Algorithmen für den Entwurf virtueller privater Netzwerke

Diplomarbeit
von

Thomas Rothvoß

Nach einem Thema von
Prof. Dr. Friedrich Eisenbrand

Zweitgutachter
PD Dr. Piotr Krysta

Fachbereich Informatik
Universität Dortmund

Eidesstattliche Erklärung

„Hiermit versichere ich an Eides statt, dass ich diese Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen angefertigt habe."

Essen, den 12. Oktober 2006

Thomas Rothvoß

Einleitung

Im Netzwerk-Design geht es darum, in einem gewichteten Graphen genügend Kapazitäten zu reservieren, um sichere Kommunikation zu gewährleisten. Viele interessante Netzwerk-Design Probleme sind $\mathcal{NP}$-schwer. In dieser Diplomarbeit widmen wir uns unter anderm dem $\mathcal{NP}$-schweren Problem des *Virtual-Private-Network-Design* (VPND), welches in der letzten Zeit sehr viel Beachtung in der Literatur gefunden hat.

Das VPND-Problem ist motiviert durch die Tatsache, dass aufkommender Fluss oder die zu bewältigenden Verkehrsszenarien nicht vorhersagbar sind. Das ist insbesondere im Internet der Fall, bei dem autonome Terminale miteinander kommunizieren. Daher schafft man Modelle, in denen ein Fluss gültig ist, wenn gewisse Schranken der Flusswerte an den Terminalen eingehalten werden. Im VPND-Modell sind jedem Terminal zwei Werte zugeordnet. Einmal eine obere Schranke für den Flussbetrag den es senden und eine obere Schranke für den Betrag den es empfangen kann. Es ist leicht einzusehen, dass man dieses Problem darauf reduzieren kann, nur zwei Sorten von Terminalen zu erlauben. Einmal Sender, die eine Flusseinheit senden, aber keinen Fluss empfangen können und Empfänger, die nur eine Einheit empfangen können. Der Entwurf von virtuellen privaten Netzwerken ist Gegenstand dieser Arbeit.

Wenn wir uns beim VPND als Ziel setzen, die Kosten des Netzwerkes zu minimieren, so erhalten wir eine Verallgemeinerung des Steinerbaum-Problems, welche von diesem direkt die $\mathcal{NP}$-Schwierigkeit erbt. Allerdings implizierte die bisher bekannte Reduktion vom Steinerbaum-Problem auf VPND dabei stets, dass die resultierende VPND-Instanz nur einen Sender bzw. einen Empfänger enthält. Interessanterweise können wir zeigen, dass wenn die Zahl der Sender annähernd mit der der Empfänger übereinstimmt, das VPND-Problem immer noch $\mathcal{NP}$-schwierig ist (Kapitel 4.2), während die Baumvariante von VPND in polynomieller Zeit lösbar ist (Kapitel 5.2).

In Kapitel 5 arbeiten wir die Struktur der Baumvariante des VPND-Problems heraus, was uns erlaubt, eine Reihe von neuen Ergebnissen zu erzielen, angefangen davon, dass wir zeigen können, dass der günstigste Shortest-Path-Tree nicht nur eine $1 + |R|/|S$ Approximation liefert [5], sondern bzgl. der optimalen Baumlösung sogar eine $\frac{|R|+|S|}{2|S|}$ Approximation, wobei S und R die Knoten angeben, die je eine Einheit senden bzw. empfangen und wir $|S| \leq |R|$ voraussetzen. Wir geben auch eine Beispielinstanz an, die zeigt, dass diese Schranke scharf ist. Danach zeigen wir, dass im Fall $|R| = |S| \pm O(1)$ eine optimale Baumlösung effizient berechnet werden kann. Bisher war dies nur für $|R| = |S|$ bekannt [10]. Für den Fall $|R| = O(1) \cdot |S|$ sind wir immerhin in der Lage ein PTAS anzugeben.

Für das *Connected-Facility-Location-Problem* (kurz CFL), welches auch *Rent-or-Buy-Problem* genannt wird, ist es die Aufgabe, ausgewählte Knoten eines Graphen kostenminimal zu verbinden, wobei man für jede Kante die Wahl hat, pauschal oder pro benötigter

Kapazitätseinheit zu bezahlen. CFL ist mittlerweile ein klassisches Problem der jüngeren Netzwerk-Design-Literatur. Der bislang beste Algorithmus für CFL stammt von Gupta, Kumar und Roughgarden [8] und besteht daraus, auf einer zufällig gewählten Teilmenge der Knoten einen Steinerbaum aufzuspannen und diesen als Backbone des Netzwerkes zu verwenden. Dieses simple Vorgehen führte zusammen mit einer trickreichen Analyse zum bis dato besten Approximationsalgorithmus mit einer Güte von 3.55. In dieser Arbeit können wir durch ein Anpassen der Auswahlwahrscheinlichkeit und durch eine neue Analysemethode eine Güte von 3.05 für diesen Algorithmus nachweisen, womit wir die bislang beste bekannte Güte für CFL auf diesen Wert verbessern. Wir geben auch Worst-Case Instanzen an, die eine untere Schranke von 2.36 für die Güte dieses Algorithmus liefern. Trotz der zu VPND vollkommen verschiedenen Definition, werden wir zeigen können, dass das CFL-Problem im Wesentlichen mit der Baumvariante von VPND übereinstimmt. Daher lässt sich die erzielte Güte von 3.05 auch auf VPND übertragen, sofern man als Lösungen nur Bäume zulässt.

Das *Maybecast-Problem* stellt ebenfalls eine Verallgemeinerung des Steinerbaum-Problems dar und besteht darin, Knoten eines Graphen kostenminimal mit einer Wurzel zu verbinden, wobei für jeden Knoten eine bestimmte Wahrscheinlichkeit gegeben ist, dass die Verbindung bezahlt werden muss. Die erste konstante Approximationsgüte für das Maybecast-Problem betrug 40.7 und wurde in [12] nachgewiesen. Wir werden das Maybecast-Problem auf CFL reduzieren und mit der in Kapitel 2.1 gezeigten Güte für CFL eine Güte von 4.83 für das Maybecast-Problem zeigen, was die bislang beste bekannte Güte für dieses Problem darstellt.

Für die Variante des VPND-Problems, in der die Last auf den Kanten minimiert werden soll, sind bislang weder Approximationsalgorithmen mit nichttrivialen Güten bekannt, noch Nichtapproximierbarkeitsbeweise. Wir werden in der Lage sein zu zeigen, dass dieses Problem einen Integrality Gap von mindestens $\Omega(\sqrt{n}/\log n)$ bzgl. seiner natürlichen Relaxation aufweist, wobei n die Anzahl der Knoten im Graphen angibt. Damit liefern wir ein erstes Indiz für die Nichtapproximierbarkeit dieses Problems. Zusätzlich wollen wir noch aufzeigen, dass randomisiertes Runden einer optimalen, relaxierten Lösung nicht einmal die bescheidene Güte von $o(n)$ liefern kann.

Wir beginnen in Kapitel 1 mit der formalen Definition der Probleme, die in dieser Diplomarbeit thematisiert werden. In Kapitel 2 behandeln wir dann das Connected-Facility-Location-Problem, welches sich als zu VPND verwandt herausstellen wird. In Kapitel 3 wenden wir Erkenntnisse über das Connected-Facility-Location-Problem auf das Maybecast-Problem an. In den darauffolgenden drei Kapitel widmen wir uns dann drei Varianten des VPND-Problems:

- Der uneingeschränkten Variante
- Der Variante in der wir uns auf Baumlösungen beschränken
- Der Variante in der nicht die Kosten, sondern die maximale Last auf einer Kante minimiert wird

Im letzten Kapitel ziehen wir dann ein Resümee und zeigen unbeantwortete Fragen auf, die es in Zukunft noch zu lösen gilt.

Inhaltsverzeichnis

1 Grundlagen

Zunächst sollten wir uns klarmachen, worüber wir reden. Daher zunächst einige Problemdefinitionen.

Def. Steinerbaum-Problem. Beim Steinerbaum-Problem ist ein ungerichteter Graph $G = (V, E)$ und eine Kostenfunktion $c : E \to \mathbb{R}^+$ gegeben, sowie eine ausgezeichnete Teilmenge $R \subseteq V$ der Knoten, die man als *Terminale* bezeichnet. Gesucht ist nun ein Baum $T \subseteq E$, der alle Terminale aufspannt und zugleich die Kosten $c(T) = \sum_{e \in T} c(e)$ minimiert.

Eine praktische Anwendung kann man sich so vorstellen, dass eine vorgegebene Menge R von Orten auf der Landkarte durch Straßen auf möglichst kostengünstige Weise verbunden werden sollen. Es gibt zahllose weitere Anwendungen des Steinerbaum-Problems. Für uns wird entscheidend sein, dass dieses Problem häufig als Teilproblem in anderen Optimierungsproblemen auftaucht. Für unsere Zwecke sind daher die folgenden Eigenschaften relevant:

- Das Steinerbaum-Problem ist $\mathcal{NP}$-schwierig. Der bis zu diesem Zeitpunkt beste bekannte Approximationsalgorithmus weist eine Güte von $1 + \frac{\ln(3)}{2} \leq 1.55$ auf [16].
- Da die APX-Vollständigkeit für dieses Problem in [1] nachgewiesen wurde, gibt es kein PTAS für das Steinerbaum-Problem, sofern $\mathcal{NP} \neq \mathcal{P}$.
- Für $|R| = O(\log |V|)$ ist das Steinerbaum-Problem mittels dynamischer Programmierung in polynomieller Zeit lösbar [3].

Nun stellen wir ein Problem vor, welches das Steinerbaum-Problem als Unterproblem enthält.

Def. Connected-Facility-Location-Problem (CFL). Beim CFL-Problem ist ein ungerichteter Graph $G = (V, E)$ mit einer Kostenfunktion $c : E \to \mathbb{R}^+$ gegeben, sowie ein Parameter $M \geq 1$ und eine ausgezeichnete Teilmenge $D \subseteq V$ der Knoten, die man als *Demands* bezeichnet. Gesucht ist nun eine Auswahl der Knoten $F \subseteq V$, die wir im weiteren *Facilities* nennen werden, und ein Baum $T \subseteq E$ der die Facilities aufspannt, so dass folgende Kostenfunktion minimiert wird

$$M \cdot c(T) + \sum_{d \in D} \ell(d, F)$$

Dabei definieren wir $\ell(v, u)$ für zwei Knoten $v, u \in V$ als die Länge des kürzesten Pfades von v nach u (bzgl. Kostenfunktion c) in G. Für eine Knotenmenge $U \subseteq V$ setzen wir $\ell(v, U) := \min_{u \in U}\{\ell(v, u)\}$ als Entfernung von v zum nächsten Knoten der Menge U.

Der beim CFL-Problem gesuchte Baum T, der alle Facilities F aufspannt, ist nichts anderes als ein Steinerbaum auf Terminalen F. Das CFL-Problem wird, wenn eine Facility r fest vorgegeben ist, auch als *Rent-or-Buy-Problem* bezeichnet, denn in diesem Fall lässt es sich so sehen, dass von vorgegebenen Orten D je eine Flusseinheit über Kabel zu r gesendet werden muss. Wir haben für jede Kante die Wahl entweder Kabel zu mieten, wobei wir dann pro Flusseinheit bezahlen müssen oder das Kabel für das M-fache des Mietpreises zu kaufen, so dass wir dann unbeschränkte Kapazitäten über die Kante laufen lassen können.

Beim Steinerbaum-Problem kann man ein beliebiges Terminal als Wurzel r deklarieren und dann das Steinerbaum-Problem so interpretieren, dass eine kostenminimale Verbindung aller Terminale mit Wurzel r gesucht ist. Das *Maybecast-Problem* [12] verallgemeinert dies in dem Sinne, dass jedes Terminal i nur mit einer bestimmten Wahrscheinlichkeit p_i mit r verbunden werden muss und die erwarteten Verbindungskosten minimiert werden sollen. Formal lautet das Problem wie folgt.

Def. Maybecast-Problem. Gegeben ist ein ungerichteter Graph $G = (V, E)$ mit nichtnegativen Kantenkosten $c : E \to \mathbb{R}^+$, eine Wurzel $r \in V$ und Terminale $R \subseteq V$, wobei jedes Terminal $i \in R$ mit einer Wahrscheinlichkeit von $p_i \in \mathbb{Q} \cap [0, 1]$ mit r verbunden werden muss. Gesucht sind Pfade P_i von i nach r für jedes Terminal $i \in R$. Im Fall, dass ein Terminal zu r verbunden werden muss, werden alle Kanten auf P_i aktiv. Minimiert werden sollen die erwarteten Kosten der aktiven Kanten.

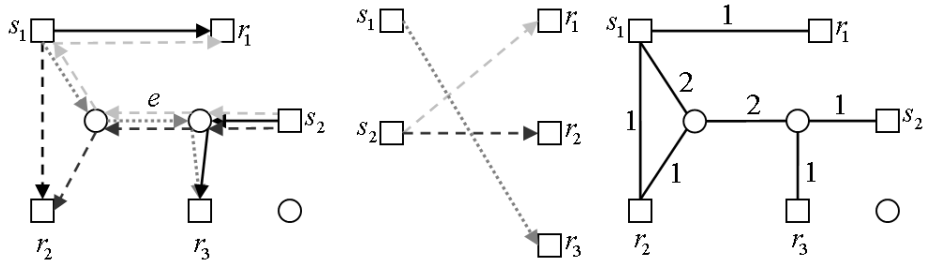
Definieren wir nun das sogenannte *Virtual-Private-Network-Design-Problem.*

Def. VPND-Problem. Beim VPND-Problem ist wieder ein ungerichteter Graph $G = (V, E)$ und eine Kostenfunktion $c : E \to \mathbb{R}^+$ gegeben. Es gibt zwei ausgezeichnete Mengen von Knoten, die Sender $S \subseteq V$ und die Empfänger $R \subseteq V$, welche beide disjunkt sind. Gesucht sind nun Pfade $\mathcal{P} = \{P_{sr} | s \in S, r \in R, P_{sr} \text{ ist } s\text{-}r\text{-Pfad}\}$ von jedem Sender zu jedem Empfänger und Kantenkapazitäten $u : E \to \mathbb{N}_0$, so dass für jedes zulässige Verkehrsszenario die reservierte Kantenkapazität u ausreichend ist und die Kosten

$$\sum_{e \in E} c(e)u(e)$$

minimiert werden. Ein zulässiges Verkehrsszenario besteht dabei aus einer Auswahl von Pfaden $\mathcal{P}' \subseteq \mathcal{P}$, wobei von jedem Sender und jedem Empfänger höchstens ein ein- bzw. ausgehender Pfad gewählt werden darf. Die Kapazität ist dabei auf einer Kante e ausreichend für ein Verkehrsszenario $\mathcal{P}'$, wenn $u(e) \geq |\{P \in \mathcal{P}' | e \in P\}|$, also die Anzahl der Pfade im Verkehrsszenario, die über e gehen, höchstens der Kapazität auf e entsprechen.

Beim VPND-Problem kann man die Mengen der Sender und Empfänger vertauschen, ohne dass sich der Wert der Optimallösung ändert. Daher können wir o.B.d.A. $|S| \leq |R|$ annehmen. Dies sei im weiteren Verlauf dieser Arbeit stets vorausgesetzt.



(a) Pfade (b) Graph G_e (c) Kapazitäten

Abbildung 1.1: Beispiel zur Kapazitätsreservierung für Sender s_1, s_2 und Empfänger r_1, r_2, r_3. Terminale (Sender oder Empfänger) werden als Quadrate dargestellt, übrige Knoten als Kreise. In (a) werden die gewählten Pfade dargestellt, in (b) der bipartite Graph G_e und in (c) die insgesamt resultierenden Kapazitäten.

Als Terminale werden wir beim VPND-Problem die Vereinigung der Sender und Empfänger verstehen.

Als Motivation für dieses Optimierungsproblem kann man sich das Problem vorstellen, dass man ein Netz von Rechnern hat, indem einige Rechner Daten senden und andere Daten empfangen. Nun müssen genug Kapazitäten reserviert werden, so dass die Rechner untereinander kommunizieren können. Man weiß allerdings vorher nicht, welcher Sender zu welchem Empfänger sendet. Insofern ist dieses Problem durchaus realistisch.

In der Literatur wird dieses Problem auch als die *asymmetrische* Variante bezeichnet. Es gibt auch noch die *symmetrische* Variante [7], in der es keine Unterscheidung in Sender und Empfänger gibt, sondern jedes Terminal sowohl senden, als auch empfangen kann. Für die symmetrische Variante ist eine optimale Baumlösung effizient berechenbar, welche höchstens doppelt so teuer ist, wir eine optimale allgemeine Lösung [7]. Es wird sogar vermutet, dass für die symmetrische Variante immer eine optimale Lösung existiert, die einen Baum formt. Dies konnte bislang jedoch nur für Ringgraphen bewiesen werden [9].

Ein entscheidender Punkt in der Definition des VPND-Problems ist, dass wir fordern, dass *jedes* zulässige Verkehrsszenario durch die Kapazitäten gedeckt sein muss. Nun muss dies doch in polynomieller Zeit überprüfbar sein, ansonsten könnten wir das Problem nicht als wohldefiniertes Optimierungsproblem ansehen[1]. Machen wir uns die Situation an einem Beispiel klar. Betrachten wir die VPND-Beispielinstanz in Abbildung 1.1 (a). Nehmen wir an, wir haben uns für die (wohl kaum optimalen) eingezeichneten Pfade entschieden. Versuchen wir zu bestimmen, wieviel Kapazität wir auf die Kante e legen müssen. Dazu konstruieren wir einen bipartiten Graphen $G_e = (S \cup R, E_e)$ mit den Sendern auf der einen Seite und den Empfängern auf der anderen. Wir fügen eine Kante (s, r) mit $s \in S$ und $r \in R$ genau dann ein, wenn der Pfad von Sender s zu Empfänger

[1] siehe Def. der Komplexitätsklasse $\mathcal{NPO}$ in [19]

r über die Kante e verläuft (siehe Abbildung 1.1 (b)). Sei $\mathcal{M} \subseteq E_e$ nun ein Matching auf G_e. Dann bedeutet dies, dass jeder Sender und jeder Empfänger mit höchstens einer Kante aus $\mathcal{M}$ inzident ist. Das Matching $\mathcal{M}$ entspricht also einem zulässigen Verkehrsszenario, in welchem über die Kante e genau $|\mathcal{M}|$ viele Pfade verlaufen. Wir wissen also, dass für die zu reservierende Kapazität $u(e) \geq |\mathcal{M}|$ gelten muss. Als Schlussfolgerung erhalten wir, dass auf der Kante e der Wert des *maximalen* Matchings im Graphen G_e als Kapazität reserviert werden muss. Da ein maximales Matching aber effizient berechnet werden kann, können auch die zu reservierenden Kapazitäten effizient berechnet bzw. verifiziert werden.

In unserem Beispiel in Abbildung 1.1 bestünde ein (nichteindeutiges) maximales Matching aus den Kanten (s_1, r_3) und (s_2, r_2), wir müssten auf e also 2 Kapazitätseinheiten reservieren. Wenn wir dieses Verfahren analog für alle übrigen Kanten des Graphen G durchführen, erhalten wir die in Abbildung 1.1 (c) eingezeichneten Kapazitäten. Wir sehen, dass die Kanten, auf denen wir echt positive Kapazität reservieren müssen, einen Kreis enthalten, also keinen Baum formen. Im Netzwerk-Design sind aber Lösungen, in denen alle Pfade auf einem Baum verlaufen, von besonderer Bedeutung. Daher definieren wir nun noch die sogenannte VPND-Tree-Variante des VPND-Problems.

Def. VPND-Tree-Problem. Ein- und Ausgabe sind identisch zum VPND-Problem, aber es sind nur Lösungen zulässig, in denen alle Pfade auf einem Baum verlaufen.

Nun können wir uns daran erinnern, dass der Pfad auf einem Baum zwischen zwei Knoten bereits durch den Baum eindeutig bestimmt ist. Also können wir bei der VPND-Tree-Variante anstatt einer Menge von Pfaden auch einfach den durch die Pfade induzierten Baum als Lösung ausgeben. Auch für die Kapazitäten ergeben sich gravierende Vereinfachungen, wenn alle Pfade über einen Baum verlaufen. Doch mehr dazu in Kapitel 5.

Die einfachste und intuitivste Baumlösung besteht sicherlich darin, einen beliebigen Knoten des Graphen als Wurzel zu wählen und von diesem ausgehend die Kanten, die auf kürzesten Wegen von der Wurzel zu den Sendern und Empfängern liegen, in die Lösung aufzunehmen. Den resultierenden Baum bezeichnet man auch als Shortest-Path-Tree bezüglich der ausgewählten Wurzel. Von besonderem Interesse ist natürlich der Shortest-Path-Tree der von allen $|V|$ vielen möglichen mit verschiedenen Wurzeln am günstigsten ist.

In der Literatur wird zumeist auch das folgende verallgemeinerte VPND-Problem definiert [4, 5, 7, 8, 10].

Def. Verallgemeinertes VPND-Problem. Es sei ein ungerichteter Graph $G = (V, E)$ und eine Kostenfunktion $c : E \to \mathbb{R}^+$ gegeben. Für jeden Knoten $v \in V$ sind Werte $b^+(v)$ und $b^-(v)$ gegeben, die angeben, wieviele Einheiten v senden und empfangen kann. Es sind Pfade zwischen jedem Knotenpaar und Kantenkapazitäten $u : E \to \mathbb{N}_0$ gesucht, so dass jedes zulässige Verkehrsszenario von u unterstützt wird, wobei ein zulässiges Verkehrsszenario durch eine Matrix $A : V \times V \to \mathbb{N}_0$ definiert wird, für die $\sum_{u \in V, u \neq v} A(u, v) \leq b^-(v)$ und $\sum_{u \in V, u \neq v} A(v, u) \leq b^+(v)$ gilt.

Dieses verallgemeinerte Problem (von dem man wieder analog eine Baumvariante definieren kann) lässt sich aber leicht auf das gewöhnliche VPND-Problem reduzieren. Dazu muss man einfach jeden Knoten v durch $b^+(v)$ viele Sender und $b^-(v)$ viele Empfänger ersetzen. Auch für Fälle, in denen die Werte $b^+(v)$ bzw. $b^-(v)$ superpolynomiell sind, lassen sich alle vorgestellten Algorithmen so implementieren, dass die Laufzeit polynomiell bleibt.

Bei vielen Flussproblemen kann man sich neben dem Minimieren der Kosten auch eine Variante vorstellen, in der die maximale Kapazität auf einer Kante minimiert werden soll [15]. Daher wollen wir nun das folgende Problem definieren.

Def. VPND-MinCongestion-Problem. Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit Sendern $S \subseteq V$ und Empfängern $R \subseteq V$ (mit $S \cap R = \emptyset$). Gesucht sind Pfade $\mathcal{P} = \{P_{sr} | s \in S, r \in R, P_{sr}$ ist s-r-Pfad$\}$ von jedem Sender zu jedem Empfänger und Kantenkapazitäten $u : E \to \mathbb{N}_0$ so dass für jedes zulässige Verkehrsszenario die reservierte Kantenkapazität u ausreichend ist und $\max\{u(e) | e \in E\}$ minimiert wird.

2 Das Connected-Facility-Location-Problem

Historie und neue Resultate

Beim Connected-Facility-Location-Problem müssen in einem Graphen $G = (V, E)$ ein Baum T und eine Menge $F \subseteq V$ gesucht werden, so dass T die Menge F aufspannt und $M \cdot c(T) + \sum_{d \in D} \ell(v, F)$ minimiert wird. Die verfügbare Approximationsgüte ist im Laufe der Zeit von 9.002 [7] auf 4.55 [17] und schließlich auf 3.55 [8] gesenkt worden. Wir werden in diesem Kapitel mit einer neuen Analyse zeigen, dass man den Algorithmus von [8] sogar auf eine Güte von 3.05 verbessern kann.

Wenn wir eine Facility $r \in V$ fest vorgegeben haben, können wir das CFL-Problem so interpretieren, dass jeder Demand einen Fluss von 1 auf einem Baum zu r sendet. Sei k die Anzahl der Flusseinheiten über eine Kante e, dann verursacht e im Fall $k \leq M$ Kosten von $k \cdot c(e)$. Für $k > M$ wird es günstiger, die Kante zu T hinzuzufügen und wir bezahlen $M \cdot c(e)$ für diese Kante. Eine alternative Interpretation sieht daher so aus, dass man sich entscheiden kann, ob man eine Kante mietet und $c(e)$ pro Flusseinheit bezahlt oder die Kante pauschal für $M \cdot c(e)$ kauft. Daher nennt man diese Variante auch *Rent-or-Buy-Problem*. Die Kosten, die abhängig vom Fluss über eine Kante bezahlt werden müssen, sind dabei stückweise linear mit 2 verschiedenen Steigungen. Daher ist das Rent-or-Buy-Problem ein Spezialfall des *Single-Sink-Buy-At-Bulk-Problems* [8].

Eine Verallgemeinerung des CFL-Problems sieht zusätzliche Kosten f_i für alle $i \in F$ vor und schränkt die Facilities auf eine Teilmenge der Knoten ein. Für dieses Problem ist 8.55 die momentan beste bekannte Approximationsgüte [17].

2.1 Der Sampling-Algorithmus

Algorithmus 1 ist der von [8] vorgestellte, bisher beste Approximationsalgorithmus für das CFL-Problem. Er wählt jeden Demand mit einer Wahrscheinlichkeit von $1/M$ (also $q = 1$) als Facility und spannt auf diesen ausgewählten Demands einen Baum T auf. Dabei nimmt man an, einen Knoten r^* zu kennen, der in der optimalen Lösung als Facility gewählt ist. Erinnern wir uns daran, dass

$$M \cdot c(T) + \sum_{d \in D} \ell(d, F)$$

die Kosten einer Lösung (F, T) sind. Wir bezeichnen $c(T)$ als *Steinerbaumkosten* und $\sum_{d \in D} \ell(d, F)$ als *Verbindungskosten*. Es sei im weiteren Verlauf (F^*, T^*) eine beliebige optimale Lösung mit Wert $OPT = M \cdot S^* + C^*$, wobei $S^* = c(T^*)$ bzw. $C^* = \sum_{d \in D} \ell(d, F^*)$ die Steinerbaum- bzw. Verbindungskosten der optimalen Lösung (F^*, T^*)

Algorithmus 1 Sampling-Algorithmus

Eingabe: Gewichteter Graph $G = (V, E)$; Demands $D \subseteq V$; Parameter $M \geq 1$

Ausgabe: Facilities F; Baum T

1. Rate eine Facility $r^* \in F^*$
2. Markiere jeden Demand $d \in D$ mit Wahrscheinlichkeit q/M
3. Setze $F := \{\text{markierte Demands}\} \cup \{r^*\}$
4. Berechne ρ-approximativen Steinerbaum T auf Terminalen F
5. Gebe (F, T) aus

darstellen. Der Faktor ρ bezeichnet dabei den Faktor mit dem sich das Steinerbaum-Problem approximieren lässt. Zur Zeit ist $\rho \leq 1.55$ die beste verfügbare Schranke [16].

Die Analyse von Gupta et al.

Die Autoren zeigen in [8], dass man bei Sampling-Wahrscheinlichkeit $1/M$ die Kosten eines optimalen Steinerbaumes, der F aufspannt, durch OPT abschätzen kann. Durch Approximation mit Faktor ρ erhält man dann $E[c(T)] \leq \rho \cdot OPT$. Mit einer innovativen Analyse zeigen die Autoren dann, dass die Verbindungskosten in der optimalen Lösung geringer sind als das M-fache der Kosten desjenigen Steinerbaumes auf F, der durch die MST-Heuristik entsteht. Somit kann man die Verbindungskosten durch $2 \cdot OPT$ abschätzen. Insgesamt können Gupta, Kumar und Roughgarden [8] so eine Güte von $\rho + 2 \leq 3.55$ zeigen.

Wir werden für den Sampling-Algorithmus eine erwartete Güte von 3.05 nachweisen, sofern man $q = 0.67$ wählt. Um dieses Ergebnis mit dem in [8] erzielten vergleichen zu können, werden wir zunächst die Analyse von Gupta et al. für eine allgemeine Sampling-Wahrscheinlichkeit von q/M durchführen. Die Abschätzung der Steinerbaumkosten werden wir dabei für unsere Analyse übernehmen, die Verbindungskosten werden wir aber auf eine neue Art und Weise abschätzen. Beginnen wir mit der Analyse der Steinerbaumkosten.

Lemma 2.1. *[8] Die Steinerbaumkosten lassen sich abschätzen durch*

$$E[c(T)] \leq \rho(S^* + \frac{q}{M} C^*)$$

Beweis. Wir zeigen, dass ein Baum T' existiert, der F aufspannt und Kosten von $E[c(T')] \leq S^* + \frac{q}{M} C^*$ hat. Durch ρ-Approximation folgt dann die Behauptung. Seien E_d alle Kanten auf einem der kürzesten Wege[1] von $d \in D$ zu F^*. Wählen wir

[1] Im Falle der Nichteindeutigkeit wählen wir einen beliebigen kürzesten Weg.

$T' := T^* \cup \bigcup_{d \in F} E_d$. Dann spannt T' die Knoten F auf und hat erwartete Kosten von

$$E[c(T')] \leq c(T^*) + \sum_{d \in D} \frac{q}{M} c(E_d) = S^* + \frac{q}{M} \sum_{d \in D} \ell(d, F^*) = S^* + \frac{q}{M} C^*$$

□

Fahren wir fort mit der Analyse der Verbindungskosten aus [8].

Satz 2.2. *[8] Die Verbindungskosten lassen sich abschätzen durch*

$$E[\sum_{d \in D} \ell(d, F)] \leq \frac{2}{q} M S^* + 2C^*$$

Beweis. Der einfachste Approximationsalgorithmus für das Steinerbaum-Problem besteht darin, im metrischen Abschluss des Graphen einen minimalen Spannbaum auf den Terminalen zu berechnen. Die Güte eines solchen Baumes beträgt höchstens 2 [18]. Die Verbindungskosten hängen nur von F ab und nicht von T, also können wir annehmen, dass T durch die MST-Heuristik entstanden ist. Wir werden nun die erwarteten Kosten, um T zu konstruieren, mit den erwarteten Verbindungskosten vergleichen. Betrachten wir einen einzelnen Schritt t der MST-Heuristik. Sei A_t die Menge der Demands, die der Algorithmus bis zum Zeitpunkt t betrachtet hat und seien $B_t \subseteq A_t$ die Terminale, die markiert wurden. Wir beginnen mit $A_1 = B_1 = \{r^*\}$. In Schritt t betrachten wir denjenigen Demand $j_{t+1} \in D \backslash A_t$, der B_t am nächsten liegt. Nun werfen wir eine Münze und mit Wahrscheinlichkeit q/M markieren wir j_{t+1} und setzen $B_{t+1} = B_t \cup \{j_{t+1}\}$ und $A_{t+1} = A_t \cup \{j_{t+1}\}$. Diese Entscheidung verursacht *Steinerbaumkosten* von $\ell(j_{t+1}, B_t)$, denn die MST-Heuristik würde nun j_{t+1} mit B_t verbinden. Da dieses Ereignis mit Wahrscheinlichkeit q/M eintritt, erhöht dies die *erwarteten* Steinerbaumkosten um $\frac{q}{M} \ell(j_{t+1}, B_t)$.

Im anderen Fall, markieren wir j_{t+1} nicht. Dann setzen wir $B_{t+1} = B_t$ und $A_{t+1} = A_t \cup \{j_{t+1}\}$. Diese Entscheidung verursacht *Verbindungskosten* von $\ell(j_{t+1}, B_t)$. Da dieses Ereignis mit Wahrscheinlichkeit $1 - \frac{q}{M}$ eintritt, erhöhen sich in diesem Schritt die *erwarteten* Verbindungskosten um höchstens $(1 - \frac{q}{M}) \cdot \ell(j_{t+1}, B_t) \leq \ell(j_{t+1}, B_t)$. Diese Situation ist in Abbildung 2.1 dargestellt.

Die Linearität des Erwartungswerts erlaubt es uns, die erwarteten Kosten aufzusummieren. Wir erhalten

$$\frac{\text{erwartete Verbindungskosten}}{\text{erwartete MST-Steinerbaumkosten}} \leq \frac{\sum_t \ell(j_{t+1}, B_t)}{\sum_t \frac{q}{M} \cdot \ell(j_{t+1}, B_t)} = \frac{M}{q}$$

Wir können die erwarteten Verbindungskosten also nach oben beschränken durch

$$\frac{M}{q} \cdot \text{erwartete MST-Steinerbaumkosten} \leq \frac{M}{q}(2S^* + 2\frac{q}{M}C^*) = \frac{2}{q} M S^* + 2C^*$$

□

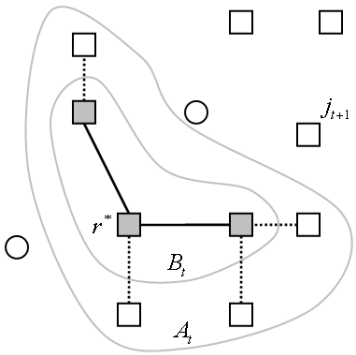


Abbildung 2.1: Schematische Darstellung der Analyse der Verbindungskosten in [8]. Rechtecke sind Demands, markierte Demands werden grau dargestellt. Durchgezogene Linien gehören zum MST-Steinerbaum, gepunktete stellen Verbindungen von Demands zu F dar.

Summieren wir nun Steinerbaum- und Verbindungskosten, so erhalten wir als Gesamtkosten unserer Lösung (F, T)

$$M \underbrace{\rho(S^* + \frac{q}{M}C^*)}_{\text{Steinerbaumkosten}} + \underbrace{\frac{2}{q}MS^* + 2C^*}_{\text{Verbindungskosten}} = \underbrace{(\rho + \frac{2}{q})}_{\searrow \text{mit } q} MS^* + \underbrace{(2 + q\rho)}_{\nearrow \text{mit } q} C^*$$

Da der erste Koeffizient $\rho + \frac{2}{q}$ mit q sinkt und der zweite Koeffizient $2 + q\rho$ mit q steigt, bestimmen wir eine Lösung der Gleichung $\rho + \frac{2}{q} = 2 + q\rho$, welche $q = 1$ lautet. Wir erhalten für diesen Fall Kosten von

$$(\rho + \frac{2}{1})MS^* + (2 + 1\rho)C^* = (2 + \rho)OPT$$

was zu der in [8] präsentierten Approximationsgüte von $2 + \rho \leq 3.55$ führt. Allein eine andere Wahl von q hätte also für diese Analyse keine Verbesserung gebracht. Es zeigt sich aber, dass die Verbindungskosten schärfer abgeschätzt werden können, was zu einer Senkung der Güte führt.

Unser Beitrag

Der Beitrag dieser Arbeit liegt nun in einer verbesserten Abschätzung der Verbindungskosten. Zentraler Punkt dazu ist der folgende

Satz 2.3. *Die Verbindungskosten lassen sich abschätzen durch*

$$E[\sum_{d \in D} \ell(d, F)] \leq \frac{1}{q}MS^* + 2C^*$$

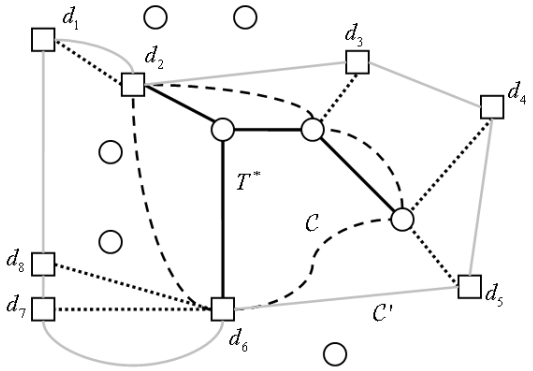


Abbildung 2.2: Beispielgraph mit vollständiger Kantenmenge. Demands werden durch Rechtecke dargestellt, gewöhnliche Knoten durch Kreise. Durchgehende Kanten gehören zu T^*, gepunktete Linien stellen Verbindungspfade dar. Der Kreis $\mathcal{C}$ ist gestrichelt eingezeichnet, Kanten von $\mathcal{C}'$ sind grau.

Beweis. Wir geben ein konkretes Verbindungsschema an, welches besagt, mit welcher Facility wir jeden Demand verbinden wollen. Da in der Zielfunktion nur kürzeste Verbindungen bezahlt werden, ergibt dies eine zulässige obere Schranke. Wir bezeichnen im weiteren Verlauf die Facility $f \in F^*$, mit der ein Demand $d \in D$ in der optimalen Lösung verbunden ist, als den *Verbindungsknoten* von d. Den Pfad von d nach f nennen wir *Verbindungspfad*.

Nun existiert ein Kreis $\mathcal{C}$ im metrischen Abschluss von G, der alle Verbindungsknoten der Demands besucht und Kosten $c(\mathcal{C}) \leq 2 \cdot c(T^*)$ aufweist. Diesen Kreis kann man erhalten, indem man alle Kanten aus T^* verdoppelt und Nicht-Verbindungsknoten übergeht (siehe Abbildung 2.2). Nun nummerieren wir die Demands $D = \{d_1, ..., d_{|D|}\}$ in derselben Reihenfolge, in der ihre Verbindungsknoten auf $\mathcal{C}$ liegen. Das bedeutet, dass wenn wir alle Kanten von $\mathcal{C}$ bei d_1 startend im Uhrzeigersinn besuchen, für $i < j$ der Verbindungsknoten von d_i vor dem Verbindungsknoten von d_j liegen muss (siehe Abbildung 2.2). Für jeden Demand müssen wir die Kosten der Verbindung zur nächsten Facility in F bezahlen. Wir tun dies, indem wir die Kosten auf die zu überquerenden Verbindungspfade und Kanten aus $\mathcal{C}$ umlegen. Unser Verbindungsschema sieht nun wie folgt aus.

> Verbinde Demand $d_i \in D$ mit dem ersten Demand der folgenden Liste[2], welcher in F ist
>
> $$d_i, d_{i+1}, d_{i-1}, d_{i+2}, d_{i-2}, d_{i+3}, ...$$

Dies bedeutet, wir verbinden d_i zu seinem nächsten markierten Nachbarn im Kreis, wobei die Kosten der Kreiskanten ignoriert werden. Die Symmetrie dieses Schemas garantiert

[2]Um die Notation einfach zu halten, nehmen wir an, dass $d_1, ..., d_{|D|}$ periodisch fortgesetzt wird, also $d_{|D|+i} = d_i$ ist.

uns, dass die erwartete Belastung der Verbindungspfade identisch ist und dass auch die erwartete Belastung der Kreiskanten identisch ist.

Schätzen wir zunächst die Belastung der Verbindungspfade ab. Für die Verbindung eines Demands $d \in D$ benötigen wir höchstens 2 Verbindungspfade, nämlich den von d selbst und den der Facility, mit der wir d verbinden. Da die Anzahl der Demands mit der Anzahl der Verbindungspfade übereinstimmt, folgt aus der Symmetrie, dass jeder Verbindungspfad im Erwartungswert höchstens zweimal belastet wird.

Nun wollen wir die Belastung auf den Kanten von $\mathcal{C}$ abschätzen. Um die Situation zu vereinheitlichen, definieren wir den Kreis $\mathcal{C}'$ als Sequenz $d_1, d_2, ..., d_{|D|}, d_1$. Legen wir nun die Kosten nicht mehr auf Verbindungspfade und Kanten von $\mathcal{C}$ um, sondern nur noch auf Kanten von $\mathcal{C}'$, dann gibt es zu jeder Kante in $\mathcal{C}$ eine Kante in $\mathcal{C}'$, die genau dieselbe Belastung erfährt (siehe Abbildung 2.2). Daher impliziert eine obere Schranke an die Belastung auf $\mathcal{C}'$ auch eine obere Schranke an die Belastung der Kanten von $\mathcal{C}$.

Stellen wir uns die Frage, wieviele Kanten von $\mathcal{C}'$ wir benötigen, um einen Demand $d_i \in D$ zu verbinden. Es sei X_i diejenige Zufallsvariable, die uns diese Zahl liefert. Wir wissen, dass $X_i > k$, wenn keines der $2k+1$ Demands $d_{i-k}, ..., d_{i+k}$ in F ist. Da wir jeden Demand unabhängig wählen, folgt

$$\Pr(X_i > k) = \Pr(d_{i-k} \notin F, ..., d_{i+k} \notin F) = \prod_{j=i-k}^{i+k} \Pr(d_j \notin F) \leq (1 - \frac{q}{M})^{2k+1}$$

Dabei ist die letzte Ungleichung strikt, wenn r^* unter den Demands $d_{i-k}, ..., d_{i+k}$ ist. Nun können wir folgern

$$\begin{aligned}
E[X_i] &\overset{(1)}{=} \sum_{k \geq 1} \Pr(X \geq k) \\
&\leq \sum_{k \geq 1} (1 - \frac{q}{M})^{2k-1} \\
&= (1 - \frac{q}{M})^{-1} \sum_{k \geq 1} ((1 - \frac{q}{M})^2)^k \\
&\overset{(2)}{=} (1 - \frac{q}{M})^{-1} \frac{(1 - \frac{q}{M})^2}{1 - (1 - \frac{q}{M})^2} \\
&= \frac{1 - \frac{q}{M}}{(1 - (1 - \frac{q}{M})) \cdot (1 + (1 - \frac{q}{M}))} \\
&= \frac{1 - \frac{q}{M}}{\frac{q}{M} \cdot (2 - \frac{q}{M}))} \\
&\leq \frac{M}{q} \cdot \frac{1 - \frac{q}{2M}}{2(1 - \frac{q}{2M}))} \\
&= \frac{M}{2q}
\end{aligned}$$

wobei (1) gilt, da $X_i \in \mathbb{N}_0$ (siehe [14], Seite 31) und (2) aus der Formel für die geometrische Reihe folgt. Wir müssen also im Erwartungswert höchstens $\frac{M}{2q}$ viele Kanten von $\mathcal{C}'$ benutzen, um einen Demand zu verbinden. Aus Symmetrie und wegen $|\mathcal{C}'| = |D|$ folgt, dass die erwartete Belastung auf $\mathcal{C}'$ und damit auch auf $\mathcal{C}$ höchstens $\frac{M}{2q}$ beträgt. Summieren wir die Belastung auf den Verbindungspfaden und auf $\mathcal{C}$, so erhalten wir wegen $c(\mathcal{C}) \leq 2 \cdot S^*$ eine obere Schranke an die Verbindungskosten von

$$\sum_{d \in D} 2\ell(d, F^*) + \frac{M}{2q} c(\mathcal{C}) \leq 2C^* + \frac{M}{q} S^*$$

□

Addieren wir nun Steinerbaumkosten und Verbindungskosten aus Lemma 2.1 und Satz 2.3, so erhalten wir den folgenden

Satz 2.4. *Für $q = 0.67$ ist Algorithmus 1 ein randomisierter 3.05-Approximationsalgorithmus.*

Beweis. Lemma 2.1 und Satz 2.3 liefern uns, dass wir die Kosten der von Algorithmus 1 berechneten Lösung beschränken können durch

$$M \underbrace{\rho(S^* + \frac{q}{M} C^*)}_{\text{Steinerbaumkosten}} + \underbrace{2C^* + \frac{M}{q} S^*}_{\text{Verbindungskosten}} = \underbrace{(\rho + \frac{1}{2q})}_{\searrow \text{ mit } q} MS^* + \underbrace{(2 + q\rho)}_{\nearrow \text{ mit } q} C^* \tag{2.1}$$

Für $q = 0.67$ ergibt Gleichung 2.1 den Wert

$$\leq 3.05 \cdot MS^* + 3.05 \cdot C^* = 3.05 \cdot OPT$$

und die Behauptung folgt. □

2.2 Worst-Case Instanzen für den Sampling-Algorithmus

Wie immer bei oberen Schranken an Approximationsgüten, stellt man sich auch hier die Frage, ob diese Schranke scharf ist. Um dieser Frage nachzugehen, geben wir nun zwei Beispielinstanzen an, auf denen sich der Sampling-Algorithmus nicht gut schlägt. Um die Analyse für eine allgemeine Sampling-Wahrscheinlichkeit q/M durchzuführen, geben wir eine Instanz an, auf der der Algorithmus versagt, wenn q/M groß ist und eine bei der kleine Sampling-Wahrscheinlichkeiten zu teuren Lösungen führen. Dabei betrachten wir in beiden Fällen den Grenzwert der Güten, die wir erhalten, indem wir n als Anzahl der Demands gegen ∞ laufen lassen und $M := n^{1/3}$ definieren. Diese Wahl hat zur Folge, dass die erwartete Anzahl von Facilities, die der Sampling-Algorithmus wählt, ebenfalls unbeschränkt ist.

Betrachten wir den in Abbildung 2.3 dargestellten Graphen G_1, der aus einem Kammgraphen und einem Stern mit Zentrum d_n besteht. Kanten (d_i, v_i) haben Kosten 1, die übrigen Kanten sind umsonst. Wir überlegen uns im folgenden Lemma, warum der Sampling-Algorithmus Probleme bei dieser Instanz hat.

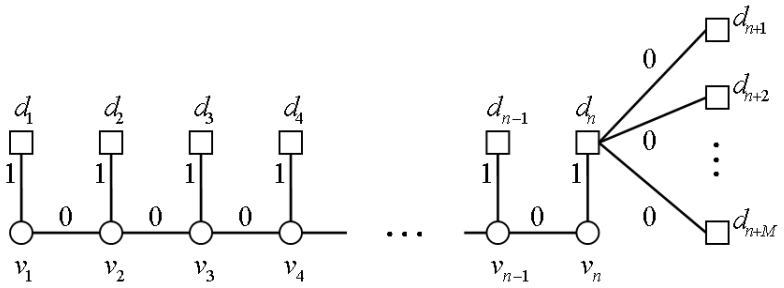


Abbildung 2.3: CFL-Instanz G_1, auf der der Sampling-Algorithmus für großes q/M eine teure Lösung liefert. Kanten sind mit ihren Kosten beschriftet.

Lemma 2.5. *Die erwartete Approximationsgüte des Sampling-Algorithmus auf Graph G_1 beträgt mindestens $(q+2) \cdot (1 - o(1))$ für $n \to \infty$, wenn $M := n^{1/3}$ gesetzt wird.*

Beweis. Eine optimale CFL-Lösung sieht so aus, dass Facilities $F^* = \{v_1, ..., v_n, d_n\}$ und der Steinerbaum $T^* = \{(v_i, v_{i+1}) | i \in \{1, ..., n-1\}\} \cup \{(d_n, v_n)\}$ mit Kosten $c(T^*) = 1$ gewählt werden. Verbindungskosten fallen in Höhe von $n - 1$ an, also ist $OPT_1 \leq n + M \cdot 1$, wobei OPT_1 den Wert einer optimalen CFL-Lösung auf G_1 bezeichnet. Die Kosten der optimalen Lösung verteilen sich wegen $M/n \to 0$ im Wesentlichen auf die Verbindungskosten.

Betrachten wir nun die Lösung (F, T), die der Sampling-Algorithmus berechnet. Der Sampling-Algorithmus geht davon aus, eine Facility aus der optimalen Lösung zu kennen, sei dies d_n. Ohne die Existenz der Demands $d_{n+1}, ..., d_{n+M}$ hätten wir diese Wahl nicht treffen können. Diese Demands haben somit ihren Zweck erfüllt und wir konzentrieren uns nun auf die Kosten, die die übrigen Demands $D' = \{d_1, ..., d_n\}$ verursachen.

Wegen $n/M \to \infty$ geht die Wahrscheinlichkeit, dass mindestens 2 Demands aus D' als Facilities gewählt werden, gegen 1. Wir können daher o.B.d.A. $|F \cap D'| \geq 2$ annehmen. Ein Steinerbaum enthält dann mindestens $|F \cap D'|$ viele Kanten mit Kosten 1. Für einen nichtgewählten Demand $d \in D' \backslash F$ müssen Verbindungskosten von 2 bezahlt werden. Im Erwartungswert kostet eine solche Lösung also mindestens

$$E[|F \cap D'|] \cdot M + E[|D' \backslash F|] \cdot 2 \geq \frac{q}{M} n \cdot M + (1 - \frac{q}{M}) n \cdot 2 = qn + (1 - \frac{q}{M}) n \cdot 2$$

Die erwartete Güte der Lösung (F, T) beträgt also mindestens

$$\frac{qn + (1 - \frac{q}{M}) n \cdot 2}{OPT_1} \geq \frac{qn + (1 - \frac{q}{M}) n \cdot 2}{n + M} = \frac{q + 2 \cdot (1 - \frac{q}{n^{1/3}})}{1 + n^{-2/3}} = (q+2) \cdot (1 - o(1))$$

wobei die letzte Gleichung für $n \to \infty$ folgt. □

Diese Güte wird also immer schlechter, je größer q wird. Betrachten wir nun die zweite Instanz G_2, die als metrischer Abschluss des in Abbildung 2.4 dargestellten Pfadgraphen definiert ist. G_2 ist also der vollständige Graph auf $d_1, ..., d_n$ mit Kantenkosten $c(d_i, d_j) = |i - j|$.

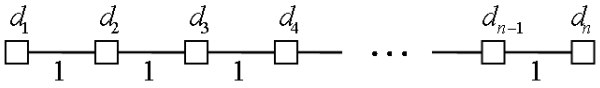


Abbildung 2.4: CFL-Instanz G_2, auf der der Sampling-Algorithmus für kleines q/M eine teure Lösung liefert, ist der metrische Abschluss des hier dargestellten Graphen. Kanten sind mit ihren Kosten beschriftet.

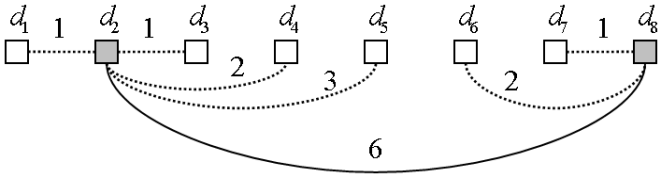


Abbildung 2.5: Verhalten des Sampling-Algorithmus auf Instanz G_2 für $n = 8$ unter der Annahme, dass $F = \{d_2, d_8\}$ gewählt wird. Verbindungen von Demands zu F sind gepunktet eingezeichnet, durchgezogene Kanten gehören zu T. Kanten sind mit Kantenkosten beschriftet.

Lemma 2.6. *Die erwartete Approximationsgüte des Sampling-Algorithmus auf Graph G_2 beträgt mindestens $(1 + \frac{1}{2q}) \cdot (1 - o(1))$ für $n \to \infty$ mit $M := n^{1/3}$.*

Beweis. Eine günstige (aber nichtoptimale) Lösung sieht so aus, dass wir alle Demands $\{d_1, ..., d_n\}$ als Facilities und $T^* = \{(d_i, d_{i+1}) | i = 1, ..., n-1\}$ wählen. Die Kosten einer solchen Lösung betragen $M(n-1)$, da keine Verbindungskosten anfallen. Sei OPT_2 der Wert einer optimalen CFL-Lösung für G_2, dann folgt $OPT_2 \leq M(n-1) \leq Mn$.

Schätzen wir nun die erwarteten Kosten einer Lösung ab, die der Sampling-Algorithmus liefert. Um uns das Verhalten des Algorithmus vor Augen zu führen, betrachten wir Abbildung 2.5, in der der Fall $n = 8$ dargestellt ist. Dabei wird angenommen, dass $F = \{d_2, d_8\}$ gewählt wird. Berechnen wir zunächst die erwarteten Kosten des Steinerbaumes T. Es gilt $c(T) \geq \max\{|i-j| \, | d_i, d_j \in F\}$. Die Wahrscheinlichkeit, dass von den ersten k Demands $d_1, ..., d_k$ oder von den letzten k Demands $d_{n-k+1}, ..., d_n$ kein Demand als Facility gewählt wird, ist exponentiell klein in k. Daher gilt $E[\max\{|i-j| \, | d_i, d_j \in F\}] = n \cdot (1 - o(1))$ und die Steinerbaumkosten betragen mindestens $n \cdot (1 - o(1))$. Bezahlt werden muss also im Wesentlichen ein Steinerbaum über die gesamte Länge des Pfadgraphen.

Betrachten wir nun die Verbindungskosten eines beliebigen Demands $d_i \in D$. Wir werden zeigen, dass für den überwiegenden Teil der Demands erwartete Verbindungskosten von etwa $\frac{M}{2q}$ anfallen. Definieren wir dazu die Zufallsvariable

$$X_i := \min\{|i-j| \, | d_j \in F\} = \text{Kosten um } d_i \text{ zu verbinden}$$

Sei $d_{i^*} = r^* \in F^*$ der Demand, den der Sampling-Algorithmus laut Voraussetzung aus

der Menge der optimalen Facilities kennt. Wir definieren nun die folgende Teilmenge der Demands

$$D' := \left\{ d_i \in D | \, |i - i^*| > n^{2/3} \right\},$$

dann gibt D' eine Menge von Demands an, deren Abschätzung der Verbindungskosten uns keine Probleme machen wird. Da wir einen Grenzwertprozess für $n \to \infty$ betrachten, genügt es die Aussage für diejenigen n zu zeigen, für die $n^{2/3}$ ganzzahlig ist. Es sei dies nun vorausgesetzt. Wir wissen, dass $X_i > k$ ist, wenn $\{d_{i-k}, ..., d_{i+k}\} \cap F = \emptyset$. Für $d_i \in D'$ und $k \leq n^{2/3}$) gilt $\Pr(X_i \geq k) \geq (1 - \frac{q}{M})^{2k-1}$. Ähnlich zur Analyse der Verbindungskosten in Satz 2.3 können wir unter der Voraussetzung $d_i \in D'$ folgern, dass

$$\begin{aligned}
E[X_i] &= \sum_{k=1}^{\infty} \Pr(X_i \geq k) \\
&\geq \sum_{k=1}^{n^{2/3}} \Pr(X_i \geq k) \\
&\geq \sum_{k=1}^{n^{2/3}} (1 - \frac{q}{M})^{2k-1} \\
&\geq \sum_{k=1}^{n^{2/3}} ((1 - \frac{q}{M})^2)^k \\
&\overset{(1)}{\geq} \frac{\left(1 - \frac{q}{n^{1/3}}\right)^2 - \left(\left(1 - \frac{q}{n^{1/3}}\right)^2\right)^{n^{2/3}+1}}{1 - (1 - \frac{q}{M})^2} \\
&\overset{(2)}{=} \frac{(1 - o(1)) - o(1)}{(1 - (1 - \frac{q}{M})) \cdot (1 + (1 - \frac{q}{M}))} \\
&= \frac{1 - o(1)}{\frac{q}{M} \cdot \left(2 - \frac{q}{n^{1/3}}\right)} \\
&= \frac{M}{2q}(1 - o(1))
\end{aligned}$$

Dabei wird in (1) die geometrische Reihe mit der Formel

$$\sum_{i=1}^{m} p^i = \frac{p - p^{m+1}}{1 - p}$$

für $-1 < p < 1$ aufgelöst und $M = n^{1/3}$ eingesetzt. (2) folgt aus $(1 - \frac{q}{n^{1/3}})^2 = 1 - o(1)$ und

$$\left(\left(1 - \frac{q}{n^{1/3}}\right)^2\right)^{n^{2/3}+1} \leq \left(\left(1 - \frac{q}{n^{1/3}}\right)^{n^{1/3}}\right)^{2n^{1/3}} \leq e^{-q \cdot 2n^{1/3}} \to 0$$

Wegen $|D'| = n \cdot (1 - o(1))$ erhalten wir insgesamt erwartete Verbindungskosten von mindestens

$$n(1 - o(1)) \cdot \frac{M}{2q}(1 - o(1)) = n\frac{M}{2q}(1 - o(1))$$

für alle Demands. Teilen wir die Kosten der Lösung (F, T) durch die obere Schranke an OPT_2, so erhalten wir eine untere Schranke an die Güte von

$$\frac{Mn(1 - o(1)) + n\frac{M}{2q}(1 - o(1))}{Mn} = (1 + \frac{1}{2q})(1 - o(1))$$

□

Korollar 2.7. *Der Sampling-Algorithmus mit Sampling-Wahrscheinlichkeit q/M hat asymptotisch für $n \to \infty$ mit $M = n^{1/3}$ eine Güte von mindestens*

$$\max\left\{q + 2, 1 + \frac{1}{2q}\right\}$$

Beweis. Füge Lemma 2.5 und Lemma 2.6 zusammen. □

Bemerkung. In Kapitel 2.1 haben wir eine *obere* Schranke von $\max\{\rho q + 2, \rho + \frac{1}{q}\}$ an die Approximationsgüte nachgewiesen. Ein Grund für die Diskrepanz zwischen unterer und oberer Schranke besteht darin, dass wir einen Steinerbaum-Algorithmus als Black-Box verwenden. Ein anderer Grund liegt in unserer Analyse der Verbindungskosten, in welcher wir jede Kante von T^* mit $\frac{M}{q}$ belasten, wobei wir kein Beispiel finden, indem jede Kante tatsächlich mit mehr als $\frac{M}{2q}$ belastet werden müsste. Es ist daher eine Vermutung des Autors, dass diese obere Schranke nicht scharf ist und sich die Verbindungskosten sogar durch

$$E[\sum_{d \in D} \ell(d, F)] \stackrel{?}{\leq} \frac{1}{2q} M S^* + 2C^*$$

abschätzen lassen.

Korollar 2.8. *Der Algorithmus in [8] hat für $q = 1$ eine Güte von mindestens 3.*

Korollar 2.9. *Der Sampling-Algorithmus mit Sampling-Wahrscheinlichkeit q/M hat eine Güte von mindestens 2.36 für jedes feste q.*

Beweis. Die Güte ist asymptotisch mindestens $\max\{q + 2, 1 + \frac{1}{2q}\}$. Beide Koeffizienten sind gleich bei $q = (\sqrt{3} - 1)/2 \geq 0.36$. Die Güte ist also mindestens $0.36 + 2 = 2.36$. □

3 Das Maybecast-Problem

Historie und neue Resultate

Die Problemstellung des Maybecast-Problems verlangt, in einem gewichteten Graphen $G = (V, E)$ von jedem Terminal einen Pfad zur Wurzel r zu berechnen, so dass die Kantenmenge, die durch die Pfade der aktiven Terminale entsteht, im Erwartungswert möglichst kostengünstig ist. Jedes Terminal $i \in R$ wird dabei mit Wahrscheinlichkeit p_i aktiv.

Natürlich ist dieses Problem $\mathcal{NP}$-schwierig, denn für $p_i = 1$ entspricht es genau dem Steinerbaum-Problem. Das Maybecast-Problem wurde zuerst in [12] untersucht, mit dem Ergebnis eines Approximationsalgorithmus mit Güte 40.7. Wir werden dieses Ergebnis hier deutlich verbessern, indem wir einen Approximationsalgorithmus mit Güte 4.83 liefern.

3.1 Eine neue Kostenfunktion

Betrachten wir eine beliebige Kante $e \in E$ mit der Menge U_e der Terminale, deren Pfad zu r über e läuft. Die Wahrscheinlichkeit k_e, dass die Kante e aktiv ist, beträgt dann

$$k_e = 1 - \prod_{i \in U_e} (1 - p_i)$$

Das bedeutet, wir müssen im Erwartungswert $k_e c(e)$ an Kosten für Kante e bezahlen. Wir überlegen uns, dass die erwarteten Kosten einer Kante bei Anheben der p_i's zwar absolut gesehen ansteigen, aber bezogen auf die Summe der p_i's sinken. Die Kostenfunktion k_e ist also konkav. Für konkave Kostenfunktionen konnte in [12] gezeigt werden, dass es immer eine optimale Lösung gibt, in der die Pfade einen Baum formen. Daher können wir uns auf Baumlösungen beschränken. Ein solcher Baum als Lösung induziert die gesuchten Pfade wieder eindeutig.

Eine wichtige Vereinfachung besteht nun darin, k_e als einen wesentlich einfacheren Ausdruck $\hat{k}_e$ darzustellen, der stückweise linear in den Wahrscheinlichkeiten p_i ist und von k_e höchstens um einen konstanten Faktor abweicht. Dies garantiert uns das folgende von Karger und Minkoff bewiesene Lemma.

Lemma 3.1. *[12] Sei* $\hat{k}_e = \min\left\{\sum_{i \in U_e} p_i, 1\right\}$. *Dann gilt*

$$k_e \leq \hat{k}_e \leq \frac{1}{1 - 1/e} k_e \tag{3.1}$$

für alle Wahrscheinlichkeiten p_i *und alle Mengen* U_e.

Beweis. Zeigen wir zunächst die erste Ungleichung. Die Union Bound liefert uns

$$k_e = \Pr(\bigcup_{i \in U_e} i \text{ ist aktiv}) \leq \sum_{i \in U_e} \Pr(i \text{ ist aktiv}) = \sum_{i \in U_e} p_i$$

Da Wahrscheinlichkeiten höchstens 1 sind, folgt $k_e \leq \min\{\sum_{i \in U_e} p_i, 1\}$. Zeigen wir nun die Ungleichung

$$\min\left\{\sum_{i \in U_e} p_i, 1\right\} \leq \frac{1}{1-1/e}\left(1 - \prod_{i \in U_e}(1-p_i)\right)$$

Halten wir $s := \sum_{i \in U_e} p_i$ fest. Wegen $1 - x \leq e^{-x}$ gilt

$$\prod_{i \in U_e}(1-p_i) \leq \prod_{i \in U_e} e^{-p_i} = e^{-s}$$

Wir müssen also

$$\min\{s, 1\} \leq \frac{1}{1-1/e}(1-e^{-s})$$

zeigen. Für $s \geq 1$ ist $\frac{1}{1-1/e}(1-e^{-s}) \geq \frac{1}{1-1/e}(1-e^{-1}) = 1$. Für $0 \leq s < 1$ ist

$$s \leq \frac{1}{1-1/e}(1-e^{-s}) \Leftrightarrow \frac{s}{1-e^{-s}} \geq \frac{1}{1-1/e}$$

zu zeigen. Da $\frac{s}{1-e^{-s}}$ in $(0,1]$ jedoch eine mit s steigende Funktion ist, nimmt diese für $s = 1$ ein Maximum an und die Behauptung folgt. □

Dieses Lemma zeigt, dass wir als Kostenfunktion auf den Kanten $\hat{k}_e$ annehmen können und dadurch eine gefundenen Lösung höchstens um den Faktor $\frac{1}{1-1/e} \leq 1.59$ teurer wird.

3.2 Reduktion auf Connected-Facility-Location

In [12] wird nach Einführung der vereinfachten Kostenfunktion $\hat{k}_e$ das Problem auf das sogenannte *Gathering-Problem* reduziert. Dieses hat als Ziel eine Auswahl von Knoten, welche als Hubs bezeichnet werden, so dass die Verbindungskosten der Terminale zu den Hubs minimiert werden, wobei die Summe der Gewichte p_i der Terminale, die einem Hub zugeordnet werden, mindestens $1/2$ betragen muss. Dieses Gathering-Problem wiederum wird auf das *Facility-Location-Problem* reduziert, bei welchem wie beim CFL-Problem Facilities gesucht sind, welche aber nicht verbunden werden müssen, sondern für die bei Öffnung fixe Kosten bezahlt werden müssen. Durch diese Vielzahl von Reduktionen kumuliert sich schließlich die Approximationsgüte auf 40.7. Wir werden nach der Einführung der Kostenfunktion $\hat{k}_e$ einen anderen Weg einschlagen und mit einer Reduktion auskommen, wodurch sich die Güte unseres Algorithmus für das Maybecast-Problems durch 4.83 abschätzen lässt.

Unser Ziel wird es nun sein, zu zeigen, dass das Maybecast-Problem mit modifizierter Kostenfunktion dem Rent-or-Buy-Problem entspricht. Erinnern wir uns zunächst, dass

wir $p_i \in \mathbb{Q}$ angenommen haben. Sei $M \in \mathbb{N}$ der Hauptnenner aller p_i, dann können wir $p_i = n_i/M$ schreiben für $n_i \in \{1, ..., M\}$. Ersetzen wir der Einfachheit jedes Terminal $i \in R$ durch n_i viele Terminale, dies ändert den Wert der Koeffizienten $\hat{k}_e$ nicht[1]. Selbst im Fall, dass $\sum_{i \in R} n_i$ superpolynomiell ist, kann dies effizient implementiert werden. Nun wird jedes Terminal mit Wahrscheinlichkeit $1/M$ aktiv. Gesucht ist nun ein Baum $T \subseteq E$, so dass

$$\sum_{e \in T} c(e) \cdot \min \left\{ \frac{1}{M} \cdot \text{Anzahl Terminale im Teilbaum unter } e, 1 \right\} \tag{3.2}$$

minimiert wird. Skalieren wir die Zielfunktion mit M, so verändert sich das Optimum nicht und die Zielfunktion lautet nun

$$\sum_{e \in T} c(e) \cdot \min\{\text{Anzahl Terminale im Teilbaum unter } e, M\}$$

Nun stimmt die Zielfunktion genau mit der des Rent-or-Buy-Problems mit Parameter M überein. Wir müssen noch zeigen, dass wir unseren Approximationsalgorithmus für das CFL-Problem auch in leicht modifizierter Form für das Rent-or-Buy-Problem verwenden können.

Lemma 3.2. *Das Rent-or-Buy-Problem ist* 3.05*-approximierbar.*

Beweis. Wenden wir Algorithmus 1 auf das Rent-or-Buy-Problem an, mit der einzigen Modifikation, dass wir nun die Wurzel r als dasjenige Element verwenden können, von dem bekannt ist, dass es in der optimalen Lösung eine Facility ist. Die Analyse des Algorithmus kann unverändert übernommen werden. Wir müssen also nur noch zeigen, dass das Rent-or-Buy-Problem identisch mit dem CFL-Problem ist, sofern $r \in F^*$ vorausgesetzt wird. Sei (F^*, T^*) eine optimale CFL-Lösung. Sei T' die Vereinigung der Kanten auf T^* und der Verbindungspfade. Wir können dabei die Verbindungspfade stets so wählen, dass T' einen Baum formt. Betrachten wir eine beliebige Kante $e \in T'$. Falls wir e aus T' entfernen, zerfällt T' in zwei Zusammenhangskomponenten. Sei k die Anzahl der Terminale in der Zusammenhangskomponente, die r *nicht* enthält. Dann verursacht diese Kante im Rent-or-Buy-Problem Kosten von $c(e) \cdot \min\{k, M\}$. Betrachten wir nun die anfallenden Kosten im CFL-Problem. Für $k < M$, ist es optimal, wenn $e \notin T^*$, dann gehen k viele Verbindungen über e, welche Kosten von $c(e) \cdot k$ erzeugen. Falls $k \geq M$, so werden die Kosten minimiert, wenn der Steinerbaum T^* auch e umfasst, dann müssen wir $c(e) \cdot M$ für e bezahlen. Dieses Argument funktioniert, da die Kanten, die eine Belastung von mindestens M aufweisen, zusammenhängend sein müssen. Wir sehen, dass die Kosten in beiden Problemen übereinstimmen und die Behauptung folgt. □

Wir können also das Maybecast-Problem auf das Rent-or-Buy-Problem reduzieren und dieses dann mit einem Approximationsalgorithmus mit Güte 3.05 lösen. Fassen wir dies explizit in einem Algorithmus zusammen. Dazu müssen wir uns noch überlegen,

[1] Die Werte k_e würden durch diese Ersetzung sehr wohl verändert werden.

Algorithmus 2 Maybecast-Algorithmus

Eingabe: Graph $G = (V, E)$; Terminale $R \subseteq V$ mit Wahrscheinlichkeiten p_i für alle $i \in R$; Wurzel $r \in V$

Ausgabe: Baum

1. Markiere jedes Terminal $i \in R$ mit Wahrscheinlichkeit $1 - e^{-0.67p_i}$
2. Setze $R' := \{\text{markierte Terminale}\} \cup \{r\}$
3. Berechne einen ρ-approximativen Steinerbaum T auf R'
4. Verbinde alle Terminale auf kürzestem Weg zu R'
5. Gebe Kanten in T und auf kürzesten Wegen aus

mit welcher Wahrscheinlichkeit ein Terminal $i \in R$, welches mit Wahrscheinlichkeit $p_i = n_i/M$ aktiv ist, nach der Reduktion vom Sampling-Algorithmus als Facility gewählt wird. Bezeichnen wir die n_i vielen Demands, in die i zerteilt wird, als $d_1, ..., d_{n_i}$. Der Sampling-Algorithmus wählt jeden der Demands $d_1, ..., d_{n_i}$ mit Wahrscheinlichkeit $0.67/M$. Der mit M gewichtete Steinerbaum enthält das Terminal i daher mit Wahrscheinlichkeit

$$\Pr\left(\bigcup_{j=1}^{n_i} d_j \text{ wird als Facility gewählt}\right) = 1 - \left(1 - \frac{0.67}{M}\right)^{n_i} = 1 - \left(\left(1 - \frac{0.67}{M}\right)^{M}\right)^{p_i}$$

Nun haben wir M als Hauptnenner der p_i's gewählt. Aber wir können M auch durch ein beliebiges Vielfaches des Hauptnenners ersetzen, die Analyse bleibt weiterhin gültig. Lassen wir M gegen ∞ laufen, so erhalten wir

$$\lim_{M\to\infty} 1 - \left(\left(1 - \frac{0.67}{M}\right)^{M}\right)^{p_i} = 1 - e^{-0.67p_i}$$

Wir können die obige Reduktion daher durch den explizit formulierten Algorithmus 2 ersetzen.

Korollar 3.3. *Algorithmus 2 liefert eine Lösung mit Güte* 4.83 *für das Maybecast-Problem.*

Beweis. Die Zielfunktion 3.2 unterscheidet sich von der originalen Zielfunktion höchstens um einen Faktor $\frac{1}{1-1/e}$. Die veränderte Zielfunktion können wir mit Güte 3.05 approximieren. Insgesamt erhalten wir also eine Güte von

$$\frac{1}{1 - 1/e} \cdot 3.05 \leq 4.83$$

□

4 Die Komplexität des VPND-Problems

Historie und neue Resultate

Für das allgemeine VPND-Problem konnte die beste verfügbare Approximationsgüte in den letzten Jahren von der ersten konstanten Güte von 5.55 [8] über 4.74 [4] schließlich auf 3.55 [5] verbessert werden. Die ersten beiden Algorithmen liefern sogar eine Baumlösung, also implizieren diese direkt einen Approximationsalgorithmus mit Güte 4.74 für die VPND-Tree-Variante.

Vor dieser Arbeit war die $\mathcal{NP}$-Schwierigkeit des VPND-Problems lediglich für die Spezialfälle bekannt, in denen nur ein [7] bzw. zwei Sender vorliegen. Die Reduktion des Steinerbaum-Problems auf VPND impliziert genau einen Sender, während die Reduktion von Max-Leaf-Spanning-Tree auf VPND eine Instanz mit genau zwei Sendern hervorbringt[1]. Die erste dieser Reduktionen werden wir in Kapitel 4.1 wiederholen. Dies wird uns die notwendige Idee liefern, um in Kapitel 4.2 auch eine Reduktion anzugeben, die zeigt, dass VPND in allen Fällen $|R| = |S| + b$ für beliebiges $b \in \mathbb{Z}\backslash\{0\}$ $\mathcal{NP}$-schwierig ist. Interessanterweise können wir in Kapitel 5.2 zeigen, dass diese Fälle für die Baumvariante des VPND-Problems effizient lösbar sind, sofern $b = O(1)$.

4.1 Die bisher bekannte Reduktion des Steinerbaum-Problems auf VPND

Betrachten wir zunächst die altbekannte Reduktion des Steinerbaum-Problems auf das VPND-Problem. Es sei also eine Steinerbaum-Instanz aus Graph $G = (V, E)$, Kostenfunktion $c : E \to \mathbb{R}^+$ und Terminalen $\{s, r_1, ..., r_k\} \subseteq V$ gegeben. Nun konstruieren wir eine VPND-Instanz, indem wir den Graphen G und die Kosten c übernehmen, als Sender $S = \{s\}$ und als Empfänger $R = \{r_1, ..., r_k\}$ wählen. In einer Lösung des VPND-Problems muss der einzige Sender mit allen Empfängern verbunden sein. Also spannt $T = \{e \in E | u(e) > 0\}$ alle Terminale auf. In einer optimalen Lösung gilt $u(e) \in \{0, 1\}$, da es nur einen Sender gibt. Desweiteren kann man Kreise aus T eliminieren − die Lösung bleibt zulässig und wird höchstens billiger. Also ist in einer optimalen Lösung der VPND-Instanz die Menge T ein kostenminimaler Baum, der alle Terminale aufspannt, und damit eine optimale Lösung des Steinerbaum-Problems (siehe Abbildung 4.1).

Diese Reduktion impliziert auch die starke $\mathcal{NP}$-Vollständigkeit und dass es kein PTAS

[1] Für die Reduktion des Max-Leaf-Spanning-Tree-Problems auf VPND ist es zusätzlich noch notwendig, Lösungen auf Bäume einzuschränken.

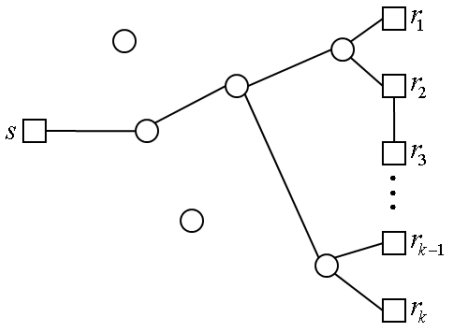


Abbildung 4.1: Graph mit eingezeichnetem optimalen Steinerbaum, der gleichzeitig optimale VPND Lösung ist, wenn auf jeder eingezeichneten Kante genau eine Kapazitätseinheit reserviert wird.

für VPND geben kann, sofern $\mathcal{NP} \neq \mathcal{P}$, da sich auch die in [1] gezeigte APX-Vollständigkeit vom Steinerbaum-Problem auf VPND überträgt.

4.2 Eine neue Reduktion

Nun wollen wir eine modifizierte Reduktion angeben, die zeigt, dass das VPND-Problem auch $\mathcal{NP}$-schwierig ist, wenn $|R| = |S| + b$ für ein beliebiges $b \geq 1$. Sei nun wie zuvor eine Steinerbaum-Instanz $\mathcal{I}_{\mathrm{St}}$ mit Graph $G = (V, E)$ und Terminalen gegeben, von denen wir willkürlich eines als r_1 und die übrigen als $s_1, ..., s_k$ bezeichnen. Wählen wir nun $m = |S| + b$ und konstruieren eine VPND-Instanz $\mathcal{I}_{\mathrm{VPND}}$ mit Sendern $S = \{s_1, ..., s_k\}$ und Empfängern $R = \{r_1, ..., r_m\}$ wie folgt

- Übernehme den Graphen G.
- Füge zusätzliche Empfänger $r_2, ..., r_m$ sowie den Dummy-Knoten d ein.
- Füge Kanten (s_i, d) mit Kosten K für alle $i = 1, ..., k$ ein. Dabei soll K genügend groß sein (z.B. $K := |V| \sum_{e \in E} c(e)$).
- Füge Kanten (d, r_i) mit Kosten 0 für alle $i = 2, ..., m$ ein.

Sei T^* optimale Lösung von $\mathcal{I}_{\mathrm{St}}$.

Satz 4.1. *Eine optimale Lösung der VPND-Instanz $\mathcal{I}_{\mathrm{VPND}}$ besteht daraus, auf jeder Kante in $T^* \cup \{(s_i, d) | i = 1, ..., k\} \cup \{(d, r_i) | i = 2, ..., m\}$ genau eine Kapazitätseinheit zu installieren. Als Pfade von den Sendern zu r_1 wählt man den eindeutigen Pfad durch T^*. Als s_i-r_j-Pfad mit $i \in \{1, ..., k\}$ und $j \in \{2, ..., m\}$ wählt man $s_i \to d \to r_j$ aus.*

Beweis. Man überlegt sich, dass die angegebene Lösung zulässig für das VPND ist, da bei derartiger Pfadaufteilung nie mehr als eine Kapazität auf einer Kante benötigt

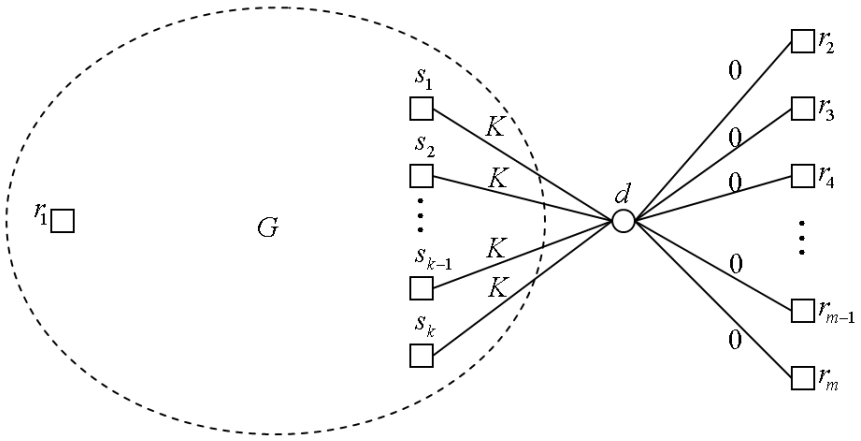


Abbildung 4.2: VPND-Instanz $\mathcal{I}_{\text{VPND}}$. Wichtig ist dabei, dass die Anzahl der Empfänger $r_2, ..., r_m$ mindestens so groß ist wie die Anzahl der Sender $s_1, ..., s_k$.

wird. Sei $OPT_{\text{St}} = c(T^*)$ der Wert der optimalen Steinerbaumlösung und OPT_{VPND} der optimale Zielfunktionswert der VPND-Instanz. Die angegebene Lösung hat Kosten von

$$OPT_{\text{St}} + k \cdot K$$

Es ist also zu zeigen, dass *jede* Lösung dieser VPND-Instanz mindestens Kosten $OPT_{\text{St}} + k \cdot K$ hat.

Betrachten wir zunächst das Verkehrsszenario, in welchem alle k Sender an die k vielen Empfänger $\{r_2, ..., r_{k+1}\}$[2] senden. Um dieses Szenario zu unterstützen sind mindestens k Kapazitätseinheiten auf den teuren Kanten (s_i, d) notwendig.

Nehmen wir an, dass die Kanten, auf denen im Teilgraph G Kapazität reserviert wird, r_1 *nicht* mit allen Sendern verbindet. Dann muss es aber Sender $s_i \neq s_j$ und einen von den Kapazitäten unterstützten Pfad $s_i \to d \to s_j \to ... \to r_1$ geben. Allerdings benötigt allein dieser Pfad schon 2 der teuren Kanten. Wählen wir ein zulässiges Verkehrsszenario, in welchem s_i an r_1 sendet und die übrigen $k-1$ Sender an $\{r_2, ..., r_k\}$ senden. Dann sind alleine für die Unterstützung dieses Verkehrsszenario schon $k+1$ viele der teuren Kanten notwendig. Wegen $(k+1)K > OPT_{\text{VPND}}$ kann diese Lösung nicht optimal sein.

Wir können schlussfolgern, dass in einer optimalen Lösung sowohl k viele der teuren Kanten enthalten sein müssen, als auch Kanten aus dem Teilgraph G, welche $\{r_1, s_1, ..., s_k\}$ aufspannen. Da T^* als optimaler Steinerbaum die günstigste Kantenmenge ist, die $\{r_1, s_1, ..., s_k\}$ aufspannt, folgt

$$OPT_{\text{VPND}} \geq c(T^*) + k \cdot K$$

Somit ist die Behauptung gezeigt, dass die angegebene Lösung eine optimale VPND-Lösung darstellt. □

[2] daher die Voraussetzung $m > k$

Die Möglichkeit einer Optimierung des VPND-Problems bei $|R| = |S| + b$ impliziert also auch eine Optimierung des $\mathcal{NP}$-schwierigen Steinerbaum-Problems. Da sich im VPND-Problem die Mengen der Sender und Empfänger austauschen lassen, erhalten wir, dass das VPND-Problem für $|R| = |S| + b$ für alle $b \in \mathbb{Z}\backslash\{0\}$ $\mathcal{NP}$-schwierig ist.

Einziger Wermutstropfen bleibt, dass obige Argumente gerade beim interessanten Fall $|S| = |R|$ in dieser Form nicht funktionieren. Man sollte sich auch vergewissern, dass eine optimale Lösung die Kreise $r_1 \to ... \to s_i \to d \to s_j \to ... \to r_1$ für alle $i, j \in \{1, ..., k\}$ enthält und somit *keinen* Baum darstellt. Diese Reduktion kann daher nicht für die Baumvariante übernommen werden.

5 Das VPND-Tree-Problem

Historie und neue Resultate

Die Baumvariante des VPND-Problems ist bisher z.B. in [7] separat untersucht worden, mit dem Ergebnis einer 9.002-Approximation. Dieses Ergebnis wurde durch Reduktion auf CFL erzielt. Diese Methode hätte zusammen mit dem bislang besten CFL-Algorithmus eine Güte von 3.55 für VPND-Tree gezeigt. Unsere verbesserte Analyse des Sampling-Algorithmus impliziert daher auch eine Güte von 3.05 für das VPND-Tree-Problem.

Dieselbe Reduktion wie in 4.1 funktioniert auch bei VPND-Tree, also ist auch dieses Problem zumindest bei $|S| = 1$ $\mathcal{NP}$-schwierig. Wir werden in diesem Kapitel zeigen, dass sich eine Unterscheidung der Baumvariante vom allgemeinen Fall durchaus lohnt, da wir für die Baumvariante Ergebnisse erzielen können, die für das allgemeine Problem nicht gelten.

Nachdem wir uns in Kapitel 5.1 die Struktur des Problems klargemacht haben, erweitern wir die in polynomieller Zeit lösbaren Fälle von $|R| = |S|$ [7] auf $|R| = |S| \pm O(1)$. Danach geben wir ein PTAS für den Fall $|R| = O(1) \cdot |S|$. In Kapitel 5.4 werden wir die Güte der durch den günstigsten Shortest-Path-Tree implizierten Lösung von $1 + |R|/|S|$ [5] auf $\frac{|R|+|S|}{2|S|}$ verbessern und zeigen, dass diese Schranke scharf ist.

5.1 Zur Struktur

Wir haben uns bereits in Kapitel 1 klar gemacht, dass ein VPND-Tree-Algorithmus als Lösung nur einen Baum $T \subseteq E$ ausgeben muss, die Pfade sind durch diesen eindeutig bestimmt. Auch die zu reservierenden Kapazitäten könnte man wie beim allgemeinen Problem mit einem Algorithmus für maximales Matching berechnen. Doch es zeigt sich, dass wir uns diese Mühe gar nicht machen müssen. Betrachten wir dazu eine Kante $e \in T$. Seien T_1 und T_2 die Teilbäume in die T beim Entfernen von e zerfällt. In T_1 seien s_1 viele Sender und r_1 viele Empfänger, in T_2 seien s_2 viele Sender und r_2 viele Empfänger. Diese Situation ist in Abbildung 5.1 dargestellt. Nehmen wir an, dass $s_1 + r_1 \leq s_2 + r_2$ (d.h. T_1 ist der Teilbaum mit weniger Terminalen). Das Worst-Case Verkehrsszenario für Kante e sieht nun doch so aus, dass $\min\{s_1, r_2\}$ viele Sender von T_1 über e nach T_2 senden und $\min\{s_2, r_1\}$ viele Sender von T_2 über e nach T_1. Die zu reservierende Kapazität $u(e)$ auf Kante e beträgt also

$$u(e) = \min\{s_1, r_2\} + \min\{s_2, r_1\}$$

Betrachten wir zunächst den Fall $s_1 > r_2$, dann gilt für die Anzahl der Terminale in T_1, dass $s_1 + r_1 > r_1 + r_2$. Da wir stets davon ausgehen, nicht weniger Empfänger als

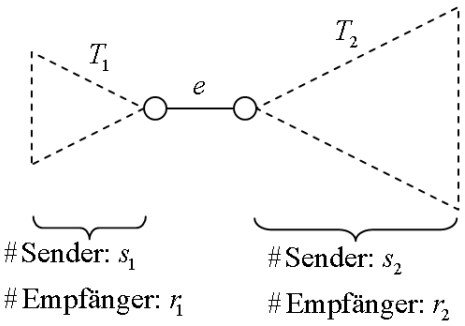


Abbildung 5.1: Darstellung der Bezeichnungen in Kapazitätsreservierung von Kante e

Sender zu haben (also $r_1 + r_2 \geq s_1 + s_2$), würde der Teilbaum T_1 mehr als die Hälfte aller Terminale enthalten. Da T_1 aber nach Voraussetzung der kleinere der beiden Teilbäume ist, folgt ein Widerspruch. Also muss stets $s_1 \leq r_2$ gelten. Es gilt also

$$u(e) = s_1 + \min\{s_2, r_1\}$$

Es sind nun zwei Fälle möglich. Falls für die Anzahl der Terminale in T_1 $s_1 + r_1 < |S|$ gilt, so ist wegen $s_1 + s_2 = |S|$ auch $r_1 < s_2$ und es folgt

$$u(e) = s_1 + \min\{s_2, r_1\} = s_1 + r_1 = \#\text{Terminale in } T_1$$

Andernfalls ist $s_1 + r_1 \geq |S|$, dann ist $r_1 \geq s_2$, also ist

$$u(e) = s_1 + \min\{s_2, r_1\} = s_1 + s_2 = |S|$$

Zusammenfassend haben wir also das folgende Lemma bewiesen.

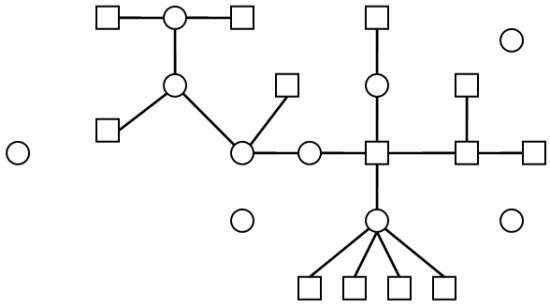
Lemma 5.1. *Sei eine VPND-Tree-Instanz aus Graph $G = (V, E)$, Sendern S und Empfängern R mit $|S| \leq |R|$ gegeben, dann gilt für die zu reservierende Kapazität $u(e)$ auf einer Kante $e \in T$ bei Lösung T*

$$u(e) = \min\{|S|, \#\text{Terminale in kleinerem Teilbaum unter } e\}$$

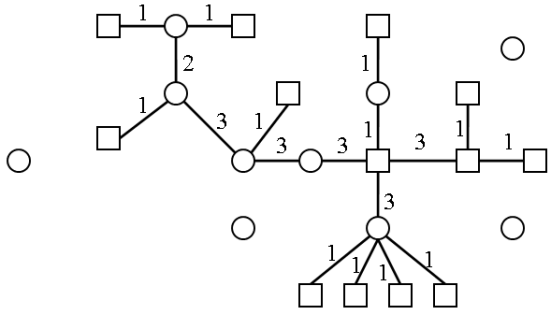
(Hierbei meinen wir mit kleinerem Teilbaum denjenigen Teilbaum von T, der unter e hängt und weniger Terminale enthält.)

Mit dieser vereinfachten Vorschrift können wir zum Beispiel den Baum T in Abbildung 5.2 (a) leicht die benötigten Kapazitäten in Abbildung 5.2 (b) herleiten. Um zu demonstrieren, dass die Kapazitäten nicht davon abhängen, welche Terminale Sender und welche Empfänger sind, haben wir die Terminale diesmal nicht weiter beschriftet. Nur die Anzahl der Sender ist noch relevant. Allgemein können wir die Kanten aus T in zwei Kategorien einteilen:

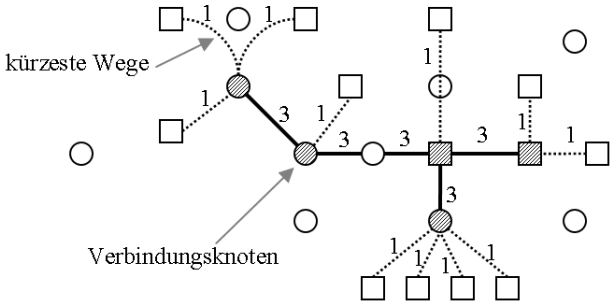
- alle Kanten mit Kapazität $|S|$



(a) Baumlösung T



(b) zu reservierende Kapazitäten



(c) Separierung in Kern und kürzeste Wege

Abbildung 5.2: Kapazitätseinteilung bei Baumlösung mit $|S| = 3$. Kantenbeschriftungen in (b) geben Kapazitäten an. Zu (c): Durchgehende Kanten gehören zu C. Gepunktete Linien sind kürzeste Wege. Schraffierte Knoten sind Verbindungsknoten.

- alle Kanten mit Kapazität $< |S|$

Man kann sich nun überlegen, dass die Kanten mit maximaler Kapazität $|S|$ zusammenhängen und somit wieder einen Baum bilden. Wir wollen diesen Teilbaum von T als *Kern* $C = \{e \in T | u(e) = |S|\}$ bezeichnen. Entfernt man den Kern C, so bleiben disjunkte Teilbäume $T_1, ..., T_k \subseteq T$ übrig, die mit dem Kern jeweils genau einen Knoten gemeinsam haben. Sei $T_i \cap C = \{v_i\}$ für alle $i \in \{1, ..., k\}$. Betrachten wir einen dieser Teilbäume T_i, der Sender S_i und Empfänger R_i enthält. Auf einer Kante e innerhalb von T_i ist die notwendige Kapazität stets kleiner als $|S|$ und entspricht der Anzahl der Terminale im Teilbaum unter e. Wir können also die Kapazität auf den Kanten innerhalb von T_i auch so interpretieren, dass von jedem Terminal aus $S_i \cup R_i$ ein separater Pfad zum Knoten v_i verläuft. Jedes Terminal in T_i hat also einen "privaten" Pfad mit einer separaten Kapazitätseinheit zum ersten Knoten auf dem Kern. Im weiteren Verlauf wollen wir v_i als *Verbindungsknoten* der Terminale in T_i bezeichnen. Bei diesen Pfaden teilen sich niemals zwei Terminale eine Kapazitätseinheit. Also können wir einen Pfad auch unabhängig von den anderen Pfaden verändern. Aber das bedeutet, dass in einer optimalen Lösung alle diese Pfade kürzeste Pfade zu v_i sein müssen.

Stellen wir uns nun vor, wir würden alle Kanten im Kern C kennen. Dann wüssten wir, dass wir auf diesen Kanten Kapazität $|S|$ installieren müssen, dann von allen Terminalen $S \cup R$ kürzeste Pfade zum nächsten Knoten in C berechnen und dort kumulativ eine Kapazitätseinheit installieren müssen. Kumulativ bedeutet dabei, dass wenn wir k viele kürzeste Pfade über eine Kante laufen lassen, auf dieser Kante Kapazität k reserviert werden muss. Die Kosten einer VPND-Tree-Lösung mit Kern C und Verbindungsknoten V' beträgt daher

$$|S| \cdot c(C) + \sum_{v \in S \cup R} \ell(v, V')$$

Der Kern C ist ein Steinerbaum, der alle Verbindungsknoten aufspannt. In einer optimalen Lösung muss C daher der optimale Steinerbaum auf den Verbindungsknoten sein.

Mit anderen Worten: Wenn wir die Verbindungsknoten in der optimalen Lösung kennen würden und wir auf diesen einen optimalen Steinerbaum berechnen könnten, so könnten wir daraus die gesamte optimale Lösung berechnen. Dies führt uns zu einem exakten Algorithmus für einen Spezialfall des VPND-Tree-Problems.

5.2 Ein Polynomialzeitalgorithmus für $|R| = |S| \pm O(1)$

Sei T^* die optimale Lösung der VPND-Instanz und C^* ihr Kern. Betrachten wir Algorithmus 3. Wie wir uns gerade überlegt haben, berechnet der Algorithmus tatsächlich eine optimale Lösung. Allerdings wird er dies im Allgemeinen nicht in polynomieller Zeit erledigen. Schritt 1 müssen wir dabei so umsetzen, dass über alle möglichen Teilmengen iteriert wird und hinterher die günstigste Lösung gewählt wird. Daher ist dieser Schritt nur effizient umsetzbar, wenn die Anzahl der Verbindungsknoten in der optimalen Lösung durch eine Konstante beschränkt ist. Schritt 2 ist, wie in Kapitel 1 erwähnt für

Algorithmus 3 Exakter Algorithmus für das VPND-Tree-Problem

Eingabe: Graph $G = (V, E)$; Kosten c; Sender $S \subseteq V$; Empfänger $R \subseteq V$

Ausgabe: Baumlösung $T \subseteq E$

1. Rate Verbindungsknoten $\rightarrow V'$
2. Berechne optimalen Steinerbaum T^* der V' aufspannt
3. Verbinde alle Terminale $S \cup R$ auf kürzesten Wegen mit T^*
4. Gebe berechneten Baum aus

$|V'| = O(\log |V|)$ noch effizient machbar. Insgesamt hat der Algorithmus nur polynomielle Laufzeit, wenn die Anzahl der Verbindungsknoten konstant ist. Dies führt uns zu der Frage: Wie viele Verbindungsknoten kann die optimale Lösung einer VPND-Tree-Instanz haben?

Lemma 5.2. *Sei eine VPND-Tree-Instanz mit Sendern S und Empfängern R gegeben. Falls $|R| = |S| + O(1)$, so gibt es $O(1)$ viele Verbindungsknoten.*

Beweis. Sei C^* der Kern der optimalen Baumlösung T^*. Dann bildet C^* einen Baum. Seien $T_1^*, ..., T_k^* \subseteq T^*$ die Teilbäume in die T^* zerfällt, wenn man C^* entfernt. An jedem Verbindungsknoten in C^* hängt einer dieser Teilbäume. Falls C^* leer ist, gibt es nur einen Verbindungsknoten und es ist nichts weiter zu zeigen. Andernfalls hat C^* als nichtleerer Baum mindestens 2 Blätter. Die Teilbäume T_i^* an Blättern von C^* müssen $|S|$ viele Terminale enthalten, sonst hätte man weniger als $|S|$ Kapazitätseinheiten auf der Kante in C^* installieren können, an der das Blatt hängt und die Lösung T^* wäre nicht optimal gewesen[1]. An jedem der mindestens 2 Blätter teilen sich daher mindestens $|S|$ viele Terminale einen Verbindungsknoten, also kann man die Anzahl der Verbindungsknoten beschränken durch

$$|S| + |R| - 2(|S| - 1) = |R| - |S| + 2 = O(1)$$

□

Im Fall, dass $|R| = |S| + O(1)$, hat Algorithmus 3 also polynomielle Laufzeit. Der Fall $|R| = |S| - O(1)$ folgt vollkommen analog. Dieser Speziallfall $|R| = |S| \pm O(1)$ ist natürlich sehr beschränkt. Sehen wir uns nun den Fall an, dass wir nicht alle Verbindungsknoten raten können − aber wenigstens die entscheidenden.

[1] Wohlgemerkt kann man über die Anzahl von Terminalen, die einen bestimmten *inneren* Knoten von C^* als Verbindungsknoten haben, keine Aussage treffen.

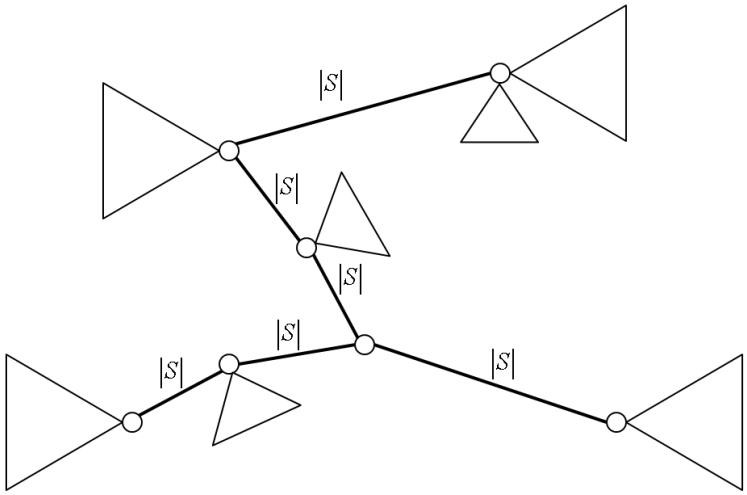


Abbildung 5.3: Darstellung der optimalen Baumlösung T^*. Die eingezeichneten Kanten sind genau diejenigen, auf denen Kapazität $|S|$ installiert ist und bilden den Kern C^*. Die Dreiecke stellen die Teilbäume dar, in die T^* zerfällt, wenn man den Kern C^* entfernt.

5.3 Ein PTAS für den Fall $|R| = O(1) \cdot |S|$

Sei erneut T^* eine optimale Baumlösung unserer VPND-Instanz, wie z.B. in Abbildung 5.3 dargestellt. Sei C^* der Kern dieser Instanz, also alle Kanten auf denen Kapazität $|S|$ reserviert werden muss. Versuchen wir nun herauszufinden, was passiert, wenn wir $2k$ viele geeignete Knoten des optimalen Kerns raten und auf diesen einen optimalen Steinerbaum aufspannen. Sei also k ein beliebiger, aber konstanter ganzzahliger Parameter. Betrachte nun Approximationsalgorithmus 4.

Da es mit höchstens n^{2k} nur polynomiell viele Möglichkeiten gibt, $2k = O(1)$ viele Knoten auszuwählen, wird die Schleife im Algorithmus nur polynomiell oft iteriert. Der optimale Steinerbaum auf V' kann ebenfalls effizient berechnet werden, da $|V'| = O(1)$. Insgesamt hat der Algorithmus eine polynomielle Laufzeit.

Zur Analyse der Güte betrachten wir den Kern C^* der optimalen Baumlösung. Da im Algorithmus alle Teilmengen der Größe $2k$ betrachtet werden, geben wir eine obere Schranke an die Approximationsgüte unseres Algorithmus, wenn wir die Kosten einer Lösung nach oben beschränken, in der die Knoten V' auf eine ganz bestimmte Art und Weise gewählt werden. Dazu hilft das folgende

Lemma 5.3. *Zu jedem Baum C^* gibt es eine Teilmenge $V' \subseteq V(C^*)$[2] mit $|V'| \leq 2k$, so dass $\ell(v, V') \leq \frac{c(C^*)}{k}$ für alle $v \in V(C^*)$.*

Beweis. Konstruiere die Menge V' mit dem folgenden Verfahren. Starte mit $V' := \emptyset$.

[2] $V(C^*)$ gibt die Knoten von C^* an

Algorithmus 4 PTAS für VPND-Tree

Eingabe: Graph $G = (V, E)$; Kosten c; $S, R \subseteq V$; Parameter $k \in \mathbb{N}$

Ausgabe: Baumlösung $T \subseteq E$

1. Für alle $V' \subseteq V$ mit $|V'| \leq 2k$:
   a) Berechne optimalen Steinerbaum C für Knoten V'
   b) Installiere Kapazität $|S|$ entlang aller Kanten von C
   c) Verbinde alle Terminale $S \cup R$ auf kürzestem Weg mit einem Knoten aus V' und installiere kumulativ eine Kapazitätseinheit
2. Gebe günstigste gefundene Lösung aus

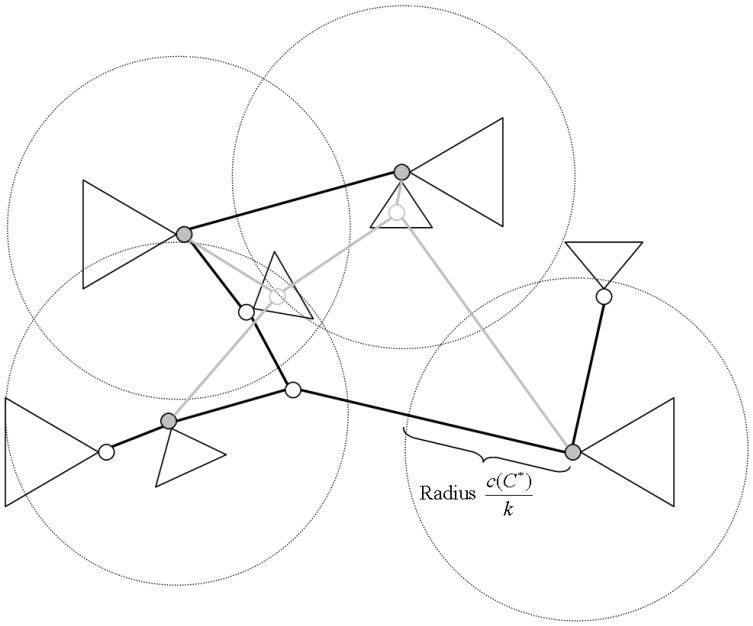


Abbildung 5.4: Darstellung der Vorgehensweise in der Analyse von Algorithmus 4. Kreise haben Radius $c(C^*)/k$. Knoten in V' sind grau. Graue Kanten gehören zu Steinerbaum C.

Solange es einen Knoten $v \in V$ gibt mit $\ell(v, V') > \frac{c(C^*)}{k}$, setze $V' := V' \cup \{v\}$. Dieses Verfahren terminiert spätestens, wenn $V' = V(C)$, denn spätestens dann ist kein Knoten mehr weiter als $\frac{c(C^*)}{k}$ von einem Knoten aus V' entfernt. Die Frage, die zu klären ist, lautet nur noch: Wieviele Knoten enthält V' am Ende?

Betrachten wir dazu diejenige Eulertour, die jede Kante des Baumes C^* genau zweimal besucht. Die Länge dieser Tour ist somit $2c(C^*)$. Natürlich besuchen wir auf dieser Tour auch jeden Knoten aus V' mindestens einmal. Wenn wir in der Tour gerade auf dem Weg zu einem Knoten $v \in V'$ sind, so bezeichnen wir dies als Annäherungsphase an v. Da sich die verschiedenen Annäherungsphasen nicht überlappen, schließen wir, dass die Länge aller Annäherungsphasen genau der Länge der Eulertour entspricht. Da die Knoten in V' per Konstruktion mindestens eine Entfernung von $\frac{c(C^*)}{k}$ zueinander haben, dauert die Annäherungsphase an jeden Knoten mindestens $\frac{c(C^*)}{k}$, also erhalten wir

$$|V'|\frac{c(C^*)}{k} \leq 2c(C^*)$$

Durch Umstellen erhalten wir $|V'| \leq 2k$. □

Eine günstige Wahl von V' und der resultierende Steinerbaum C wird in Abbildung 5.4 veranschaulicht. Wir zeigen die Güte unseres Algorithmus im folgenden

Satz 5.4. *Die Güte von Algorithmus 4 beträgt höchstens* $1 + \frac{|S|+|R|}{k|S|}$.

Beweis. Da unser Algorithmus alle Teilmengen der Größe $2k$ betrachtet, wird er auf Grund von Lemma 5.3 auch eine Menge V' finden, für die $\ell(v, V') \leq \frac{c(C^*)}{k}$ für alle $v \in V(C^*)$ gilt. Schätzen wir die Kosten für eine solche Wahl von V' ab. Wieder zerfällt T^* beim Entfernen des Kerns C^* in einzelne Teilbäume. Bezeichnen wir mit $w : S \cup R \to V$ diejenige Funktion, die zu einem Terminal seinen Verbindungsknoten liefert. Schätzen wir nun die Kosten unserer Lösung ab, durch

$$\begin{aligned}
& |S|c(C) + \sum_{v \in S \cup R} \ell(v, V') \\
\overset{(1)}{\leq} \; & |S|c(C^*) + \sum_{v \in S \cup R} \ell(v, w(v)) + \sum_{v \in S \cup R} \ell(w(v), V') \\
\overset{(2)}{=} \; & OPT + \sum_{v \in S \cup R} \ell(w(v), V') \\
\overset{(3)}{\leq} \; & OPT + |S \cup R| \cdot \frac{c(C^*)}{k} \\
\overset{(4)}{\leq} \; & OPT + |S \cup R| \cdot \frac{OPT}{k|S|} \\
= \; & OPT \left(1 + \frac{|S| + |R|}{k|S|}\right)
\end{aligned}$$

Dabei folgt (1) aus der Dreiecksungleichung und daraus, dass $c(C) \leq c(C^*)$, weil der optimale Steinerbaum C, der $V' \subseteq V(C^*)$ aufspannt, nicht teurer sein kann, als ein

Steinerbaum C^* auf $V(C^*)$. Zu (2): In der optimalen Lösung ist jedes Terminal v mit seinem Verbindungsknoten $w(v)$ über einen privaten Pfad verbunden, auf dem kumulativ eine Kapazitätseinheit für v reserviert ist, daher gilt

$$|S|c(C^*) + \sum_{v \in S \cup R} \ell(v, w(v)) = OPT.$$

Schritt (3) folgt aus unserer Voraussetzung an V', die wir aus Lemma 5.3 abgeleitet haben, die letzte Ungleichung folgt schließlich aus $|S|c(C^*) \leq OPT$. □

Wir folgern nun direkt das folgende

Korollar 5.5. *Falls $|R| \leq l|S|$ für eine Konstante $l \geq 1$, so liefert Algorithmus 4 ein PTAS.*

Beweis. Sei $\varepsilon > 0$ beliebig. Die Kosten der Lösung sind

$$OPT \cdot \left(1 + \frac{|S| + |R|}{k|S|}\right) \leq OPT \cdot \left(1 + \frac{|S| + l|S|}{k|S|}\right) = OPT \cdot \left(1 + \frac{1+l}{k}\right)$$

Wählen wir nun $k := \left\lceil \frac{1+l}{\varepsilon} \right\rceil$, so erhalten wir Kosten von höchstens $OPT \cdot (1 + \varepsilon)$, also eine Güte von $1 + \varepsilon$. □

5.4 Der Shortest-Path-Tree ist eine $\frac{|R|+|S|}{2|S|}$-Approximation

Die einfachste Approximationslösung, die man sich für alle VPND-Varianten überlegen kann, besteht im günstigsten Shortest-Path-Tree. Das bedeutet, man wählt der Reihe nach alle Knoten $v \in V$ als Wurzel und nimmt von v aus gesehen die Kanten auf kürzesten Wegen zu allen Terminalen in die Lösung auf. Dabei installiert man kumulativ pro kürzestem Weg je eine Kapazitätseinheit. Man kann zeigen, dass die Güte einer solchen Lösung höchstens $1 + |R|/|S|$ mal die Kosten einer optimalen VPND-Lösung ist [5]. Dabei werden die Kosten für alle Shortest-Path-Trees aufsummiert und argumentiert, dass der günstigste von diesen nicht teurer sein kann, als der Durchschnitt über die Kosten aller Shortest-Path-Trees.

Wenn wir jedoch wissen, dass die optimale Lösung eine Baumlösung ist, können wir gezielter argumentieren. Betrachten wir dazu eine optimale Baumlösung $T^* \subseteq E$ für unser VPND-Problem mit Kapazitäten $u : E \to \mathbb{N}_0$ und dem Kern $C^* = \{e \in T^* | u(e) = |S|\}$. Wir zeigen nun, dass ein Knoten $v^* \in V$ existiert, der als Wurzel eines Shortest-Path-Trees zu einer akzeptablen Lösung führt. Dabei hilft uns das folgende

Lemma 5.6. *In jedem gewichteten Baum $T = (V, E)$ existiert ein Knoten $v^* \in V$, so dass $\sum_{v \in V} \ell(v, v^*) \leq |V| \frac{c(T)}{2}$.*

Beweis. Falls es einen Knoten mit $\ell(v, v^*) \leq \frac{c(T)}{2}$ für alle $v \in V$ gibt, so ist nichts weiter zu zeigen. Leider muss dies aber nicht der Fall sein, wie man sich anhand von Abbildung 5.5 klarmachen kann. Dann muss es aber zumindest eine Kante $e = (u_1, u_2) \in T$ geben,

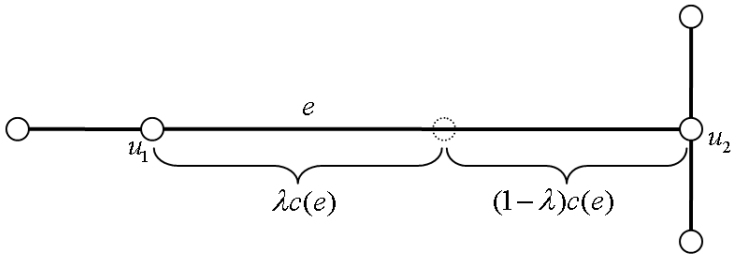


Abbildung 5.5: Darstellung eines Baumes T indem kein Knoten existiert, der höchstens $\frac{c(T)}{2}$ von jedem Knoten von T entfernt ist. Der Punkt auf der Kante $e = (u_1, u_2)$ der dies jedoch erfüllen würde, ist gepunktet eingezeichnet.

so dass beim Entfernen von e der Baum T in zwei Hälften zerfällt, welche beide Kosten $< \frac{c(T)}{2}$ haben. Bezeichnen wir die Knoten der ersten Hälfte mit V_1 und die der anderen Hälfte als $V_2 = V \backslash V_1$. Stellen wir uns die Kante e als eine Strecke der Länge $c(e)$ vor. Dann gibt es irgendwo auf dieser Strecke einen Punkt, zu dem alle anderen Knoten in T eine Entfernung von $\leq \frac{c(T)}{2}$ haben. Es sei die Entfernung von diesem Punkt zu u_1 mit $\lambda c(e)$ bezeichnet, während die Entfernung zu u_2 genau $(1-\lambda)c(e)$ ist. Setzen wir an dieser Stelle einen neuen Knoten v^* ein. Wir ersetzen e durch die Kanten (v^*, u_1) und (v^*, u_2) mit den Kosten $c(v^*, u_1) := \lambda c(e)$ und $c(v^*, u_2) := (1-\lambda)c(e)$. Für unseren künstlichen Knoten v^* gilt $\ell(v, v^*) \leq \frac{c(T)}{2}$ für alle $v \in V$. Für V_1 folgern wir daher

$$|V_1|\frac{c(T)}{2} \geq \sum_{v \in V_1} \ell(v, v^*) = \sum_{v \in V_1} \ell(v, u_1) + \lambda |V_1| c(e)$$

während für die andere Hälfte der Terminale gilt

$$|V_2|\frac{c(T)}{2} \geq \sum_{v \in V_2} \ell(v, v^*) = \sum_{v \in V_2} \ell(v, u_2) + (1-\lambda)|V_2| c(e).$$

Addieren wir beide Ungleichungen zusammen, so erhalten wir

$$\sum_{v \in V_1} \ell(v, u_1) + \lambda |V_1| c(e) + \sum_{v \in V_2} \ell(v, u_2) + (1-\lambda)|V_2| c(e) \leq (|V_1| + |V_2|)\frac{c(T)}{2}$$

Durch Umformen ergibt dies

$$\underbrace{\sum_{v \in V_1} \ell(v, u_1) + \sum_{v \in V_2} \ell(v, u_2) + |V_2| c(e)}_{\text{unabhängig von } \lambda} + \lambda c(e)\,(|V_1| - |V_2|) \leq |V|\frac{c(T)}{2} \tag{5.1}$$

Hierbei wurde $V_1 \cup V_2 = V$ verwendet. Wenn wir die linke Seite über $\lambda \in [0, 1]$ minimieren dann erreichen wir entweder für $\lambda = 0$ oder $\lambda = 1$ ein Minimum. Der erste Fall tritt ein,

wenn $|V_1| \geq |V_2|$, dann wählen wir $v^* := u_1$, sonst setzen wir $v^* := u_2$. Da die linke Seite von Ungleichung 5.1 mit $\sum_{v \in V} \ell(v, v^*)$ identisch ist, folgt in jedem Fall

$$\sum_{v \in V} \ell(v, v^*) \leq |V| \frac{c(T)}{2}$$

□

Mit diesem Lemma haben wir bereits die Hauptarbeit geleistet, um nun die Güte des Shortest-Path-Trees nachweisen zu können.

Satz 5.7. *Der günstigste Shortest-Path-Tree hat eine Güte von höchstens* $\frac{|R|+|S|}{2|S|}$ *für das VPND-Tree-Problem.*

Beweis. Sei C^* der Kern der optimalen Lösung, dann definieren wir $w(v)$ als den Verbindungsknoten eines Terminals $v \in S \cup R$ in der optimalen Lösung. Wenden wir Lemma 5.6 für den Baum C^* an und betrachten V als Multimenge, die alle Knoten von C^* so oft enthält, wie sie als Verbindungsknoten von Terminalen vorkommen. Dann erhalten wir einen Knoten v^* mit den Eigenschaften

$$\sum_{v \in R \cup S} \ell(w(v), v^*) \leq |R \cup S| \frac{c(C^*)}{2} \tag{5.2}$$

Wir schätzen nun die Kosten ab, die der Shortest-Path-Tree mit Wurzel v^* verursacht

$$\begin{aligned}
& \sum_{v \in R \cup S} \ell(v, v^*) \\
\overset{(1)}{\leq} & \sum_{v \in R \cup S} \ell(v, w(v)) + \sum_{v \in R \cup S} \ell(w(v), v^*) \\
\overset{(2)}{\leq} & \sum_{v \in R \cup S} \ell(v, w(v)) + (|R| + |S|) \frac{c(C^*)}{2} \\
= & \sum_{v \in R \cup S} \ell(v, w(v)) + 2|S| \frac{c(C^*)}{2} + (|R| - |S|) \frac{c(C^*)}{2} \\
\overset{(3)}{\leq} & OPT + (|R| - |S|) \frac{OPT}{|S|2} \\
= & OPT \frac{|R| + |S|}{2|S|}
\end{aligned}$$

In (1) wenden wir dabei die Dreiecksungleichung an, für (2) setzen wir Ungleichung 5.2 ein. (3) folgt aus

$$\sum_{v \in R \cup S} \ell(v, w(v)) + |S| c(C^*) = OPT$$

und $|S| c(C^*) \leq OPT$. □

Wir stellen uns nun die Frage, wie gut diese Abschätzung ist. Das nächste Kapitel zeigt jedoch, dass unsere Analyse nicht schärfer hätte sein können.

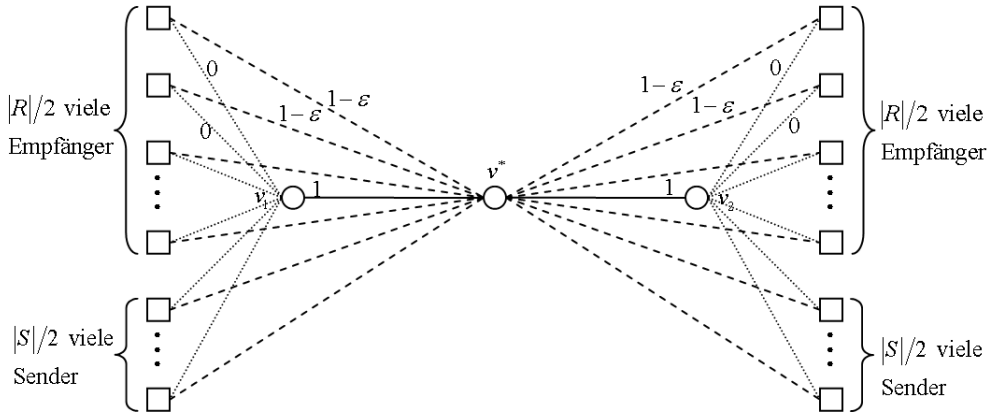


Abbildung 5.6: Worst-Case Beispiel $\mathcal{I}$ für die Shortest-Path-Tree-Lösung. Die gepunkteten Kanten haben Kosten 0, gestrichelte Kanten haben Kosten $1-\varepsilon$ und durchgezogene Kanten haben Kosten 1. Die rechteckigen Knoten sind Terminale, die übrigen Knoten sind Nichtterminale.

5.5 Ein Worst-Case Beispiel für die Shortest-Path-Tree-Lösung

Betrachten wir die in Abbildung 5.6 dargestellte Instanz $\mathcal{I}$, wobei wir voraussetzen, dass die Kardinalitäten von R und S gerade sind. Grob gesprochen enthält der dargestellte Graph eine Hälfte der Terminale auf der linken Seite und die andere Hälfte auf der rechten Seite. Zeigen wir nun die Güte der Lösung, die der Shortest-Path-Tree definiert.

Lemma 5.8. *Die Güte des Shortest-Path-Tree auf Instanz $\mathcal{I}$ beträgt $\frac{|R|+|S|}{2|S|}(1-\varepsilon)$.*

Beweis. Von v^* aus beträgt die Länge des Weges zu jedem Terminal $1-\varepsilon$, also kostet ein Shortest-Path-Tree mit v^* als Wurzel $(|R|+|S|)\cdot(1-\varepsilon)$. Für jeden anderen Knoten beträgt die Entfernung zu den Terminalen auf seiner Seite 0 und die Entfernung zu den Terminalen auf der anderen Seite $2\cdot(1-\varepsilon)$, also kostet auch hier ein Shortest-Path-Tree $\frac{|R|+|S|}{2}\cdot 2\cdot(1-\varepsilon) = (|R|+|S|)\cdot(1-\varepsilon)$. Auch der günstigste Shortest-Path-Tree erzielt somit Kosten von $(|R|+|S|)\cdot(1-\varepsilon)$.

Je größer nun R im Vergleich zu S ist, desto günstiger wird im Vergleich diejenige Lösung, in der wir auf den Kanten (v_1, v^*) und (v_2, v^*) jeweils Kapazität $|S|$ reservieren und von v_1 bzw. v_2 aus die Kanten mit Kosten 0 zu den Terminalen jeweils mit Kapazität 1 belegen. Diese Lösung hat Kosten $2|S|$. Die Güte der Shortest-Path-Tree Lösung ist daher nur $\frac{|R|+|S|}{2|S|}(1-\varepsilon)$. □

Da wir $\varepsilon > 0$ beliebig klein wählen können, ist die Güte der Shortest-Path-Tree-Lösung bezogen auf die beste Baumlösung im Allgemeinen nicht besser als $\frac{|R|+|S|}{2|S|}$ und stimmt somit mit der gezeigten oberen Schranke überein.

5.6 Ein Approximationsalgorithmus mit Güte 3.05

Erinnern wir uns an die Erkenntnis aus Kapitel 5.1. Das VPND-Tree-Problem kann man auch sehen als das Problem, die Verbindungsknoten zu wählen, einen Steinerbaum auf diesen zu konstruieren und die Sender und Empfänger auf kürzestem Weg mit einem Verbindungsknoten zu verbinden. Bezeichnen wir mit V' die Menge der Verbindungsknoten, dann können wir den Wert einer optimalen VPND-Tree-Lösung formal ausdrücken als

$$\min_{V' \subseteq V} \left\{ |S| \cdot \text{Kosten eines Steinerbaumes auf } V' + \sum_{v \in S \cup R} \ell(v, V') \right\}$$

Erinnern wir uns nun auch an das CFL-Problem, für welches Demands $D \subseteq V$ und ein Parameter M gegeben waren. Dann lautet für dieses Problem ein optimaler Zielfunktionswert

$$\min_{F \subseteq V} \left\{ M \cdot \text{Kosten eines Steinerbaumes auf } F + \sum_{v \in D} \ell(v, F) \right\}$$

Wir sehen, dass wir das VPND-Tree-Problem sehr einfach in ein Connected-Facility-Location-Problem umwandeln können, indem wir $S \cup R$ als Demands wählen und $M := |S|$ setzen. Nun können wir den Algorithmus mit Güte 3.05 aus Kapitel 2.1 auch auf das VPND-Tree-Problem anwenden und erhalten direkt einen randomisierten 3.05 Approximationsalgorithmus für das VPND-Tree-Problem.

6 Das VPND-MinCongestion-Problem

Historie und neue Ergebnisse

Zum VPND-MinCongestion-Problem sind bisher noch keine nennenswerten Ergebnisse veröffentlicht worden. Weder sind Approximationsalgorithmen mit nichttrivialen Güten bekannt, noch konnte eine untere Schranke an die Approximierbarkeit nachgewiesen werden. Nur die $\mathcal{NP}$-Vollständigkeit des Problems ist bekannt. In diesem Kapitel werden wir in der Lage sein zu zeigen, dass man das relaxierte Problem in polynomieller Zeit lösen kann. In Kapitel 6.3 werden wir eine allgemeine untere Schranke für den Zielfunktionswert einer optimalen Lösung nachweisen, die uns erlaubt einen Integrality Gap des VPND-MinCongestion-Problems von $\Omega(\sqrt{n}/\log n)$ bzgl. seiner natürlichen Relaxation zu zeigen. Dies ist ein Hinweis, dass sich das MinCongestion-Problem nicht adäquat approximieren lässt, aber noch kein formaler Beweis. Zusätzlich werden wir zeigen, dass randomisiertes Runden einer optimalen relaxierten Lösung zu einer katastrophal schlechten Güte führt.

6.1 Das relaxierte Problem ist in polynomieller Zeit lösbar

Definieren wir zunächst das relaxierte MinCongestion-Problem.

Def. Relaxiertes VPND-MinCongestion-Problem. Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit Sendern $S \subseteq V$ und Empfängern $R \subseteq V$ (mit $S \cap R = \emptyset$). Gesucht sind Pfade $P^i_{s,r}$ von jedem Sender $s \in S$ zu jedem Empfänger $r \in R$ und Gewichte $\lambda^i_{s,r} \geq 0$ mit $\sum_i \lambda^i_{s,r} = 1$ sowie Kantenkapazitäten $u : E \to \mathbb{N}_0$, so dass für jedes zulässige Verkehrsszenario die reservierte Kantenkapazität u ausreichend ist und der Ausdruck $\max\{u(e)|e \in E\}$ minimiert wird.

Analog zum VPND-Problem können wir auch hier aus berechneten Pfaden die induzierten Kantenkapazitäten wie folgt erhalten: Erstelle für $e \in E$ einen vollständigen bipartiten Graphen $G_e = (S \cup R, S \times R)$ mit Gewicht $\sum_{i \text{ mit } e \in P^i_{s,r}} \lambda^i_{s,r}$ auf Kante (s, r). Der Wert eines maximalen gewichteten Matchings gibt dann die zu reservierende Kapazität auf e an.

Stellen wir nun ein ganzzahliges Programm für das VPND-MinCongestion-Problem auf. Wir wollen als Bestandteil des ILPs die in der Literatur üblichen Flussbedingungen verwenden [15, 13]. Diese funktionieren allerdings nur auf gerichteten Graphen. Wir ersetzen daher jede ungerichtete Kante $\{u, v\}$ in unserem Graphen durch zwei gerichtete Kanten (u, v) und (v, u). Als Last auf $\{u, v\}$ definieren wir dann die Last auf (u, v) zuzüglich der Last auf (v, u).

Führen wir die folgenden ganzzahligen Variablen ein

$$\begin{aligned} x_e^{s,r} &= \begin{cases} 1 \text{ falls Kante } e \text{ auf dem Pfad von } s \in S \text{ nach } r \in R \text{ liegt} \\ 0 \text{ sonst} \end{cases} \\ y &= \text{maximale Last} \end{aligned}$$

Stellen wir nun das ganzzahlige, lineare Programm (ILP) *auf*

$$\begin{aligned} & & \min y & & \\ \sum_{e \in \delta^+(v)} x_e^{s,r} - \sum_{e \in \delta^-(v)} x_e^{s,r} &= \begin{cases} 1 \text{ falls } v = s \\ -1 \text{ falls } v = r \\ 0 \text{ sonst} \end{cases} & & \forall s \in S \; \forall r \in R \; \forall v \in V \\ \sum_{(s,r) \in \mathcal{M}} x_e^{s,r} + x_{e^{-1}}^{s,r} &\leq & y & \forall e \in E \; \forall \text{Matchings } \mathcal{M} \text{ in } S \times R \\ & & x_e^{s,r} \in & \{0,1\} & \forall e \in E \; \forall s \in S \; \forall r \in R \end{aligned}$$

Zu einer Kante $e = (u, v)$ sei $e^{-1} = (v, u)$ die Kante in Gegenrichtung. Mit $\delta^+(v)$ bezeichnen wir alle von v ausgehenden Kanten und mit $\delta^-(v)$ die eingehenden. Die Ganzzahligkeit von y muss nicht gefordert werden, sondern folgt aus der Ganzzahligkeit von $x_e^{s,r}$. Die erste Ungleichung sorgt dafür, dass für jedes Paar $(s, r) \in S \times R$ von Sendern und Empfängern ein separater Fluss der Stärke 1 von s nach r geschickt wird.

Die zweite Ungleichung besagt, dass für jedes Matching − also jedes zulässige Verkehrsszenario − und jede Kante e die maximale Last y mindestens der Last entspricht, die das Verkehrsszenario über e und die entgegengesetzte Kante e^{-1} schickt.

Bezeichnen wir mit (LP) die Relaxation von (ILP), die wir erhalten, indem wir die Bedingung $x_e^{s,r} \in \{0, 1\}$ durch $0 \leq x_e^{s,r} \leq 1$ ersetzen. Doch wir können (LP) nicht wie gewohnt lösen, denn die zweite Ungleichung sorgt für Probleme: Da es exponentiell viele Matchings auf $S \times R$ gibt, gibt es auch exponentiell viele Ungleichungen vom Typ 2. Dennoch können wir das folgende Lemma beweisen.

Lemma 6.1. *Eine optimale Lösung von (LP) kann in polynomieller Zeit bestimmt werden.*

Beweis. Wir können dieses Problem beheben, indem wir den Ellipsoidalgorithmus verwenden, denn dieser benötigt gar nicht alle Ungleichungen in expliziter Form. Er kann eine optimale Lösung bereits in polynomieller Zeit berechnen, wenn wir ihm zu einer gegebenen Lösung $(x_e^{s,r}, y)$ sagen können, ob diese Lösung zulässig ist oder wenn nicht, ihm eine verletzte Ungleichung berechnen können. Dies ist das Prinzip der *Optimierung durch Separierung* [6, 13].

Es sei also eine Lösung $(x_e^{s,r}, y)$ gegeben, für die das Separationsproblem gelöst werden soll. Verwende dazu folgendes Verfahren:

1. Überprüfe Ungleichungen vom Typ 1 und 3: Falls eine dieser polynomiell vielen Ungleichungen durch Einsetzen von $(x_e^{s,r}, y)$ nicht erfüllt wird, gebe die entsprechende Ungleichung zurück.

2. Überprüfe Ungleichungen vom Typ 2: Für alle Kanten $e \in E$ konstruieren wir einen vollständigen bipartiten Graphen $G_e = (S \cup R, S \times R)$ in dem eine Kante $(s,r) \in S \times R$ mit Gewicht $x_e^{s,r} + x_{e^{-1}}^{s,r}$ aufgeführt ist. Falls der Wert eines maximalen Matchings auf G_e größer ist als y, gebe die verletzte Ungleichung

$$\sum_{(s,r)\in\mathcal{M}} x_e^{s,r} + x_{e^{-1}}^{s,r} \leq y$$

aus, wobei $\mathcal{M}$ das maximale Matching auf G_e darstellt.

3. Falls in Schritt (1) und (2) keine verletzte Ungleichung gefunden wurde, gebe "Lösung zulässig" aus.

Da polynomiell viele Ungleichungen in polynomieller Zeit auf Zulässigkeit überprüft werden können und ein maximales Matching effizient berechnet werden kann, ist die Separierung in polynomieller Zeit durchführbar. Insgesamt folgt, dass eine optimale, Lösung des LPs effizient berechnet werden kann. □

Nun sind wir noch nicht ganz am Ziel, denn wir haben erst die Gewichte $x_e^{s,r}$ die die s-r-Pfade auf die einzelnen Kanten legen, aber noch nicht die s-r-Pfade und deren Gewichte. Definieren wir dazu einen gewichteten Graphen $G_{s,r} = (V, E_{s,r})$ mit

$$e \in E_{s,r} :\Leftrightarrow x_e^{s,r} > 0$$

und Gewicht $x_e^{s,r}$ auf Kante e. Suchen wir nun einen beliebigen Weg in $G_{s,r}$ von s nach r und bezeichnen diesen mit $P_{s,r}^1 = (s = v_1, v_2, ..., v_l = r)$. Wir setzen das Gewicht dieses Pfades auf

$$\lambda_{s,r}^1 := \min_{e\in P_{s,r}^1} \{x_e^{s,r}\} > 0$$

Danach vermindern wir das Gewicht $x_e^{s,r}$ auf allen Kanten von $P_{s,r}^1$ um $\lambda_{s,r}^1$. Kanten mit Gewicht 0 werden entfernt. Hierbei wird zumindest die Kante, bei der das Minimum angenommen wird, entfernt. Wenn wir nun analog fortfahren, erhalten wir $\leq |E|$ viele s-r-Pfade mit Gewichten $\lambda_{s,r}^i$. Dieses Prinzip der Flussdekomposition ist auch in [15] unter der Bezeichnung *Path Stripping* verwendet worden. Wir erhalten die Schlussfolgerung

Korollar 6.2. *Eine optimale Lösung des relaxierten VPND-MinCongestion-Problems kann in polynomieller Zeit berechnet werden.*

6.2 Permutationsrouting

Das VPND-MinCongestion-Problem kann man so verstehen, dass man Routingpfade zwischen Sendern und Empfängern berechnen soll, entlang derer Pakete versandt werden sollen, so dass auch im Worst-Case möglichst wenige Pakete über eine Kante verlaufen. Wir wollen nun ein wohlbekanntes Routing-Problem präsentieren, welches mit unserem Problem verwandt ist.

Def. Permutationsrouting. Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit $V = \{1, ..., n\}$. Es sollen Pfade $\mathcal{P} = \{P_{u,v} | u, v \in V, P_{u,v}$ ist u-v-Pfad$\}$ zwischen allen Knoten berechnet werden, so dass für alle Permutationen $\sigma \in S_n$ die maximale Anzahl von Schritten, die n Pakete von allen Knoten $i \in \{1, ..., n\}$ zu Knoten $\sigma(i)$ über Pfade $P_{i,\sigma(i)}$ benötigen, minimiert wird. Das Versenden der Pakete geht dabei derart von Statten, dass in einem Zeitschritt jede Kante höchstens von einem Paket passiert werden kann.

Die Unterschiede zu unserem MinCongestion-Problem bestehen darin, dass beim Permutationsrouting jeder Knoten zugleich Sender und Empfänger ist und nicht die maximale Last, sondern die Anzahl der Schritte, bis alle Pakete ihr Ziel erreicht haben, minimiert werden soll. Aber wenn über eine Kante k viele Pakete gehen, werden auch mindestens k viele Schritte benötigt, bis alle Pakete am Ziel sind, da ja pro Schritt nur ein Paket die Kante passieren kann. Die Ähnlichkeiten beider Probleme sind also größer als ihre Unterschiede. Da das Permutationsrouting in der Literatur bereits gut untersucht ist, können wir diese Ergebnisse im Folgenden leicht übertragen.

6.3 Eine allgemeine untere Schranke

Unser Ziel ist es, in Kapitel 6.4 zeigen zu können, dass das VPND-MinCongestion-Problem einen großen Integrality Gap aufweist. Dazu müssen wir jedoch zunächst etwas Vorarbeit leisten. Unsere Behauptung ist es, dass es in Graphen, in denen die Knoten nur einen geringen Grad haben, gar keine günstige Lösung für das VPND-MinCongestion-Problem geben kann. In [2] wurde gezeigt, dass deterministisches Permutationsrouting in einem Graphen mit n Knoten und Grad d mindestens Zeit $\Omega(\sqrt{n}/d^{3/2})$ benötigt, was in [11] auf $\Omega(\sqrt{n}/d)$ verbessert werden konnte. Wir werden nun den Satz aus [11] für unsere Zwecke modifizieren.

Satz 6.3. *Es sei ein Graph $G = (V, E)$ mit n Knoten, maximalem Grad d, Sendern $S \subseteq V$ und Empfängern $R \subseteq V$ gegeben, wobei $|S| = \Omega(n)$ und $|R| = \Omega(n)$ gilt. Dann beträgt der Wert jeder Lösung des VPND-MinCongestion Problems $\Omega(\sqrt{n}/d)$.*

Beweis. Es seien Pfade $P_{s,r}$ von jedem Sender zu jedem Empfänger gegeben. Betrachten wir zunächst einen festen Empfänger $r \in R$ und alle Pfade, die r als Ziel haben. Die Vereinigung dieser Pfade formt einen Graphen G_r, den wir als *Zielgraph* zu r bezeichnen (siehe Abbildung 6.1). Eine Kante in G_r, über die k viele Pfade zu r verlaufen, bezeichnen wir als *k-belastet*. Die Menge aller k-belasteten Kanten sei S_k und die Knoten, die zu k-belasteten Kanten inzident sind, seien $V(S_k)$. Im weiteren Verlauf sei stets $k \leq |S|/d$. Da auch r nur Grad $\leq d$ hat und $|S|$ viele Pfade zu r führen, muss $r \in V(S_k)$ sein. Da eine k-belastete Kante zu 2 Knoten inzident ist, gilt sicherlich $|V(S_k)| \leq 2|S_k|$.

Für einen Sender $s \in S$ betrachten wir den Pfad $P_{s,r} = (s = v_1, v_2, ..., v_{l-1}, v_l = r)$ von s nach r. Sei v_i der erste Knoten auf diesem Pfad, der zu $V(S_k)$ gehört (dennoch muss keine Kante auf dem Pfad $P_{s,r}$ k-belastet sein). Ein solcher Knoten muss existieren, da $r \in V(S_k)$. Dann ist $v_{i-1} \notin V(S_k)$ und somit ist (v_{i-1}, v_i) nicht k-belastet. Verteilen

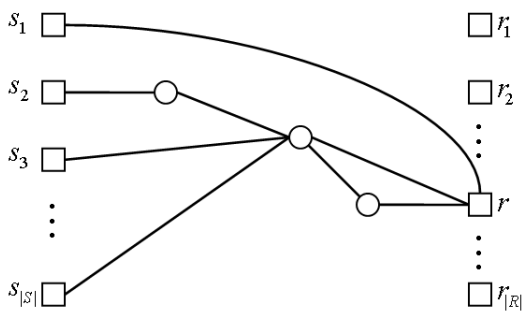


Abbildung 6.1: Zielgraph G_r zu Empfänger r

wir gedanklich die $|S|$ vielen Pfade auf die ersten Knoten, die in diesen Pfaden in $V(S_k)$ liegen. Auf jeden Knoten in $V(S_k)$ werden nicht mehr als $(k-1)d = O(kd)$ viele Pfade verteilt, da diese Pfade die Knoten nur über nicht k-belastete Kanten betreten können. Also gilt $|S| = O(kd) \cdot |V(S_k)|$. Wegen $|S| = \Omega(|V|)$ folgt

$$\begin{aligned} n &= |V| \\ &= O(kd \cdot |V(S_k)|) \\ &= O(kd \cdot |S_k|) \end{aligned}$$

Also gilt für die Anzahl der k-belasteten Kanten:

$$|S_k| = \Omega\left(\frac{n}{kd}\right)$$

Definieren wir das Gewicht einer Kante als die Anzahl von Zielgraphen, in denen die Kante k-belastet ist. Da es $|R| = \Omega(n)$ viele Zielgraphen gibt, lautet das Gesamtgewicht aller Kanten $\Omega(\frac{n^2}{kd})$. Da es höchstens nd viele Kanten gibt, muss per Schubfachprinzip mindestens eine Kante $e \in E$ Gewicht $\Omega(\frac{n^2}{kd \cdot nd}) = \Omega(\frac{n}{kd^2})$ haben. Wählen wir nun k so, dass $k = \Theta(\frac{n}{kd^2}) \Leftrightarrow k = \Theta(\frac{\sqrt{n}}{d})$, dann haben wir k maximal gewählt, so dass e in k vielen Zielgraphen k-belastet ist. Wir können also ein Verkehrsszenario konstruieren in welchem über diese Kante mindestens $k = \Omega(\frac{\sqrt{n}}{d})$ viele Pfade mit disjunkten Sendern und Empfängern gehen. Zielfunktionswert einer jeden VPND-MinCongestion-Lösung lautet also $\Omega(\frac{\sqrt{n}}{d})$. □

6.4 Integrality Gap

Um einen hohen Integrality Gap für das MinCongestion-Problem nachweisen zu können, benötigen wir einen Graphen, der

1. einen kleinen Grad hat, damit die untere Schranke aus Satz 6.3 gute Ergebnisse liefert

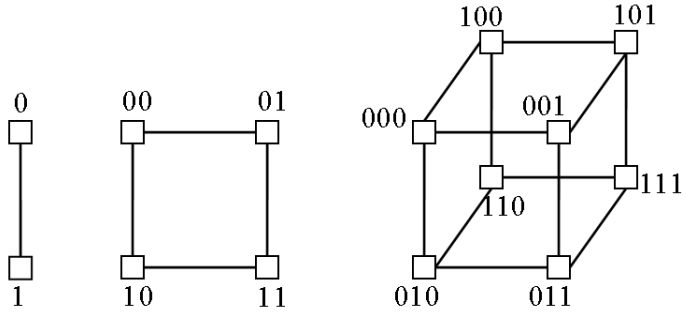


Abbildung 6.2: Hypercubes H_1, H_2 und H_3

2. eine günstige fraktionale Lösung erlaubt

Es zeigt sich, dass der wohlbekannte k-dimensionale Hypercube eben diesen Anforderungen entspricht.

Def. Hypercube. Es sei der k-dimensionale Hypercube der Graph $H_k = (V, E)$ mit $V = \{0,1\}^k$ in der $(x_1, ..., x_k), (y_1, ..., y_k) \in \{0,1\}^k$ mit einer Kante verbunden sind, genau dann, wenn sich $(x_1, ..., x_k)$ und $(y_1, ..., y_k)$ in genau einer Stelle unterscheiden (siehe Abbildung 6.2).

Wir können nun den gerade bewiesenen Satz 6.3 verwenden, um das folgende Lemma zu zeigen.

Lemma 6.4. *Seien die Knoten $S = \{(0, x_2, ..., x_k) | x_i \in \{0,1\}\}$ die Sender und $R = \{(1, x_2, ..., x_k) | x_i \in \{0,1\}\}$ die Empfänger im Hypercube $H_k = (V, E)$ mit $n = |V| = 2^k$. Dann gilt für den den Wert einer optimalen VPND-MinCongestion-Lösung auf dieser Instanz $OPT = \Omega(\sqrt{n}/\log n)$.*

Beweis. Der k-dimensionale Hypercube hat in jedem seiner n Knoten den Grad $k = \log n$. Wegen $|S| = |R| = n/2$ liefert Theorem 6.3 eine untere Schranke von $\Omega(\sqrt{n}/\log n)$. □

Bemerkung. In [11] wird auch ein deterministisches Permutationsrouting im Hypercube angegeben, welches in Zeit $O(\sqrt{n}/\log n)$ alle Pakete an ihr Ziel bringt. Verwendet man dieselben Pfade als Sender-Empfänger-Pfade für das VPND-MinCongestion-Problem, so erhält man eine Lösung, in der keine Kante mit mehr als $O(\sqrt{n}/\log n)$ belastet wird. Also gilt sogar $OPT = \Theta(\sqrt{n}/\log n)$.

Sei nun OPT_f der Wert der optimalen fraktionalen MinCongestion-Lösung auf H_k mit Sendern $S = \{(0, x_2, ..., x_k) | x_i \in \{0,1\}\}$ und Empfängern $R = \{(1, x_2, ..., x_k) | x_i \in \{0,1\}\}$. Konstruieren wir nun eine günstige fraktionale Lösung, um eine obere Schranke an OPT_f zu erhalten.

Geben wir dazu die fraktionalen Pfade zwischen $s \in S$ und $r \in R$ an. Wir gehen alle Knoten $w \in V$ als Zwischenstationen durch und setzen ein Gewicht von $\frac{1}{2^k}$ auf den Pfad $s \to w \to r$, wobei die Pfade von s bis w bzw. von w bis r die kürzesten Verbindungen mittels Bitfixing darstellen. Bitfixing-Routing von $(x_1, ..., x_k)$ zu $(y_1, ..., y_k)$ bedeutet dabei, dass der Pfad bei $(x_1, ..., x_k)$ startet, dann zu $(y_1, ..., y_{i^*}, x_{i^*+1}, ..., x_k)$ geht, wobei $i^* = \min\{i | x_i \neq y_i\}$ ist und dann von dort aus rekursiv per Bitfixing weiter zu $(y_1, ..., y_k)$ geht. Im Fall $k = 3$ würde beispielsweise der Pfad von 100 zu 010 über die Zwischenstation 111 lauten

$$100 \to 110 \to 111 \to 011 \to 010$$

Die Idee ist hier, dass wir als Gewicht für einen Pfad die Wahrscheinlichkeit setzen, mit welcher der in [14] präsentierte randomisierte Algorithmus zum Permutationsrouting diesen Pfad wählen würde. Dies erleichtert die Analyse ungemein, denn wir können nun analoge Rechnungen wie in [14] verwenden.

Lemma 6.5. *Die maximale Last der angegebenen fraktionalen Lösung auf H_k beträgt höchstens 1.*

Beweis. Betrachten wir die Belastung, die die erste Hälfte der Pfade von den Sendern bis zu den Zwischenstationen auf die Kanten legt (also nicht die vollen Pfade $s \to w \to r$, sondern nur $s \to w$). Die Zwischenstationen sind unabhängig von den Zielen der Pfade, also gilt die folgende Analyse für alle Verkehrsszenarios. Aus Symmetriegründen, lässt sich die aus der zweiten Hälfte der Pfade resultierende Last analog analysieren.

Betrachten wir eine beliebige Kante $e = ((a_1, ..., a_{j-1}, a_j, a_{j+1}, ..., a_k), (a_1, ..., a_{j-1}, 1 - a_j, a_{j+1}, ..., a_k))$ des Hypercubes. Eine Kante ist die Verbindung zweier Hammingnachbarn, die sich an genau einem Bit unterscheiden. Sei dies das Bit j. Dann lauten alle möglichen Pfade, die per Bitfixing diese Kante passieren können

$$\begin{aligned}(*, ..., *, a_j, a_{j+1}, ..., a_k) &\to (a_1, ..., a_{j-1}, a_j, a_{j+1}, ..., a_k) \\ &\to (a_1, ..., a_{j-1}, 1 - a_j, a_{j+1}, ..., a_k) \\ &\to (a_1, ..., a_{j-1}, 1 - a_j, *, ..., *)\end{aligned}$$

wobei $*$ für beliebige Werte aus $\{0, 1\}$ steht. Als mögliche Startknoten für Pfade, die diese Kante benutzen, kommen nur Knoten in Frage, deren Bits ab dem j-ten Bit mit denen des Startknotens der Kante übereinstimmen. Für das Ziel müssen die ersten j Bits mit dem Endknoten der Kante übereinstimmen. Es gibt also 2^{j-1} viele möglichen Startpunkte von denen Pfade über unsere Kante e gehen. Von jedem dieser Startpunkte aus gibt es 2^{k-j} viele Zwischenstationen zu denen der Pfad über e geht. Das Gesamtgewicht der 2^{k-j} Pfade von einem Startpunkt über e beträgt $(\frac{1}{2})^j$. Summiert über alle Startknoten ergibt sich ein Gesamtgewicht von höchstens $\frac{1}{2}$. Nehmen wir noch die bisher unterschlagene zweite Hälfte der Pfade hinzu, so erhalten wir $OPT_f \leq 1$. □

Insgesamt folgt

Korollar 6.6. *Das VPND-MinCongestion-Problem weist einen Integrality Gap von $OPT/OPT_f = \Omega(\sqrt{n}/\log n)$ bzgl. seiner natürlichen Relaxation auf.*

Die fraktionale VPND-MinCongestion-Lösung kann man so interpretieren, dass wir die Pfade nur mit der Wahrscheinlichkeit auswählen, die ihrem Gewicht entspricht. Beim nichtrelaxierten MinCongestion haben wir diese Möglichkeit der Randomisierung nicht. Dieser Unterschied entspricht dem Unterschied von randomisierten Routing zu deterministischem Routing. Dabei ist schon lange bekannt, dass deterministisches Routing zu sehr viel teureren Lösungen bzgl. der maximalen Last führt, wie randomisiertes Routing [2, 11].

Falls das VPND-MinCongestion-Problem in der Praxis auftritt, sollte man die Option prüfen, ob es die Anwendung nicht zulässt, Pfade zufällig auszuwählen. In diesem Fall könnte man entsprechende Pfade und ihre Wahrscheinlichkeiten, wie wir in 6.1 gesehen haben, optimal berechnen.

6.5 Warum randomisiertes Runden nicht funktionieren kann

Bei $\mathcal{NP}$-schwierigen Optimierungsproblemen, stellt sich zu meist die Frage, ob man denn aus einer fraktionalen Lösung nicht durch randomisiertes Runden auch eine akzeptable ganzzahlige (und damit für das Ursprungsproblem zulässige) Lösung erhalten kann. Da dieses Vorgehen in den meisten Fällen zu mehr oder weniger erfolgreichen Approximationsalgorithmen führt, sollte untersucht werden, ob diese Methode auch beim VPND-MinCongestion-Problem auf fruchtbaren Boden fällt.

Die Vereinfachung unseres Problems auf eine Variante, in der nur ein einziges Verkehrsszenario unterstützt werden muss, wird unter dem Namen MinCongestion in [15] behandelt. Der Algorithmus, der dort vorgestellt wird, verwendet randomisiertes Runden und seine Güte lässt sich durch $O(\log n / \log\log n)$ abschätzen, wobei n die Anzahl der Sender-Empfänger Paare angibt.

In Kapitel 6.4 haben wir zwar einen Integrality Gap von $\Omega(\sqrt{n}/\log n)$ nachgewiesen, aber man könnte sicherlich frohen Mutes sein, aus einer fraktionalen Lösung mit Wert OPT_f durch randomisiertes Runden eine zulässige Lösung mit Wert $O(\sqrt{n}/\log n) \cdot OPT_f \leq O(\sqrt{n}/\log n) \cdot OPT$ zu konstruieren, um wenigstens eine Approximationsgüte von $O(\sqrt{n}/\log n)$ zu erhalten.

Leider zeigt sich, dass dieser Weg zumindest in der üblichen Art und Weise nicht gangbar ist. Betrachten wir dazu den in Abbildung 6.3 dargestellten Graphen $G = (V, E)$ mit Knoten $V = \{s_1, ..., s_k, r_1, ..., r_k, u_1, ..., u_k, v_1, ..., v_k\}$ (also $n = |V| = 4k$), Sendern $S = \{s_1, ..., s_k\}$ und Empfängern $R = \{r_1, ..., r_k\}$, indem alle Knotenpaare (s_i, u_j) sowie (v_i, r_j) für alle $i, j = 1, ..., k$ mit einer Kante verbunden sind. Zudem existieren alle Kanten (u_i, v_i) für $i = 1, ..., k$. Wählen wir den Pfad $s_i \to u_i \to v_i \to r_j$ als s_i-r_j-Pfad, so liefert dies eine ganzzahlige Lösung mit Last 1, also gilt $OPT = 1$.

Betrachten wir eine fraktionale Lösung mit demselben Wert. Seien die Pfade zwischen einem Sender s_i und einem Empfänger r_j die Menge $\mathcal{P}_{ij} = \{s_i \to u_l \to v_l \to r_j | l = 1, ..., k\}$, wobei jeder dieser k Pfade ein Gewicht von $1/k$ erhalten soll. Gleich, welches Matching von k Sendern und k Empfängern wir betrachten, über jede Kante (u_i, v_i) verlaufen genau k Pfade mit Gewicht $1/k$, also ist der Wert dieser fraktionalen Lösung 1. Aber jedes Matching welches maximal ist, schickt k Flusseinheiten von den Sendern

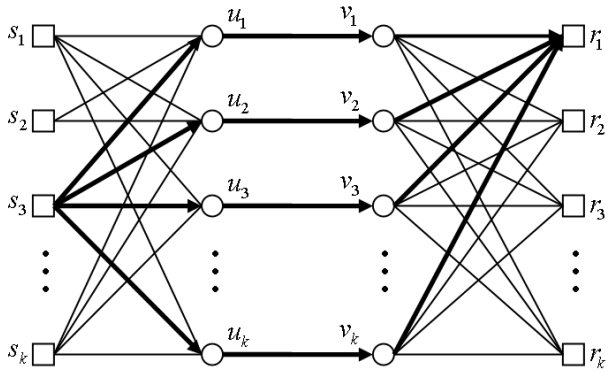


Abbildung 6.3: Worst-Case Graph für randomisiertes Runden. Die s_3-r_1 Pfade der fraktionalen Lösung mit Gewicht von jeweils $1/k$ sind dick gezeichnet.

zu den Empfängern. Da die Kanten $\{(u_i, v_i)|i = 1, ..., k\}$ einen Schnitt von k Kanten bilden, muss es per Schubfachprinzip unter diesen Kanten mindestens eine geben, über die dieses Matching einen Fluss von mindestens 1 schickt. Also kann es auch keine bessere Lösung geben und es folgt $OPT_f = 1$.

Stellen wir uns nun die Frage, wie gut diejenige Lösung ist, die man mit standardisierten, randomisierten Runden erhält.

Satz 6.7. *Wählen wir in Graph G für das Paar $(s_i, r_j) \in S \times R$ unabhängig mit Wahrscheinlichkeit $1/k$ den Pfad $s_i \to u_l \to v_l \to r_j$, dann liegt mit Wahrscheinlichkeit $1 - 2^{-\Omega(k)}$ eine Last von $\Omega(k)$ auf einer Kante.*

Beweis. Sicherlich erhalten wir eine untere Schranke an die maximale Last auf einer der Kanten, wenn wir die Last auf der Kante (u_1, v_1) nach unten beschränken. Das Problem, die Last auf (u_1, v_1) zu bestimmen, lässt sich auch wie folgt darstellen.

Stellen wir uns einen bipartiten Graph $G' = (S \cup R, E')$ vor, mit den Sendern auf der einen und den Empfängern auf der anderen Seite, der eine Kante (s_i, r_j) genau dann enthält, wenn der zufällig ausgewählte Pfad von s_i nach r_j über die Kante (u_1, v_1) führt. Wir erhalten also einen bipartiten Graphen G' mit k Knoten in jeder der beiden Partitionen, in dem jede Kante mit Wahrscheinlichkeit $1/k$ existiert. Die Kardinalität des maximalen Matchings in G' gibt dann die Last auf der Kante (u_1, v_1) (im Graphen G) an.

Wir geben nun eine untere Schranke an die Kardinalität des maximalen Matchings an, indem wir zeigen, wie wir ein Matching auswählen können, welches mit hoher Wahrscheinlichkeit genügend groß ist (aber nicht maximal sein muss). Dazu beginnen wir mit dem leeren Matching $\mathcal{M} := \emptyset$. Wir gehen dann alle Sender s_i mit $i = 1, ..., k/2$ durch und testen stets, ob eine vom i-ten Sender ausgehende Kante $(s_i, *)$ existiert, die zu keiner der schon in $\mathcal{M}$ enthaltenen Kanten adjazent ist und fügen diese eine Kante im positiven Fall zu $\mathcal{M}$ hinzu. Bezeichnen wir Empfänger, die zu keiner Kante aus $\mathcal{M}$ inzident sind,

als *freie* Empfänger. Von Schritt zu Schritt gibt es immer weniger freie Empfänger, aber da wir beim Sender $s_{k/2}$ unser Verfahren abbrechen, ist $|\mathcal{M}| \leq k/2$ und wir haben in jedem Schritt mindestens $k/2$ freie Empfänger. Modifizieren wir unser Verfahren so, dass wir stets nur die ersten $k/2$ vielen freien Empfänger als Zielknoten unserer Kanten akzeptieren, dann gibt es in jedem Schritt *genau* $k/2$ viele freie Empfänger. Die Kantenmenge $\mathcal{M}$ bleibt natürlich die ganze Zeit ein Matching, da niemals eine Kante eingefügt wird, die inzident zu einer anderen Kante aus $\mathcal{M}$ ist. Bezeichnen wir nun mit $X_i : \Omega \to \{0, 1\}$ eine Zufallsvariable mit

$$X_i := \begin{cases} 1 \text{ falls Kante } (s_i, *) \in \mathcal{M} \\ 0 \text{ sonst} \end{cases}$$

Betrachten wir einen Sender s_i mit $i \in \{1, ..., k/2\}$. Es gibt nun $k/2$ viele Kanten, die von s_i ausgehen könnten, jede Kante existiert mit einer Wahrscheinlichkeit von $1/k$. Wir können also die Wahrscheinlichkeit, dass wir im i-ten Schritt eine Kante in unser Matching aufnehmen können, nach unten abschätzen durch

$$\Pr(X_i = 1) \geq 1 - \left(1 - \frac{1}{k}\right)^{k/2} = 1 - \left(\left(1 - \frac{1}{k}\right)^{k}\right)^{1/2} \overset{(1)}{\geq} 1 - e^{-1/2} \overset{(2)}{\geq} \frac{1}{3}$$

Dabei folgt Ungleichung (1) aus $(1 - \frac{1}{k})^k \leq \frac{1}{e}$ und (2) aus $1 - e^{-1/2} \approx 0.3935.. > \frac{1}{3}$. Die Zufallsvariable $X := X_1 + ... + X_{k/2}$ gibt nun die Kardinalität des ausgewählten Matchings $\mathcal{M}$ an. Die erwartete Größe des maximalen Matchings muss daher mindestens

$$E(X) = E(X_1 + ... + X_{k/2}) = E(X_1) + ... + E(X_{k/2}) \geq \underbrace{\frac{1}{3} + ... + \frac{1}{3}}_{k/2 \text{ Summanden}} = \frac{k}{6}$$

sein. Dabei ergibt sich die zweite Gleichung aus der Linearität des Erwartungswertes und die Ungleichung aus

$$E(X_i) = 0 \cdot \Pr(X_i = 0) + 1 \cdot \Pr(X_i = 1) = \Pr(X_i = 1) \geq \frac{1}{3}$$

Da beim randomisierten Runden die Pfade in G unabhängig voneinander ausgewählt werden, ist auch die Verteilung der Kanten in unserem bipartiten Graphen G unabhängig voneinander. Also ist X die Summe unabhängig 0/1-verteilter Zufallsvariablen X_i, was eine Anwendung der Chernoff-Ungleichung möglich macht. (Um Unabhängigkeit zu erreichen, haben wir oben die Zahl der freien Empfänger auf $k/2$ begrenzt). Die hier verwendete Chernoff-Variante lautet

Satz 6.8. *(Chernoff-Schranke) Seien die Zufallsvariablen $Y_1, ..., Y_m$ unabhängig 0/1-verteilt. Dann gilt für $Y := Y_1 + ... + Y_m$ und $0 < \delta < 1$*

$$\Pr(Y \leq (1 - \delta)E(Y)) \leq e^{-E(Y)\delta^2/2}$$

Beweis. Siehe [19, 14]. □

Nun können wir den Beweis von Satz 6.7 vollenden, indem wir die Chernoff-Schranke auf die Kardinalität X unseres Matchings $\mathcal{M}$ anwenden

$$\Pr(X \leq k/12) = \Pr(X \leq (1-1/2)E(X)) \leq e^{-\frac{k}{6}\frac{1}{4}\frac{1}{2}} = e^{-k/48} = \left(\frac{1}{2}\right)^{\Omega(k)}$$

Die Schlussfolgerung aus dieser Rechnung lautet: Mit einer Wahrscheinlichkeit exponentiell nahe an 1, ist der Wert der gerundeten Lösung mindestens $k/12 = \Omega(k)$. □

Wegen $OPT = 1$ und $n = 4k$, können wir mit randomisierten Runden nur eine Approximationsgüte von $\Omega(n)$ erreichen, was asymptotisch die schlechtest mögliche Güte ist, da der Wert jeder Lösung für das VPND-MinCongestion-Problem höchstens $|S| \leq |V|$ sein kann.

In einigen Approximationsalgorithmen, die randomisiertes Runden verwenden, werden die speziellen Eigenschaften ausgenutzt, die Basislösungen von linearen Programmen haben [18]. Leider ist die hier dargestellte fraktionale Lösung *keine* solche Basislösung, d.h. sie definiert keinen Extremalpunkt des durch das LP beschriebenen Polyeders. Zeigen kann man dies z.B., indem man die Lösung als Konvexkombination anderer zulässiger Lösungen des LPs ausdrückt. Betrachten wir dazu alle $k!$ vielen Permutationen auf $\{1, ..., k\}$ und definieren für jede Permutation $\sigma \in S_k$ eine Lösung mit den Pfaden $\mathcal{P}_{ij}^{\sigma} = \{s_i \to u_{\sigma(i)} \to v_{\sigma(i)} \to r_j\}$. Wir wählen also für jedes Sender-Empfänger Paar einen einzigen Pfad mit Gewicht 1. Da die Pfade eines Senders s_i stets über dieselbe Kante $(u_{\sigma(i)}, v_{\sigma(i)})$ laufen, ist der Wert dieser Lösungen immer 1 und damit optimal. Bildet man nun den arithmetischen Durchschnitt über diese $k!$ vielen Lösungen, so wird jeder Pfad der Form $s_i \to u_l \to v_l \to r_j$ in jeder k-ten Lösung enthalten sein, also erhält er das Gewicht $1/k$. Somit erhalten wir genau die oben angegebene fraktionale Lösung bestehend aus Pfaden $\mathcal{P}_{ij}$.

Aus der Tatsache, dass randomisiertes Runden für VPND-MinCongestion in einer Katastrophe endet, sollten wir allerdings noch nicht auf eine schlechte Approximierbarkeit schließen. Schließlich gilt dasselbe für VPND. Wenn wir denselben Graphen wie eben mit denselben Pfaden verwenden, wobei die Kanten (u_i, v_i) Kosten 1 erhalten und die übrigen Kanten umsonst sind, erhalten wir durch Runden eine Lösung, in der auf jeder Kante (u_i, v_i) mit hoher Wahrscheinlichkeit Kapazität $\Omega(k)$ reserviert werden muss. Dies resultiert in einer Lösung mit Kosten $\Omega(k^2)$ die gegenüber einer optimalen Lösung mit Kosten k ebenfalls extrem schlecht ist. Dennoch gibt es für VPND Approximationsalgorithmen mit konstanter Güte. Allerdings ist dem Autor im Gegensatz zur VPND-MinCongestion-Variante kein Beispiel bekannt, welches VPND einen nichtkonstanten Integrality Gap bescheinigen würde.

Ein Ausweg aus diesem Dilemma für VPND-MinCongestion könnte die Möglichkeit des iterierten randomisierten Rundens sein. Dies sähe so aus, dass wir zunächst nur einen Pfad gemäß der berechneten Gewichte zufällig wählen, dann erneut eine optimale fraktionale Lösung (in der die Variablen des bereits gewählten Pfades konstant gesetzt sind) berechnen und wieder den nächsten Pfad zufällig wählen. Dies wird solange iteriert, bis Pfade zwischen allen Sender-Empfänger Paaren gewählt sind. Aber auch dieses Verfahren kann keine Güte liefern, die geringer ist als der Integrality Gap von $\Omega(\sqrt{n}/\log n)$.

7 Ausblick

Wie so oft im Leben wirft auch in dieser Arbeit jede Antwort wieder neue Fragen auf. Wir wollen hier noch die aufgeworfenen Fragen darstellen.

Für das Steinerbaum-Problem und das VPND-Tree-Problem sind Polynomialzeitalgorithmen im Falle konstant vieler Terminale bzw. Sender und Empfänger bekannt. Für das VPND-Problem ist die Frage der effizienten Lösbarkeit in diesem Spezialfall noch unbeantwortet. Ebenso konnten wir leider für den interessanten Fall $|R| = |S|$ die $\mathcal{NP}$-Schwierigkeit des VPND-Problems nicht nachweisen.

Am unbefriedigendsten ist sicherlich die Behandlung der VPND-MinCongestion-Variante, denn mit dem Nachweis eines hohen Integrality Gaps haben wir lediglich ein Indiz für eine schlechte Approximierbarkeit geliefert. Einen formalen Nichtapproximierbarkeitsbeweis konnten wir leider nicht angeben.

Ein weiterer äußerst interessanter Punkt ist der folgende. Es sei eine VPND-Instanz mit optimalem Zielfunktionswert OPT gegeben. Sei OPT_{Tree} der Zielfunktionswert der besten Baumlösung auf derselben Instanz. Natürlich ist $OPT \leq OPT_{\text{Tree}}$, aber die entscheidende Frage ist: Um wieviel kann OPT_{Tree} größer sein als OPT? Der beste VPND-Algorithmus, der auch eine Baumlösung liefert, hat eine Güte von 4.74 [4]. Also gilt $OPT_{\text{Tree}} \leq 4.74 \cdot OPT$. Allerdings ist uns kein Beispiel bekannt, indem das Verhältnis OPT_{Tree}/OPT auch nur annähernd diese Größenordnung aufweisen würde. Es ist eine Vermutung des Autors, dass zumindest $OPT_{\text{Tree}}/OPT \leq 2$ gilt. Leider ist es bisher noch nicht möglich gewesen, dies zu zeigen. Aber vielleicht hat der Leser beim Beweisen dieser Hypothese ja mehr Erfolg...

Literaturverzeichnis

[1] M. Bern and P. Plassmann. The steiner problem with edge lengths 1 and 2,. *Inf. Process. Lett.*, 32(4):171–176, 1989.

[2] A. Borodin and J. E. Hopcroft. Routing, merging, and sorting on parallel models of computation. *Journal of Computer and System Sciences*, 30, 1985.

[3] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[4] F. Eisenbrand and F. Grandoni. An improved approximation algorithm for virtual private network design. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 05)*, volume 16, pages 928–932, Vancouver, Canada, 2005. SIAM.

[5] F. Eisenbrand, F. Grandoni, G. Oriolo, and M. Skutella. New approaches for virtual private network designs. In *Annual International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 1151–1162, Lisboa, Portugal, 2005. Springer.

[6] M. Grötschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[7] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network: a network design problem for multicommodity flow. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 389–398, New York, NY, USA, 2001. ACM Press.

[8] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In *ACM Symposium on Theory of Computing (STOC)*, 2003.

[9] C. Hurkens, J. Keijsper, and L. Stougie. Virtual private network design: A proof of the tree routing conjecture on ring networks. In *11th Integer Programming and Combinatorial Optimization Conference*, 2005.

[10] G. Italiano, S. Leonardi, and G. Oriolo. Design of networks in the hose model. *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 65–76, 2002.

[11] C. Kaklamanis, D. Krizanc, and T. Tsantilas. Tight bounds for oblivious routing in the hypercube. In *Theory of Computing Systems, Issue 1*, volume 24, pages 223 – 232, 1991.

[12] D. R. Karger and M. Minkoff. Building steiner trees with incomplete global knowledge. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 613, Washington, DC, USA, 2000. IEEE Computer Society.

[13] B. Korte and J. Vygen. *Combinatorial Optimization - Theory and Algorithms*. Springer-Verlag, Second Edition, 2002.

[14] M. Mitzenmacher and E. Upfal. *Probability and computing*. Cambridge, 2005.

[15] P. Raghavan and C. D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[16] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. *Proc. of ACM/SIAM Symposium on Discrete Algorithms (SODA)*, 2000.

[17] C. Swamy and A. Kumar. Prima-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.

[18] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[19] I. Wegener. *Komplexitätstheorie, Grenzen der Effizienz von Algorithmen*. Springer, 2003.