac{|D_i| - 1}{2}$.

Für C und D summieren wir nun auf und erhalten als Gesamtformel:

$$\text{Kardinalität maximales Matching} = \min_{A \subseteq V} \left\{ |A| + \sum_{C_i} \frac{|C_i|}{2} + \sum_{D_i} \frac{|D_i| - 1}{2} \right\}.$$

Können Sie die Bedeutung der dualen Variablen für das gewichtete Matching erklären?
Ich konnte nur den bipartiten Fall grob erklären. Bei gewichteten Matchings haben wir wieder Kantengewichte und suchen nun ein Matching maximalen Gewichts. Im bipartiten Fall ergibt sich für zwei Knoten folgendes Bild, welches ich aufgemalt habe:



Im primalen Programm sind die Entscheidungsvariablen die Kantengewichte c und im dualen Programm sind die Entscheidungsvariablen die „Knotengewichte“ für v und w, wobei $v + w \leq c$ gelten muss. Wir verbessern nun die duale Lösung solange, bis $v + w = c$. Dann haben wir die optimale Lösung gefunden. Da ich den nichtbipartiten Fall nicht erklären konnte, hat Herr Hochstättler das vorgerechnet. Ich kann die Antwort aber nicht mehr wiedergeben.

Allgemeine Bemerkungen:
Das Prüfungsgespräch verlief in einer sehr freundlichen und entspannten Atmosphäre. Es erinnerte teilweise eher an ein Fachgespräch als an ein strenges Frage-Antwort-Spiel. Ich war ein paar Minuten zu früh da, so dass wir uns ca. 5–10 Minuten noch unterhalten haben. Dabei fragte ich Herrn Hochstättler, warum im Buch von „gutartigen“ Problemen die Rede ist und was denn eigentlich „bösartige“ Probleme sind. Er ging dann auf das $\mathcal{P} = \mathcal{NP}$-Problem ein und meinte: Ich stelle Ihnen nachher, wenn die Prüfung losgeht, eine Frage, da müssen Sie mit „$\mathcal{NP}$-vollständig“ anworten. (Das war die Frage nach der Komplexität des Hamilton-Pfades.)

Wir sind durch die einzelnen Themen recht zügig durch. Wenn Herr Hochstättler das Gefühl hat, dass man einen Algorithmus verstanden hat, unterbricht er oft die Erklärung und führt sie selbst zu Ende; wahrscheinlich um etwas Zeit zu sparen. Das wird aber nicht negativ bewertet. Weiß man etwas nicht auf Anhieb, hilft er auch mit Nachfragen und lässt zu, dass man seine Antwort nochmal neu formuliert.

Bei Mathematikern, so Herr Hochstättler, nimmt die Prüfung eine etwas andere „Geschmacksrichtung“ an. Bei Informatikern berücksichtigt er, dass diese in der Regel weniger Übung mit abstrakten, algebraischen Problemen und den Grundlagen der Linearen Algebra und Polyeder-Geometrie haben. Er hat mir auch keine allgemeinen Frage hierzu gestellt, sondern alle Fragen bezogen sich auf ein konkretes im Kurs behandeltes Problem oder ein konkretes Lösungsverfahren. Man muss keine Algorithmen oder Beweise im Detail aufschreiben können.

Für die Prüfungsvorbereitung sollte man sich darauf konzentrieren, für alle Probleme erklären zu können: Was sind die Bedingungen und was wird maximiert bzw. minimiert? Welche Algorithmen gibt es dafür? Was ist die Methode oder der Trick, mit dem der Algorithmus arbeitet? Welche Laufzeitkomplexität hat ein Algorithmus und wie kommt man darauf? (Welche Einzeloperationen werden gezählt und wieviele muss man machen?) Gibt es mehrere Algorithmen, welcher ist besser und warum? Bei primal-dualen Verfahren, inwiefern liefert ein Algorithmus auch die duale Lösung gleich mit?

Zusammenfassend möchte ich sagen, wer wegen seiner Kurswahl ein Modul aus dem Katalog M1 belegen muss, dem kann ich diesen Kurs und den Prüfer durchaus empfehlen. Bei der Kursbearbeitung sollte man unbedingt die Newsgroup zum Fragen benutzen, diese wird vom Lehrgebiet sehr gut betreuut.

Erinnerungsprotokoll

Mündliche Prüfung Kurs 01216 „Kombinatorische Optimierung“

30 Minuten, 07.04.2015, Prof. Hochstättler

- Unterschied zwischen Minimum Spanning Tree und Shortest Path Tree?
  Der MST versucht die Gesamtsumme der Kantengewichte zu minimieren, der andere die Wege zu den Kanten,
- Welcher ist einfacher? Und Warum?
  Der MST. Die Anzahl der Kanten ist durch die Knotenzahl vorgegeben und damit können negative Kreise anhand eines konstanten Summanden auf der Gewichtsfunktion behandelt werden. Beim anderen geht das nicht.
- Wie findet man negative Kreise?
  Ausgehend vom Bellmann-Ford Algorithmus wird gesucht. Idee ist, dass dieser Algorithmus die Knoten abhängig von der Anzahl Kanten zum Startknoten sortiert in eine Queue stellt. Immer bei einer Grenze dieser Anzahl in der Queue verschiebt der Suchalgorithmus von der NOW in die NEXT Queue.
- Nächstes Thema: Dualität. Was ist der Unterschied zwischen Ford-Fulkerson und Preflow Push?
  Ford-Fulkerson ist ein Primaler Algorithmus. Er hat jederzeit einen zulässigen Fluss. Preflow Push ist ein Dualer.
- Was ist die zentrale Idee des Ford-Fulkerson?
  Das ein residual Network gespeichert und darin nach dem Shortest Path gesucht wird. So lassen sich Fehler in den ersten Durchläufen wieder korrigieren.
- Was ist die große Schwierigkeit beim Preflow Push? Und wie löst man die?
  Saturating und non saturating Push. Die Lösung ergibt sich über das Potential, das gespeichert wird.
- Nächstes Thema: Was ist ein Matching? Und was ist die zentrale Idee im Algorithmus es zu finden?
  Ein Matching verbindet Knoten miteinander. Dabei unterscheidet maximal, perfekt, …. Die Zentrale Idee ist der Augmenting Path.
- Woher weiß man, ob ein Matching maximal ist?
  Mit Hilfe des Vertex Covers. Satz von König.
- Warum liefert der Algorithmus diesen Maximalitätsbeweis gleich mit?
  Zwei Klassen von Knoten. Auf der einen Seite die ungematchten und auf der anderen Seite die nicht zu erreichenden. Zusammen zu minimieren. Das ist auch ein Ergebnis des Algorithmus.
- Was ist das Problem, wenn der auf nicht-bipartite Graphen angewendet wird? Und wie kann das gelöst werden?
  Es kann zu Problemen bei Kreisen ungerader Länge kommen. Blossoms sind hypermatchable Teilgraphen. Diese können wie ein ungematchter Knoten betrachtet werden.
- Was ist hypermatchable?
  Ein Graph, der zwar mehr als ein mögliches, maximales Matching hat, aber immer mit einem ungematchtem Knoten.

Es ging dem Prüfer nicht um die Algorithmen oder Beweise, sondern um das Warum? Nur Algorithmen auswendig lernen ist (Zitat) „Techniker Niveau“. Man soll ja später Algorithmen entwickeln, die es halt noch nicht gibt und Erfahrung darin haben, die Korrektheit und Komplexität abzuschätzen. Das ging bei meiner Prüfungsvorbereitung völlig an mir vorbei. Daher ist es auch nur gerade so etwa geworden. Der Prof. redet (mein Eindruck) schnell und mir war meist nicht so direkt

klar, worauf er hinaus wollte. Ich hatte dann mit einer Antwort angefangen, wurde aber abgebrochen bevor das Stichwort kam, das er hören wollte. Obwohl ich das Stichwort wusste und hätte aufschreiben können.

Im Protokoll fehlen noch einige Stichpunkte, die er ansprach, zu denen ich aber nichts Genaueres sagen konnte. Zum Beispiel Tutte-Berge.