

Melanie Schmidt

**Earliest Arrival Flüsse mit
mehreren Senken**

Diplomarbeit

07.01.2009 – 07.07.2009

INTERNE BERICHTE
INTERNAL REPORTS

COGA
Fakultät II
Technische Universität Berlin

Lehrstuhl II
Fakultät für Informatik
Technische Universität Dortmund

Gutachter:

Prof. Dr. Martin Skutella
Dr. Thomas Jansen

 fakultät für
informatik

GERMANY · D-44221 DORTMUND

Ich danke meinem Freund Daniel Plümpe und meinem Vater Georg Schmidt sowie Jan-Philipp Kappmeier für das Korrekturlesen verschiedener Versionen meiner Diplomarbeit. Prof. Dr. Martin Skutella danke ich für die Betreuung der Arbeit, interessante Gespräche und viele Anregungen sowie Dr. Thomas Jansen für seine Bereitschaft, die Zeitkorrektur zu übernehmen. Der nerdlichen Flurhälfte der COGA danke ich für die schöne Zeit in Berlin. Auf mein gesamtes Studium bezogen danke ich Christine Zarges für viele hilfreiche Tipps und meiner Familie für ihre Unterstützung.

Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| 1 | Einleitung | 7 |
| 2 | Dynamische Transshipments | 11 |
| 2.1 | Definitionen | 11 |
| 2.1.1 | Graphen und Netzwerke | 12 |
| 2.1.2 | Netzwerkflüsse | 15 |
| 2.2 | Netzwerke mit einer Senke | 22 |
| 2.3 | Algorithmen zur Berechnung dynamischer Transshipments (Überblick) . . . | 30 |
| 2.4 | Ein LP-basierter Existenztest | 33 |
| 3 | Dynamische Transshipments mit Nullfahrzeiten | 39 |
| 3.1 | Einleitung | 39 |
| 3.2 | Netzwerke mit Nullfahrzeiten | 43 |
| 3.3 | Existenzaussagen | 47 |
| 3.3.1 | Netzwerke mit einer Engstelle | 49 |
| 3.3.2 | In-Trees und Out-Trees | 53 |
| 3.3.3 | Charakterisierung gerichteter Graphen | 55 |
| 4 | Zusammenfassung und Ausblick | 71 |
| | Index | 74 |
| | Literaturverzeichnis | 79 |

Inhaltsverzeichnis

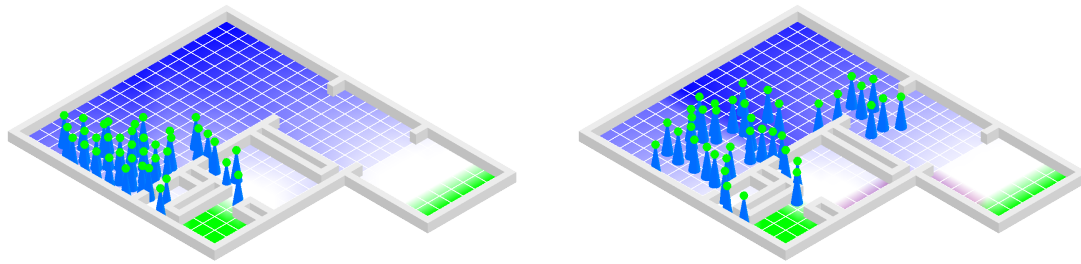
1 Einleitung

Die praktische Bedeutung von *Earliest Arrival Flüssen* besteht vor allem darin, dass sie sich hervorragend dazu eignen, Evakuierungsprobleme zu modellieren. Das liegt zum einen daran, dass es sich um *dynamische Netzwerkflüsse* handelt. *Statische* Netzwerkflüsse modellieren eine Situation, z.B. ein Gebäude, durch ein Netzwerk aus Knoten und Kanten mit Kantenkapazitäten sowie ausgezeichneten Quellen- und Senkenknoten, zwischen denen sich Flusseinheiten (z.B. Personen) bewegen. Zusätzlich dazu beziehen *dynamische* Netzwerkflüsse *Zeit* mit in die Betrachtung ein: Durch Fahrzeiten auf Kanten wird die Zeit modelliert, die Flusseinheiten benötigen, um den Weg über eine Kante zurückzulegen. Fluss fließt also nicht „direkt“ von Quellen zu Senken, sondern wird auf dem Weg aufgehalten. Durch diese Erweiterung ist im Vergleich zu statischen Flüssen eine deutlich exaktere Abbildung verschiedener Szenarien im Verkehr, in Computernetzwerken oder eben in Evakuierungssituationen möglich.

Zum anderen modelliert die *Earliest Arrival* Eigenschaft eine zentrale Anforderung an eine „bestmögliche“ Evakuierung. In einer Gefahrensituation ist es nicht nur von Bedeutung, eine Evakuierung *insgesamt* in minimaler Zeit durchzuführen. Bei ansteigender Gefahr ist es mindestens genauso wichtig, dass zu jedem Zeitpunkt der Evakuierung bereits so viele Personen den Gefahrenbereich verlassen haben, wie ihn bis zu diesem Zeitpunkt maximal verlassen haben können. Ein Earliest Arrival Fluss erfüllt genau diese Bedingung: Zu jedem Zeitpunkt ist die Flussmenge, die die Senken bereits erreicht hat, maximal.

Mit Earliest Arrival Flüssen kann also berechnet werden, wie sich Personen bei einer Evakuierung verhalten sollten, um die Anzahl evakuierter Personen zu jedem Zeitpunkt zu maximieren. Damit lässt sich einschätzen, was (in einem bestimmten, möglicherweise noch nicht gebauten Gebäude) durch Einüben von guten Fluchtwegen bestenfalls erreichbar ist. Eine spannende aktuelle Forschungsrichtung behandelt außerdem die Frage, inwieweit Earliest Arrival Flüsse zur Berechnung guter Fluchtpläne verwendet werden können. In [5] werden Earliest Arrival Flüsse dazu verwendet, abzuschätzen, wie viele Personen ein Gebäude durch welchen Ausgang verlassen sollten. Wenn sich in einem Gebäude immer ungefähr die gleichen Personen an ungefähr den gleichen Orten befinden, ist auch eine

1 Einleitung



- (a) Simulation ohne Netzwerkflussunterstützung: Alle Personen versuchen, den unteren Ausgang zu erreichen, und erzeugen einen Stau.
- (b) Simulation mit persönlichen Fluchtplänen, berechnet mit Hilfe von Earliest Arrival Flüssen: Ein Teil der Personen bewegt sich zur oberen Tür und vermeidet so Stau.

Abbildung 1.1: Diese Abbildung zeigt eine Evakuierungssimulation, in der sich Individuen nach persönlichen Fluchtplänen richten, die mit einem Earliest Arrival Fluss berechnet wurden. Es handelt sich um Screenshots der Evakuierungssoftware ZET.

Berechnung persönlicher Fluchtwege für jede einzelne Person mit Earliest Arrival Flüssen möglich (siehe Abbildung 1.1¹).

Es ist allerdings zunächst nicht selbstverständlich, dass ein Earliest Arrival Fluss überhaupt existiert. Tatsächlich ist dies aber in Netzwerken mit einer Senke immer der Fall (siehe Abschnitt 2.2). Für Netzwerke mit mehreren Senken ist die Situation leider anders. Bevor wir uns ein Beispiel dazu ansehen, gehen wir noch kurz auf einen Aspekt bei der Modellierung mit dynamischen Flüssen ein. Ein dynamischer Fluss dehnt sich zwangsläufig auf mehrere Zeitschritte aus und ist daher im Gegensatz zu einem klassischen statischen Fluss zunächst nicht auf natürliche Weise beschränkt. Damit kann man auf zwei verschiedene Weisen umgehen: Einerseits gibt es Problemstellungen, bei denen in vorgegebener Zeit eine maximale Flussmenge transportiert werden soll, d.h. der Zeithorizont des dynamischen Flusses wird begrenzt und Fluss darf nur fließen, wenn er vor Ablauf der Frist die Senken erreicht. Andererseits kann man auch die vorhandene Flussmenge begrenzen, indem man für jede Quelle die vorhandenen Flusseinheiten angibt und für jede Senke die Anzahl gewünschter Flusseinheiten festlegt. Dieses bietet sich für Evakuierungen an: Eine bestimmte Anzahl von Personen möchte von ihrem momentanen Aufenthaltsort zu einem der sicheren Orte gelangen. Solange alle sicheren Orte unbegrenzte Aufnahmekapazität haben, kann die Situation durch ein Netzwerk mit einer einzelnen Senke modelliert werden, indem wir die eigentlichen Senken mit einer Kante mit unbegrenzter Kapazität an eine sogenannte *Supersenke* anschließen. Sobald wir jedoch Ziele mit begrenzter Kapazität betrachten (wie z.B. Rettungsboote auf einem Schiff oder Innenhöfe von Gebäuden), gelangen

¹Die Open-Source-Software ZET berechnet Evakuierungen von Gebäuden mit Hilfe von sowohl Simulation als auch dynamischen Flüssen. Für weitere Informationen siehe www.zet-evakuierung.de.

wir in Situationen, in denen mehrere Senken mit konkreten Bedarfen an Flusseinheiten zur Modellierung notwendig sind.

Jetzt gehen wir auf Abbildung 1.2 ein. Diese zeigt ein Netzwerk mit einer Quelle s und zwei Senken t_1 und t_2 . In der Quelle befinden sich zwei Flusseinheiten, jede der Senken kann genau eine Flusseinheit aufnehmen. Gewünschte Einflussmengen werden hier und im Folgenden durch negative Zahlen kenntlich gemacht – Senken haben also eine negative vorhandene Flussmenge. Das Beispiel verwendet Einheitskapazitäten, d.h. in jede Kante kann zu jedem Zeitpunkt genau eine Flusseinheit hineinfließen. Die Fahrzeiten sind jeweils an den Kanten angegeben. Wir gehen davon aus, dass sich der Fluss nur zu diskreten Zeitpunkten ändern kann (damit bewegen wir uns im *diskreten Zeitmodell*), deshalb müssen wir nur diskrete Zeitpunkte betrachten.

Zum Zeitpunkt $t = 1$ kann genau eine Flusseinheit aus der Quelle in die Kante (s, a) fließen. Diese gelangt nach einer Zeiteinheit zu Knoten a (sie erreicht a also zur Zeit $t = 2$) und muss sich dort entscheiden, zu welcher Senke sie weiterfließt. Schicken wir die Flusseinheit zu Senke t_1 , so kommt sie bereits nach einer weiteren Zeiteinheit, also zum Zeitpunkt $t = 3$, in t_1 an². Was geschieht dann mit der zweiten Flusseinheit? Diese kann erst zum Zeitpunkt $t = 2$ starten und gelangt bei $t = 3$ zu Knoten a . Die obere Senke kann keinen Fluss mehr aufnehmen, d.h. die zweite Flusseinheit fließt zu t_2 und kommt dort zum Zeitpunkt $t = 5$ an. Es ist also möglich, dass zum Zeitpunkt $t = 3$ bereits eine Flusseinheit eine Senke erreicht hat, die zweite erreicht ihr Ziel dann aber erst zum Zeitpunkt $t = 5$. Alternativ können wir die erste Flusseinheit von b aus zur Senke t_2 schicken, die sie dann erst zum Zeitpunkt $t = 4$ erreicht, weil der Weg einen Zeitschritt länger dauert. Der Vorteil ist, dass die zweite Flusseinheit nun zu t_1 fließen kann und dadurch ebenfalls zum Zeitpunkt $t = 4$ ankommt. Es ist also möglich, beide Flusseinheiten bis zum Zeitpunkt $t = 4$ zu verschicken. Dadurch ist aber die Flussmenge zur Zeit $t = 3$ nicht maximal. Das bedeutet, dass es nicht möglich ist, die Anzahl der Flusseinheiten, die die Senken bereits erreicht haben, für die Zeitpunkte $t = 3$ und $t = 4$ gleichzeitig zu maximieren. Mit anderen Worten: In diesem Netzwerk gibt es keinen Earliest Arrival Fluss.

Die Betrachtung von Earliest Arrival Flüssen hat sich daher bisher auf Netzwerke mit einer Senke konzentriert. Da die Existenz der Earliest Arrival Flüsse hier gesichert ist, besteht die Herausforderung in der effizienten Berechnung oder der Entwicklung geeigneter Approximationen. In Abschnitt 2.3 gehen wir darauf genauer ein.

Diese Forschungsrichtung steht jedoch nicht im Fokus dieser Arbeit, vielmehr interessieren wir uns explizit für Netzwerke mit mehreren Senken. Betrachten wir unser Beispiel erneut:

²Es ist in gewisser Weise willkürlich, $t = 1$ als Startzeitpunkt zu wählen. An dieser Stelle wäre $t = 0$ möglicherweise intuitiver, da die erste Flusseinheit ihr Ziel dann zum Zeitpunkt $t = 2$ erreichen würde. Für die im weiteren Verlauf der Arbeit betrachteten Flüsse scheint $t = 1$ jedoch die bessere Wahl zu sein, so dass es zur Vereinheitlichung auch hier gebraucht wird.

1 Einleitung

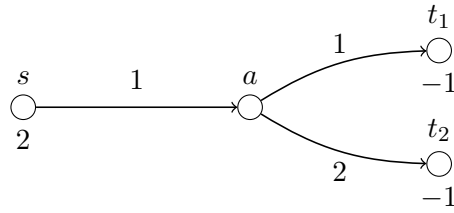


Abbildung 1.2

Die Problematik verschwindet, wenn die Fahrzeit der unteren Kante ebenfalls Eins ist. Die zweite Flusseinheit erreicht dann ihre Senke zur Zeit $t = 4$, auch wenn die erste bereits zum Zeitpunkt $t = 3$ ankommt. Ebenso würde das Problem durch eine geeignete zusätzliche Kante gelöst. Mehrere Senken müssen also nicht generell ein Problem darstellen. Diese Arbeit untersucht, *welche* Netzwerke mit mehreren Senken Earliest Arrival Flüsse ausschließen bzw. ob es Netzwerke gibt, in denen die Existenz trotz mehrerer Senken gesichert ist.

Dabei stellt sich heraus, dass es auch in dem sehr eingeschränkten Szenario, dass alle Fahrzeiten im Netzwerk Null sind, Beispiele gibt, in denen kein Earliest Arrival Fluss existiert (siehe Abschnitt 3.1). In diesem Fall legen Flusseinheiten ihren Weg ohne Verzögerung zurück und erreichen ihr Ziel in dem Zeitschritt, in dem sie gestartet sind. Dadurch zerfällt der dynamische Fluss in eine Abfolge statischer Flüsse: einen statischen Fluss für jeden Zeitschritt. Allein die Vorgabe von gewünschten Flussmengen und die Betrachtung mehrerer aufeinanderfolgender Zeitschritte reicht also aus, um Earliest Arrival Flüsse zu verhindern. (Echt positive) Fahrzeiten sind dafür nicht notwendig (bringen aber zusätzliche Schwierigkeiten). Es macht demnach Sinn, sich speziell mit dem eingeschränkten Fall zu beschäftigen, um der Frage nach der Existenz von Earliest Arrival Flüssen näher zu kommen.

Dies ist das Ziel von Kapitel 3. Wir werden dort verschiedene Fragestellungen betrachten, die sich mit der Existenz von Earliest Arrival Flüssen in Netzwerken mit Nullfahrzeiten beschäftigen. In Kapitel 2 werden zuvor dynamische Flüsse ohne diese Einschränkung untersucht. Dabei werden zuerst grundlegende Begriffe definiert, anschließend wird auf dynamische Flüsse in Netzwerken mit einer Senke eingegangen. Außerdem enthält das Kapitel einen Überblick über verwandte Arbeiten sowie einen einfachen Existenztest für Earliest Arrival Flüsse. Kapitel 4 fasst die Ergebnisse zusammen und gibt einen Überblick über offene Fragen.

2 Dynamische Transshipments

Dieses Kapitel besteht aus vier Abschnitten. Zuerst definieren wir in Abschnitt 2.1, was wir unter Graphen, Netzwerken, statischen und dynamischen Netzwerkflüssen verstehen. Insbesondere definieren wir dynamische Transshipments und Earliest Arrival Transshipments. Anschließend beschäftigen wir uns in Abschnitt 2.2 mit den Besonderheiten dynamischer Transshipments in Netzwerken mit einer Senke. Abschnitt 2.3 gibt dann einen kurzen Überblick über existierende Algorithmen zur Berechnung dynamischer Flüsse in Netzwerken mit einer oder mehreren Senken. In Abschnitt 2.4 entwickeln wir schließlich einen einfachen (pseudopolynomiellen) Test, der überprüft, ob es in einem gegebenen Netzwerk mit mehreren Quellen und Senken ein Earliest Arrival Transshipment gibt.

2.1 Definitionen

In diesem Abschnitt definieren wir Graphen und Netzwerke sowie statische und dynamische Flüsse. Dabei verfolgen wir das Ziel, alle für die später betrachteten Problemstellungen oder für andere Definitionen benötigten Aspekte abzudecken, aber die Anzahl der notwendigen Definitionen möglichst gering zu halten. Unsere Begriffe sind daher zu anderen Definitionen von Graphen, Netzwerken und Flüssen kompatibel, lassen aber bewusst viele Teilaspekte aus.

Ein ausführlicher Überblick über verschiedene Variationen statischer Flüsse findet sich z.B. in [1]. Einige der folgenden Definitionen sind außerdem an [21] und [26] angelehnt. Eine schöne Einführung in dynamische Flüsse bietet [27]. Diese verwendet jedoch das *kontinuierliche Zeitmodell*, wohingegen sich die hier verwendeten Definitionen dynamischer Flüsse nach dem *diskreten Zeitmodell* richten und daher an die in [18, 7] verwendete Notation angelehnt sind. Für einen Überblick über die Unterschiede zwischen diskretem und kontinuierlichem Zeitmodell sei auf [9] verwiesen.

2.1.1 Graphen und Netzwerke

Graphen und Netzwerke und die mit ihnen verbundenen Bezeichnungen bilden die Basis für die Definition von Flüssen. Wir werden uns sowohl mit gerichteten als auch mit ungerichteten Graphen beschäftigen. Dabei beschränken wir uns auf *einfache* Graphen: Mehrfachkanten und Schleifen werden ausgeschlossen, d.h. es gibt von einem Knoten zu einem anderen Knoten maximal eine Kante und keine Kanten von Knoten zu sich selbst.

Definition 1. Ein gerichteter Graph $G = (V, E)$ besteht aus einer Knotenmenge V und einer Kantenmenge $E \subseteq \{(u, v) \mid u, v \in V \wedge u \neq v\}$, d.h. E besteht aus geordneten Paaren verschiedener Knoten. Dabei ist $\delta^{\leftarrow}(v) := \{(u, v) \mid u \in V, (u, v) \in E\}$ die Menge der eingehenden Kanten eines Knotens $v \in V$ und $\delta^{\rightarrow}(v) := \{(v, w) \mid w \in V, (v, w) \in E\}$ die Menge der ausgehenden Kanten eines Knotens $v \in V$. Außerdem bezeichnen wir $|\delta^{\leftarrow}(v)|$ als Eingangsgrad von $v \in V$ und $|\delta^{\rightarrow}(v)|$ als Ausgangsgrad von v . Als gerichtetes Netzwerk bezeichnen wir einen Graphen, zu dessen Knoten und Kanten zusätzliche Informationen gegeben sind, zum Beispiel Kantenkapazitäten. Netzwerke bezeichnen wir üblicherweise mit N .

Welche Zusatzinformationen zu einem Netzwerk gehören, hängt vom betrachteten Problem ab. Wir kommen nach der Definition von ungerichteten Graphen und Netzwerken auf diesen Punkt zurück.

Definition 2. Ein ungerichteter Graph $G = (V, E)$ besteht aus einer Knotenmenge V und einer Kantenmenge $E \subseteq \{\{u, v\} \mid u, v \in V \wedge u \neq v\}$, d.h. E besteht aus ungeordneten Paaren verschiedener Knoten. Zwei Knoten $v, w \in V$ heißen adjazent, wenn $\{v, w\} \in E$ gilt. Ein ungerichtetes Netzwerk N ist ein ungerichteter Graph mit Zusatzinformationen zu den Knoten und Kanten.

Nicht immer möchten wir zwischen gerichteten und ungerichteten Graphen unterscheiden. Wir verwenden daher die Bezeichnung *Graph* als Oberbegriff für einen gerichteten oder ungerichteten Graphen, ebenso ist *Netzwerk* der Oberbegriff für ein gerichtetes oder ungerichtetes Netzwerk.

Definition 3. Sei $G = (V, E)$ ein Graph. Ein Teilgraph $U = (V_U, E_U)$ von G ist ein Graph mit $V_U \subseteq V$ und $E_U \subseteq E$. Sei $V' \subseteq V$ eine Teilmenge der Knotenmenge von G . Der von V' induzierte Teilgraph ist der Teilgraph $G(V') = (V', E')$ mit $E' = \{\{v, w\} \mid v, w \in V'\}$ im Falle eines ungerichteten Graphen G bzw. $E' = \{(v, w) \mid v, w \in V'\}$ im Falle eines gerichteten Graphen G .

Für den Umgang mit Netzwerken ist es hilfreich, eine einheitliche Notation zu verwenden, mit der wir das Vorhandensein verschiedener Zusatzinformationen ausdrücken können, oh-

ne jedes Mal alle genau aufzuzählen. Diese legen wir jetzt bereits fest, auch wenn wir auf den Sinn der einzelnen Informationen erst in Abschnitt 2.1.2 eingehen.

Notation 4. Wir schreiben ein Netzwerk N als 6-Tupel $(G, u, \tau, p, S^+, S^-)$. Der Graph $G = (V, E)$ ist dabei der einzige Bestandteil, der zwingend vorhanden ist. Die anderen Einträge können durch ein „-“ ersetzt werden, in diesem Fall enthält das Netzwerk die entsprechende Zusatzinformation nicht oder es ist unbedeutend, ob das Netzwerk sie enthält. Die weiteren Einträge haben folgende Bedeutung: Die Abbildung u ist eine Kapazitätsfunktion $u : E \rightarrow \mathbb{Z}_{\geq 0}$, die jeder Kante in G eine nichtnegative Kantenkapazität zuweist. Die Abbildung $\tau : E \rightarrow \mathbb{Z}_{\geq 0}$ weist jeder Kante eine nichtnegative Fahrzeit zu, weiterhin weist die Abbildung $p : V \rightarrow \mathbb{Z}$ jedem Knoten ein Potential zu. Die Quellenmenge $S^+ \subseteq V$ und die Senkenmenge $S^- \subseteq V$ sind disjunkte Teilmengen von V .

Folgende Notation dient der Abkürzung einiger oft verwendeter Schreibweisen.

Notation 5. Sei $N = (G, u, -, -, -, -)$ ein Netzwerk. Wir kürzen $|V|$ üblicherweise durch n und $|E|$ durch m ab. Außerdem lassen wir doppelte Klammern weg, wenn wir eine Kante $e = (v, w) \in E$ als Argument einer Funktion schreiben, z.B. schreiben wir $u(v, w)$ und meinen damit $u(e)$.

Bäume

Eine spezielle und wichtige Graphklasse, die uns in Abschnitt 3.3.2 besonders interessieren wird, sind *Bäume*. Dabei möchten wir für gerichtete Graphen gerne auf die Definition ungerichteter Bäume zurückgreifen können und betrachten dafür zu einem gerichteten Graphen den ungerichteten Graphen, der seine zugrundeliegende Struktur abbildet.

Definition 6. Sei $G = (V, E)$ ein gerichteter Graph. Der induzierte ungerichtete Graph $G_U = (V, E_U)$ ist der Graph, den man erhält, wenn man die Kanten von G ungerichtet interpretiert, d.h. $E_U = \{\{u, v\} \mid (u, v) \in E\}$. Da es sein kann, dass für zwei Knoten $u, v \in V$ sowohl $(u, v) \in E$ als auch $(v, u) \in E$ gilt, erweitern wir die Graphdefinition für G_U um Multikanten, indem wir zulassen, dass E_U eine Multimenge ist, in der jedes Element maximal zweimal enthalten sein kann.

Für die Definition von (ungerichteten) Bäumen müssen wir zuerst wissen, wann ein Graph *zusammenhängend* ist. Die dafür benötigte Definition von Pfaden ist so zentral, dass wir sie auch direkt für gerichtete Graphen einführen.

Definition 7. Ein Pfad von v_1 nach v_k in einem gerichteten oder ungerichteten Graphen $G = (V, E)$ ist eine Folge $P = (v_1, \dots, v_k)$, $1 \leq k \leq |V|$ paarweise verschiedener Knoten

2 Dynamische Transshipments

$v_i \in V$, so dass aufeinanderfolgende Knoten durch eine Kante verbunden sind, genauer gilt $(v_i, v_{i+1}) \in E \forall i = 1, \dots, k-1$ für gerichtete bzw. $\{v_i, v_{i+1}\} \in E \forall i = 1, \dots, k-1$ für ungerichtete Graphen. Ein ungerichteter Graph heißt zusammenhängend, wenn es für alle Knoten $u, v \in V, u \neq v$ einen Pfad von u nach v gibt. Ein gerichteter Graph G heißt ungerichtet zusammenhängend, wenn der induzierte ungerichtete Graph G_U zusammenhängend ist.

Beobachtung und Definition 8. Sei G ein ungerichteter Graph. Zusammenhang zwischen Knoten in G definiert eine Äquivalenzrelation, deren Äquivalenzklassen wir Zusammenhangskomponenten nennen.

Kreise in Graphen sind geschlossene Pfade. Wir definieren sie sowohl für ungerichtete als auch für gerichtete Graphen.

Definition 9. Ein Kreis in einem gerichteten Graphen $G = (V, E)$ ist eine Folge $C = (v_1, \dots, v_k, v_{k+1}), 2 \leq k \leq |V|$ von Knoten $v_i \in V$ mit $v_1 = v_{k+1}$, so dass v_1, \dots, v_k paarweise verschieden sind und $(v_i, v_{i+1}) \in E \forall i = 1, \dots, k-1$ gilt. Wir übertragen die Definition auf ungerichtete Graphen, indem wir $\{v_i, v_{i+1}\} \in E \forall i = 1, \dots, k$ fordern und verlangen hier zusätzlich $k \geq 3$.

Jetzt können wir ungerichtete Bäume definieren. Im gerichteten Fall interessieren uns Graphen, deren unterliegende ungerichtete Struktur ein Baum ist, und die einen ausgezeichneten Wurzelknoten besitzen.

Definition 10. Ein Baum ist ein ungerichteter Graph, der keinen Kreis enthält und zusammenhängend ist. Ein gerichteter Graph ist ein In-Tree, wenn sein induzierter ungerichteter Graph G_U ein Baum ist und jeder Knoten maximal eine ausgehende Kante besitzt. Ein gerichteter Graph ist ein Out-Tree, wenn sein induzierter ungerichteter Graph G_U ein Baum ist und jeder Knoten maximal eine eingehende Kante besitzt.

Es gibt zahlreiche äquivalente Definitionen für Bäume (und In-/Out-Trees). Insbesondere besagt [21, Theorem 2.4], dass ein Baum mit n Knoten immer genau $n-1$ Kanten besitzt. Das bedeutet, dass es in einem In-Tree genau einen Knoten ohne ausgehende Kante geben muss, ebenso gibt es in einem Out-Tree genau einen Knoten ohne eingehende Kante.

Definition 11. In einem In-Tree nennen wir den eindeutigen Knoten v mit $\delta^-(v) = 0$ Wurzel, Knoten mit $\delta^-(v) = 0$ heißen Blatt. In einem Out-Tree bezeichnen wir den eindeutigen Knoten v mit $\delta^+(v) = 0$ als Wurzel, wohingegen die Knoten mit $\delta^+(v) = 0$ Blätter sind.

Weiterhin liefert [21, Theorem 2.5], dass es in einem In-Tree für jeden Knoten einen eindeutigen Pfad zur Wurzel gibt, ebenso gibt es in einem Out-Tree für jeden Knoten einen eindeutigen Pfad von der Wurzel zum Knoten. Das führt zu folgender Definition:

Definition 12. Die Tiefe eines Knotens $v \in V$ in einem In- oder Out-Tree ist die Anzahl der Kanten auf dem eindeutigen Pfad zwischen v und der Wurzel, dabei ist die Tiefe der Wurzel Null. Als Tiefe von G bezeichnen wir die Tiefe des Knotens in V mit der größten Tiefe.

2.1.2 Netzwerkflüsse

Als nächstes werden zuerst statische und anschließend dynamische *Netzwerkflüsse* definiert. In beiden Fällen kürzen wir der Term *Netzwerkfluss* später üblicherweise durch *Fluss* ab.

Statische Netzwerkflüsse

Definition 13. Sei $N = (G, u, -, -, S^+, S^-)$ ein gerichtetes Netzwerk. Ein (statischer) Netzwerkfluss in G ist eine Abbildung $x : E \rightarrow \mathbb{R}_{\geq 0}$, die jeder Kante einen nichtnegativen Flusswert zuweist. Ein Netzwerkfluss x heißt zulässig, wenn für alle Kanten $e \in E$ $x(e) \leq u(e)$ gilt.

Fluss soll im Netzwerk von bestimmten Knoten (den Quellen) zu anderen ausgezeichneten Knoten (den Senken) fließen. Nur in Quellknoten darf Fluss entstehen, und nur in Senkenknoten darf Fluss verschwinden. An allen Zwischenknoten möchten wir, dass die Flussmenge, die in den Knoten hineinfließt, den Knoten auch wieder verlässt.

Definition 14. Sei $N = (G, u, -, -, S^+, S^-)$ ein gerichtetes Netzwerk und x ein zulässiger Fluss in N . Wir setzen

$$\text{wert}_x^{st}(v) := \sum_{e \in \delta^-(v)} x(e) - \sum_{e \in \delta^+(v)} x(e).$$

Dann respektiert x Flusserhaltung in N , wenn für alle $v \in V \setminus (S^+ \cup S^-)$ die Gleichung $\text{wert}_x^{st}(v) = 0$ sowie für alle $s \in S^+$ die Ungleichung $\text{wert}_x^{st}(s) \leq 0$ und für alle $t \in S^-$ die Ungleichung $\text{wert}_x^{st}(t) \geq 0$ gilt.

Es gibt eine Reihe interessanter statischer Flussprobleme. Das bekannteste und grundlegendste ist das Problem, einen Fluss mit maximalem Flusswert zu finden. Es wurde bereits 1956 von Ford und Fulkerson [10] eingeführt.

2 Dynamische Transshipments

Definition 15. Sei $N = (G, u, -, -, S^+, S^-)$ ein gerichtetes Netzwerk. Ein maximaler statischer Fluss in N ist ein zulässiger statischer Fluss x in N , der Flusserhaltung respektiert und für den gilt

$$\text{wert}_x^{st} = \max\{\text{wert}_{x'}^{st} \mid x' \text{ ist zulässiger Fluss in } N \text{ und respektiert Flusserhaltung}\}.$$

Den ersten Algorithmus zur Berechnung maximaler Flüsse gaben Ford und Fulkerson selbst an [10, 11]. Der Ford-Fulkerson-Algorithmus hat eine pseudopolynomielle Laufzeit, wurde aber bereits 1970 bzw. 1972 von Dinic [4] sowie Edmonds und Karp [6] (unabhängig von einander) zu einem polynomiellen Algorithmus verbessert. Bis heute sind zahlreiche Algorithmen zur Berechnung maximaler Flüsse veröffentlicht worden, unter anderem mit Laufzeiten von $o(n^3)$ [20] oder $o(mn^2)$ [13].

Dynamische Netzwerkflüsse

Dynamische Flüsse unterscheiden sich von statischen Flüssen dadurch, dass der Flusswert auf einer Kante nicht fest ist, sondern sich mit der Zeit ändern kann. Wir betrachten hier das diskrete Zeitmodell, d.h. dass sich Fluss nur zu diskreten Zeitpunkten ändert. Der Fluss auf einer Kante ist damit eine Abbildung von Zeitpunkten auf Flusswerte. Diese definieren wir vom Startzeitpunkt 1 bis zu einem Zeitpunkt T , wobei T auch unendlich sein darf.

Flusseinheiten benötigen nun auch eine gewisse Fahrzeit, um den Weg durch eine Kante zurückzulegen. Wenn der Zeithorizont T endlich ist, möchten wir nicht, dass nach dem Zeitpunkt T noch Fluss auf Kanten „unterwegs“ ist. Deshalb darf Fluss nur dann in eine Kante hineinfließen, wenn er spätestens zum Zeitpunkt T ankommt.

Definition 16. Sei $N = (G, u, \tau, -, S^+, S^-)$ ein gerichtetes Netzwerk. Ein (diskreter) dynamischer Netzwerkfluss mit Zeithorizont $T \in \mathbb{Z}_{\geq 0} \cup \{\infty\}$ ist eine Abbildung

$$f : (E \times \{1, \dots, T\}) \rightarrow \mathbb{R}_{\geq 0},$$

die jeder Kante zu jedem Zeitpunkt einen Flusswert zuweist und für $T < \infty$ außerdem $f(e, t) = 0 \forall e \in E \forall t > T - \tau(e)$ erfüllt. Wenn die Fahrzeiten für jede Kante Null sind (d.h. $\tau(e) = 0 \forall e \in E$), nennen wir f auch einen dynamischen Nullfahrzeitenfluss mit Zeithorizont T .

Den Zusatz „mit Zeithorizont T “ lassen wir im Folgenden weg, sofern wir auf den (trotzdem gegebenen) Zeithorizont eines dynamischen Flusses gerade keinen Bezug nehmen möchten. Auf die Besonderheiten dynamischer Nullfahrzeitenflüsse gehen wir in Abschnitt 3.2 genauer ein.

Jetzt möchten wir auch für dynamische Flüsse definieren, wann sie *zulässig* sind. Dies unterscheidet sich vom statischen Fall nur darin, dass wir die Kapazitäten für jede Kante auch in jedem Zeitschritt überprüfen müssen.

Definition 17. Sei $N = (G, u, \tau, -, S^+, S^-)$ ein gerichtetes Netzwerk. Dann ist ein dynamischer Fluss f mit Zeithorizont T zulässig, wenn für alle $e \in E$ und für alle $t \in \{1, \dots, T\}$ die Ungleichung $f(e, t) \leq u(e)$ erfüllt ist.

Bei der Definition der Flusserhaltung müssen wir hingegen genauer aufpassen: Fluss erreicht das Ende einer Kante erst nach einer gewissen Fahrzeit und muss *dann* den Endknoten wieder verlassen. Wir definieren zuerst den Flusswert in einem Knoten zu einem Zeitpunkt t und meinen damit den Fluss, der den Knoten zum Zeitpunkt t über eingehende Kanten erreicht, abzüglich des Flusses, der ihn zu diesem Zeitpunkt wieder verlässt¹. Mit diesem Grundbaustein definieren wir weitere Flusswerte, indem wir den Fluss zum einen für alle Quellen und zum anderen über den gesamten Zeithorizont addieren.

Definition 18. Sei $N = (G, u, \tau, -, S^+, S^-)$ ein gerichtetes Netzwerk und sei f ein zulässiger dynamischer Fluss in G mit Zeithorizont T . Für einen Knoten $v \in V$ und einen Zeitpunkt $1 \leq t \leq T$ definieren wir den Flusswert in v zur Zeit t durch

$$\text{wert}_f(v, t) := \sum_{e \in \delta^-(v) \wedge t - \tau(e) \geq 1} f(e, t - \tau(e)) - \sum_{e \in \delta^+(v)} f(e, t).$$

Aufbauend darauf definieren wir außerdem

- $\text{wert}_f(t) := \sum_{s \in S^+} \text{wert}_f(s, t)$, den Flusswert von f zur Zeit t ,
- $\text{wert}_f(v) := \sum_{t=1}^T \text{wert}_f(v, t)$, den Gesamtflusswert an $v \in V$ und
- $\text{wert}_f := \sum_{s \in S^-} \text{wert}_f(s)$, den (gesamten) Flusswert von f .

Mit Hilfe der Flusswerte können wir jetzt bequem definieren, was wir unter Flusserhaltung verstehen. Wir haben die Flusswerte so definiert, dass sie negativ sind, wenn mehr Fluss aus dem Knoten fließt als hinein, und positiv, wenn der Fluss in den Knoten größer ist als der Fluss aus dem Knoten. Daher verlangen wir bei der Definition der Flusserhaltung für die Quellen, dass der Gesamtflusswert nicht positiv ist, ebenso darf der Gesamtflusswert bei den Senken nicht negativ sein. Für Knoten, die weder Quelle noch Senke sind, fordern wir, dass in jedem einzelnen Zeitschritt die gesamte eingehende Flussmenge den Knoten wieder verlässt.

¹Hier ist also insbesondere *nicht* der Fluss *bis* zum Zeitpunkt t gemeint, sondern der Fluss *zur* Zeit t .

2 Dynamische Transshipments

Definition 19. Sei $N = (G, u, \tau, -, S^+, S^-)$ ein gerichtetes Netzwerk und sei f ein zulässiger dynamischer Fluss in G . Dann respektiert f Flusserhaltung in N , wenn gilt:

$$\begin{aligned} \text{wert}_f(v, t) &= 0 \quad \forall v \in V \setminus (S^+ \cup S^-) \quad \forall t = 1, \dots, T \\ \text{wert}_f(v) &\leq 0 \quad \forall v \in S^+ \\ \text{wert}_f(v) &\geq 0 \quad \forall v \in S^-. \end{aligned}$$

Damit wissen wir, was wir unter einem zulässigen dynamischen Fluss verstehen, der Flusserhaltung respektiert. Den Flusswert von f haben wir als Flusswert in alle Senken definiert. Das folgende Lemma zeigt auf, dass wir den Flusswert aufgrund der Flusserhaltung ebenso gut an den Quellen messen können, sofern wir das Vorzeichen umkehren.

Lemma 20. Sei $N = (G, u, \tau, -, S^+, S^-)$ ein gerichtetes Netzwerk und sei f ein dynamischer Fluss in G , der Flusserhaltung respektiert. Dann gilt $\sum_{s \in S^+} \text{wert}_f(s) = - \sum_{t \in S^-} \text{wert}_f(t)$.

Beweis. Der Beweis folgt fast direkt aus folgender Beobachtung: Wenn man den Flusswert aller Knoten zu einem Zeitpunkt addiert, wird der Fluss auf jeder Kante einmal positiv und einmal negativ gezählt. Dadurch addieren sich die Flusswerte zu Null, d.h. es gilt

$$\forall t \in \{1, \dots, T\} : \sum_{v \in V} \text{wert}_f(v, t) = \sum_{v \in V} \sum_{e \in \delta^-(v) \wedge t - \tau(e) \geq 1} f(e, t - \tau(e)) - \sum_{e \in \delta^+(v)} = 0$$

Außerdem gilt $\text{wert}_f(v) = 0$ für alle Knoten $v \in V \setminus (S^+ \cup S^-)$ und damit folgt schließlich

$$\sum_{s \in S^+} \text{wert}_f(s) = \sum_{v \in V \setminus S^-} \text{wert}_f(v) - \sum_{s \in S^-} \text{wert}_f(s) = - \sum_{s \in S^-} \text{wert}_f(s).$$

□

Analog zu statischen Flüssen kann man auch für dynamische Flüsse nach einem *maximalen* Fluss suchen. Die Berechnung eines Flusses mit maximalem Flusswert ist in polynomieller Zeit möglich, den ersten Algorithmus hierfür haben bereits Ford und Fulkerson vorgeschlagen [10, 11, 27]².

Definition 21. Sei $N = (G, u, \tau, -, S^+, S^-)$ ein gerichtetes Netzwerk. Ein maximaler dynamischer Fluss in N ist ein zulässiger dynamischer Fluss f in N , der Flusserhaltung respektiert und für den gilt

$$\text{wert}_f = \max\{\text{wert}_{f'} \mid f' \text{ ist zulässiger (dyn.) Fluss in } N \text{ und respektiert Flusserhaltung}\}.$$

²Der Algorithmus verwendet einen Algorithmus zur Berechnung von statischen *Minimalkostenflüssen*, und seine Laufzeit wird von dieser Berechnung dominiert.

Jetzt beschäftigen wir uns noch mit dem Aspekt, wie man für Knoten bestimmte Ausfluss- und Einflussmengen vorgibt. In Notation 4 haben wir dafür bereits eine Abbildung $p : V \rightarrow \mathbb{Z}$ vorgesehen, die jedem Knoten ein Potential zuweist. Potentiale sind eine alternative Möglichkeit, Quellen und Senken vorzugeben und zusätzlich für jede Quelle bzw. Senke die vorhandene Flussmenge bzw. den Flussbedarf festzulegen. Wir tun dies in gewisser Weise invers zu den Flusswerten: Wenn wir einem Knoten positives Potential zuweisen, dann liegt dort ein Überschuss an Fluss vor – aus dem Knoten soll deshalb mehr herausfließen, als hineinfließt, d.h. der von uns definierte Flusswert soll negativ sein. Wenn in einem Knoten Flussmangel herrscht (das Potential ist negativ), dann wünschen wir an diesem Knoten einen positiven Flusswert. Wir stellen uns vor, dass der Fluss von Knoten mit positivem Potential zu Knoten mit negativem Potential fließt. Dadurch legen wir implizit auch die Quellen und Senken fest.

Definition 22. Sei $N = (G, u, \tau, p, -, -)$ ein gerichtetes Netzwerk. Wir nennen Knoten $s \in V$ mit $p(s) > 0$ Quelle und bezeichnen $|p(s)|$ in diesem Fall als Angebot von s . Knoten $t \in V$ mit $p(t) < 0$ heißen Senke, hier nennen wir $|p(t)|$ den Bedarf von t .

Wenn wir zu einem Netzwerk sowohl p als auch S^+ und S^- als Zusatzinformationen angeben, achten wir darauf, dass sich diese nicht widersprechen. Dazu definieren wir, wann p , S^+ und S^- konsistent sind.

Definition 23. Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk. Wir nennen p und (S^+, S^-) konsistent, wenn gilt:

$$\begin{aligned} p(v) &= 0 \quad \forall v \in V \setminus (S^+ \cup S^-) \\ p(v) &> 0 \quad \forall v \in S^+ \\ p(v) &< 0 \quad \forall v \in S^-. \end{aligned}$$

Da es zu einem gegebenen p nur eine konsistente Wahl von Quellen- und Senkenmenge gibt, können wir den oben definierten Begriff der Flusserhaltung direkt auf Potentiale übertragen.

Definition 24. Sei $N = (G, u, \tau, p, -, -)$ ein Netzwerk und f ein zulässiger dynamischer Fluss in N . Dann erfüllt f Flusserhaltung bzgl. p , wenn f Flusserhaltung bzgl. der zu p konsistenten Quellen- und Senkenmengen erfüllt.

Flüsse, die nicht nur Flusserhaltung bzgl. eines Potentials erfüllen, sondern alle vorgegebenen Flusswerte genau erfüllen, heißen *Transshipments*. Diese werden weiter unten definiert. Vorher stellen wir jedoch sicher, dass wir nur Potentiale p verwenden, die tatsächlich ausgeglichen werden können: Flussüberschüsse und Flussbedarfe müssen sich gleich sein, damit man sie unter Beachtung von Flusserhaltung vollständig erfüllen kann.

2 Dynamische Transshipments

Definition 25. Sei $N = (G, u, \tau, p, -, -)$ ein gerichtetes Netzwerk. Wir nennen p zulässig, wenn $\sum_{v \in V} p(v) = 0$ gilt.

Da wir nur zulässige und konsistente Potentiale verwenden möchten, erweitern wir Notation 4 um zusätzliche Voraussetzungen.

Notation 26 (erweitert Notation 4). Wenn wir angeben, dass ein Netzwerk eine Funktion p als Zusatzinformation besitzt, dann ist p zulässig. Wenn wir ein Netzwerk angeben, das sowohl Quellen- und Senkenmengen S^+ und S^- als auch eine Funktion p als Zusatzinformationen besitzt, dann sind p und (S^+, S^-) konsistent.

Jetzt können wir den für diese Arbeit zentralen Begriff eines *Transshipments* definieren. Transshipments sind nur für Netzwerke mit Potentialen definiert und erfüllen an jedem Knoten die von seinem Potential vorgegebene Flussmenge.

Definition 27. Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk. Wir nennen einen dynamischen Fluss f in N ausgleichend, wenn für alle $v \in V$ $|\text{wert}_f(v)| \leq |p(v)|$ gilt³ und f zulässig ist und Flusserhaltung respektiert. Außerdem nennen wir einen ausgleichenden Fluss f vollständig ausgleichend, wenn $\forall v \in V$ $\text{wert}_f(v) + p(v) = 0$ gilt. Einen vollständig ausgleichenden dynamischen Fluss mit Zeithorizont T bezeichnen wir auch als dynamisches Transshipment mit Zeithorizont T . Ein dynamisches Transshipment mit Zeithorizont T in einem Netzwerk mit Nullfahrzeiten bezeichnen wir auch als dynamisches Transshipment mit Nullfahrzeiten oder dynamisches Nullfahrzeitentransshipment.

Bei der Definition eines ausgleichenden Flusses müssen wir nicht explizit fordern, dass das Vorzeichen von $\text{wert}_f(v)$ für Knoten $v \in V$ invers zum Vorzeichen von $p(v)$ ist, da dieses bereits durch die Flusserhaltung sichergestellt wird. Wir müssen nur dafür sorgen, dass die Flussmenge, die den Knoten verlässt bzw. in den Knoten hineinfließt, den durch das Potential vorgegebenen Wert nicht übersteigt. Bestenfalls erfüllt ein dynamischer Fluss f , dass sich $\text{wert}_f(v)$ und $p(v)$ für alle Knoten genau ausgleichen (wir sagen, f erfüllt alle Potentiale oder *gleicht sie aus*). In diesem Fall handelt es bei dem Fluss um ein Transshipment.

Wenn wir einen bestimmten Zeithorizont vorgeben, ist aber nicht unbedingt garantiert, dass ein dynamischer Fluss mit diesem Zeithorizont existiert, dem dies gelingt. Wenn es einen Fluss mit Zeithorizont T gibt, der alle Potentiale erfüllt, dann sprechen wir davon, dass die Potentiale p bis zum Zeitpunkt T vollständig ausgleichbar sind.

³Diese Bezeichnung ist von der Vorstellung inspiriert, dass die Potentiale ein Ungleichgewicht im Netzwerk ausdrücken, das durch den Fluss ausgeglichen werden soll. Sie ist für den Nullfluss sprachlich nicht ganz korrekt, da dieser nicht tatsächlich „ausgleicht“.

Sofern man vorgegebene Potentiale mit irgendeinem Zeithorizont vollständig ausgleichen kann, ist es naheliegend, nach einem Fluss zu fragen, der dies besonders schnell tut. Solche Flüsse heißen *Quickest Transshipments*:

Definition 28. Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk. Ein Quickest Transshipment ist ein dynamisches Transshipment mit minimalem Zeithorizont T^* , d.h. für alle $1 \leq T < T^*$ gilt, dass es kein dynamisches Transshipment mit Zeithorizont T gibt. Ein Quickest Transshipment in einem Netzwerk mit Nullfahrzeiten bezeichnen wir auch als Quickest Transshipment mit Nullfahrzeiten.

Die *Earliest Arrival* Eigenschaft führt eine Maximalitätseigenschaft für ausgleichende dynamische Flüsse ein. Diese ist erfüllt, wenn zu jedem Zeitpunkt so viele Flusseinheiten wie möglich in den Senken eintreffen – ohne dass jedoch die vorgegebenen Flussmengen überschritten werden, da wir ja einen ausgleichenden Fluss wünschen.

Definition 29. Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk. Mit $w_{\max}^+(t)$ bezeichnen wir für $t \geq 0$ den maximalen Wert, den ein ausgleichender Fluss nach t Zeitschritten maximal erreicht, d.h.

$$w_{\max}^+(t) := \max \left\{ \sum_{t'=1}^t \text{wert}_{f'}(t') \mid f' \text{ ist ausgleichender Fluss in } N \right\}.$$

Ein Earliest Arrival Transshipment ist ein dynamisches Transshipment f , das für jeden Zeitpunkt $t \geq 1$ die Gleichung $\sum_{t'=1}^t \text{wert}_f(t') = w_{\max}^+(t)$ erfüllt. Ein Earliest Arrival Transshipment in einem Netzwerk mit Nullfahrzeiten bezeichnen wir auch als Earliest Arrival Transshipment mit Nullfahrzeiten.

Da wir für die Definition von Earliest Arrival Transshipments ausgleichende Flüsse als Grundlage nehmen (anstatt von z.B. dynamischen Transshipments), müssen Earliest Arrival Transshipments vorgegebene Potentiale nicht vollständig ausgleichen – sie müssen „nur“ in jedem Zeitschritt so viel Fluss verschicken, wie möglich. Normalerweise bedeutet dies, dass ein Earliest Arrival Transshipment irgendwann alle Potentiale ausgleicht, und zwar zum frühest möglichen Zeitpunkt: Wenn der maximale Flusswert eines ausgleichenden Flusses ($p(S^-) := \sum_{t \in S^-} p(t)$) tatsächlich erreichbar ist, gibt es einen minimalen Zeitpunkt T^* ,

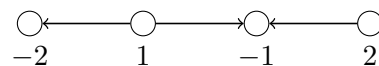


Abbildung 2.1: Ein gerichtetes Netzwerk mit Einheitskapazitäten und -fahrzeiten, das unabhängig vom Zeithorizont kein dynamisches Transshipment besitzt. Ein Earliest Arrival Transshipment existiert, es versendet im ersten Zeitschritt je eine Flusseinheit auf den beiden äußeren Kanten.

2 Dynamische Transshipments

zu dem ihn ein ausgleichender Fluss $p(S^-)$ erreichen kann. Es gilt also $w_{\max}^+(T^*) = p(S^-)$, so dass jedes Earliest Arrival Transshipment zum Zeitpunkt T^* den Wert $p(S^-)$ erreicht. Ein Earliest Arrival Transshipment ist damit in diesem Fall insbesondere auch ein Quickest Transshipment. Falls $p(S^-)$ aber für keinen Zeithorizont erreicht werden kann (siehe Abbildung 2.1), dann gibt es zwar kein dynamisches Transshipment und auch kein Quickest Transshipment, das Earliest Arrival Transshipment ist aber weiter wohldefiniert und kann existieren, falls die einzelnen Werte von $w_{\max}^+(t)$ für $t \geq 0$ von einem einzigen ausgleichenden Fluss gleichzeitig erreicht werden. Ein solches Earliest Arrival Transshipment ist *kein* dynamisches Transshipment.

2.2 Netzwerke mit einer Senke

In diesem Abschnitt geht es um die bereits angekündigte Tatsache, dass es in Netzwerken mit einer Senke immer ein Earliest Arrival Transshipment gibt. Diese Aussage wurde für Netzwerke, die zusätzlich auch nur eine Quelle haben, zuerst von Gale [12] bewiesen. Wir sehen uns aber einen Beweis an, der auf der Arbeit von Minieka [24] beruht und sich auch auf Netzwerke mit mehreren Quellen und einer Senke anwenden lässt. Dafür benötigen wir zwei wichtige Zutaten: Zum einen werden wir *zeitexpandierte Netzwerke* definieren, die dazu dienen, (diskrete) dynamische Flüsse auf statische Weise darzustellen. Zum anderen benötigen wir *lexikographisch maximale* Flüsse, aus deren Existenz dann die Existenz von Earliest Arrival Flüssen folgt.

Bei zeitexpandierten Netzwerken handelt es sich um ein generell wichtiges Werkzeug, um dynamische Flüsse in eine statische Form zu bringen, so dass man Algorithmen und Existenzaussagen von statischen auf dynamische Flüsse übertragen kann. Die Idee der *Zeitexpansion* ist es, für jeden der (diskreten) Zeitschritte eine Kopie des Netzwerkes anzulegen. Knoten werden dabei für jeden Zeitschritt kopiert – Kanten beginnen in dem Zeitschritt, zu dem sie gehören, und enden in einem ihrer Fahrzeit entsprechenden späteren Zeitschritt.

Definition 30. Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk und $T < \infty$ ein Zeithorizont. Den zeitexpandierten Graphen $G^T = (V^T, E^T)$ zu G definieren wir durch:

$$V^T := \{v^t \mid v \in V, t \in \{1, \dots, T\}\}$$

$$E^T := \{(u^t, v^{t+\tau(u,v)}) \mid (u, v) \in E, t \in \{1, \dots, T\} \text{ und } (t + \tau(u, v)) \leq T\}$$

Die Kante $(u^t, v^{t+\tau(u,v)})$ nennen wir auch t -te Kopie der Kante $e = (u, v)$ und bezeichnen sie durch e^t .

Wenn die Fahrzeit einer Kante so groß ist, dass für bestimmte Startzeitschritte der Endknoten oberhalb der T -ten Zeitscheibe liegen würde, so wird die Kante nicht in den zeitex-

pandierten Graphen aufgenommen. Das liegt daran, dass wir einen dynamischen Fluss mit Zeithorizont T modellieren wollen, und dabei auf Kanten kein Fluss gesetzt werden darf, der nicht vor Ende des Zeithorizonts ankommt (siehe Definition 16).

Der zeitexpandierte Graph ist in dieser Form noch unvollständig und enthält nur die Knoten und Kanten, die man für eine Zeitexpansion immer benötigt. Wir werden dieses Grundgerüst nun für den Fall von Netzwerken mit einer Senke so erweitern, wie es für unser Vorhaben, die Existenz von Earliest Arrival Transshipments mit Hilfe von lexikographisch maximalen Flüssen zu beweisen, hilfreich ist.

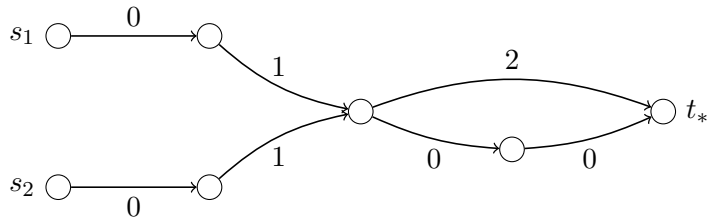
Dazu führen wir eine Superquelle s_* ein und verbinden diese mit den jeweils ersten Kopien der ursprünglichen Quellen. Die Kapazität einer solchen Kante entspricht dem Angebot der ursprünglichen Quelle, auf die sie zeigt. Für alle anderen Kanten im Netzwerk übernehmen wir die Kapazitäten der Kanten, deren Kopien sie sind. Die Superquelle ist die einzige Quelle unseres neuen Netzwerkes, während wir alle Kopien der (einzigen) Senke in die Senkenmenge aufnehmen.

Zusätzlich führen wir *Holdover*-Kanten mit unendlicher Kapazität ein, die modellieren, dass Fluss in Quellen wartet, bevor er fließt. Da nur s_* Quelle des neuen Netzwerkes ist, ist dies notwendig, damit (beliebig viel) Fluss zu den verschiedenen Kopien der Quellen gelangen und von dort aus „losfließen“ kann. Wenn man zeitexpandierte Netzwerke in anderem Zusammenhang verwendet, fügt man oft auch Holdover-Kanten für die Senken ein. Darauf verzichten wir hier, da wir jede Kopie der (einzigen) Senke als eigene Senke betrachten. Fluss kann also auch ohne Holdover-Kanten in jedem Zeitschritt in der entsprechenden Kopie der Senke verschwinden. In Abschnitt 2.4 werden wir hingegen eine Version des zeitexpandierten Netzwerkes benutzen, die auch an den Senken Holdover-Kanten verwendet.

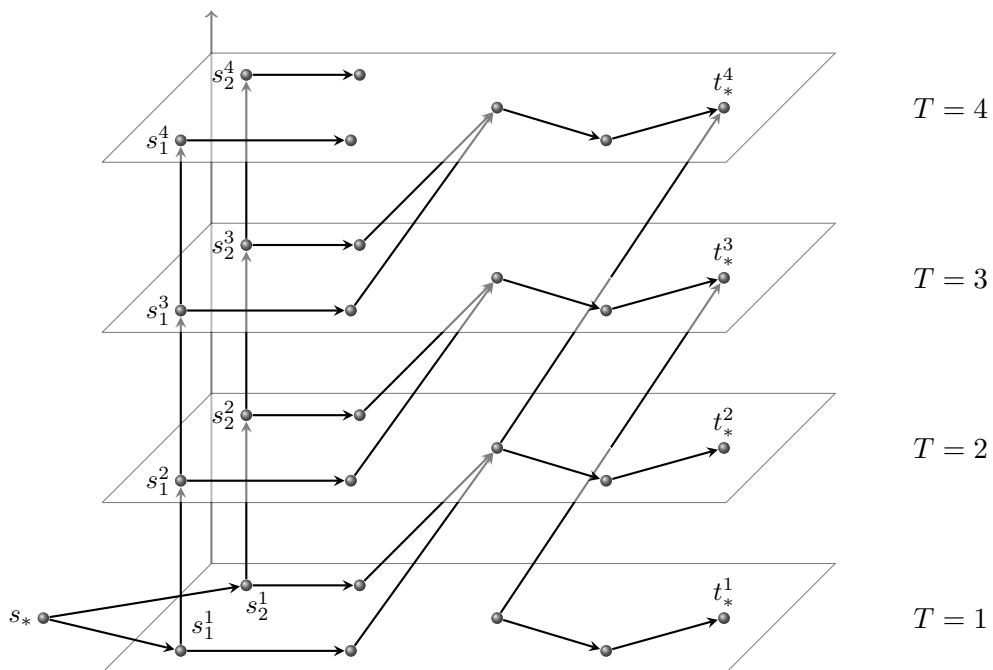
Definition 31. Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk mit $S^- = \{t_*\}$ (also $|S^-| = 1$) und $T < \infty$ ein Zeithorizont. Sei G^T der zeitexpandierte Graph zu G bei Zeithorizont T . Das zeitexpandierte Netzwerk $N^T = ((G^T)', u^T, -, -, (S^+)^T, (S^-)^T)$ zu N definieren wir durch

$$\begin{aligned}
 (G^T)' &:= ((V^T)', (E^T)') \\
 (V^T)' &:= V^T \cup \{s_*\} && (s_* \notin V^T) \\
 (E^T)' &:= E^T \cup \{(s_*, s^1) \mid s \in S^+\} \\
 &\quad \cup \{(s^t, s^{t+1}) \mid s \in S^+, t \in \{1, \dots, T-1\}\} \\
 u^T(e^t) &:= u(e) && \forall e \in E, \forall t \in \{1, \dots, T - \tau(e)\} \\
 u^T(s_*, s^1) &:= p(s) && \forall s \in S^+ \\
 u^T(s^t, s^{t+1}) &:= \infty && \forall s \in S^+, \forall t \in \{1, \dots, T-1\} \\
 (S^+)^T &:= \{s_*\} \\
 (S^-)^T &:= \{t_* \mid \{1 \leq t \leq T\}
 \end{aligned}$$

2 Dynamische Transshipments



(a) Ein gerichtetes Netzwerk N mit Einheitskapazitäten, zwei Quellen s_1 und s_2 sowie einer Senke t_* . Die Fahrzeiten stehen jeweils an den Kanten.



(b) Das zeitexpandierte Netzwerk N^T zum Netzwerk aus (a) für $T = 4$.

Abbildung 2.2: Diese Abbildung illustriert das zeitexpandierte Netzwerk.

Abbildung 2.2 zeigt ein gerichtetes Netzwerk N und sein zeitexpandiertes Netzwerk N^T zum Zeithorizont $T = 4$. Es wird deutlich, dass die Kanten mit Nullfahrzeiten innerhalb der Zeitschichten verlaufen, während die Kanten mit größerer Fahrzeit in höheren Zeitschichten enden.

Das folgende Lemma formuliert den zentralen Nutzen von zeitexpandierten Netzwerken: Sie sind genau so konstruiert, dass ausgleichende Flüsse in N durch statische Flüsse in N^T modelliert werden können. Dies wird im Beweis deutlich, für den wir hauptsächlich die einzelnen Definitionen zusammentragen.

Lemma 32. *Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk und N^T das zeitexpandierte Netzwerk zu N zum Zeithorizont T . Dann induziert jeder ausgleichende Fluss in N einen zulässigen statischen Fluss in N^T , der Flusserhaltung respektiert, und umgekehrt. Der Flusswert des dynamischen Flusses zu einem Zeitpunkt t für $1 \leq t \leq T$ stimmt dabei mit dem Flusswert des statischen Flusses am Knoten t_*^t überein.*

Beweis. Sei f ein ausgleichender Fluss in N . Wir definieren $x_f : E^T \rightarrow \mathbb{R}_{\geq 0}$ durch

$$\begin{aligned} x_f(e_t) &:= f(e, t) && \forall e \in E, \forall t \in \{1, \dots, T - \tau(e)\} \\ x_f(s_*, s^1) &:= |\text{wert}_f(s)| && \forall s \in S^+ \\ x_f(s^t, s^{t+1}) &:= - \sum_{t'=t+1}^T \text{wert}_f(s, t') && \forall s \in S^+, \forall t \in \{1, \dots, T - 1\} \end{aligned}$$

Da die Kapazität jeder Kante $e^t \in E^T$, $e \in E$, $t \in \{1, \dots, T - \tau(e)\}$ der Kapazität von $e \in E$ entspricht und f zulässig ist, werden die Kapazitäten für diese Kanten eingehalten. Außerdem gilt $|\text{wert}_f(s)| \leq |p(v)| = u^T(s_*, s^1)$, da f ausgleichend ist. Die Holdover-Kanten besitzen unendliche Kapazität, dort kann die Kapazitätsbedingung also nicht verletzt werden. Also hält x_f für alle Kanten in E^T die Kapazitätsbedingungen ein. Für die Flusserhaltung stellen wir zunächst fest, dass der Flusswert, den man an Knoten vom Typ $v^t \in V^T$ aus den Kanten vom Typ $e^t \in E^T$ erhält, gerade $\text{wert}_f(v, t)$ entspricht:

$$\begin{aligned} \forall v \in V, \quad \forall t \in \{1, \dots, T\} : & \sum_{e \in \delta^{\leftarrow}(v) \wedge t - \tau(e) \geq 1} x_f(e^{t - \tau(e)}) - \sum_{e \in \delta^{\rightarrow}(v) \wedge t + \tau(e) \leq T} x_f(e^t) \\ &= \sum_{e \in \delta^{\leftarrow}(v) \wedge t - \tau(e) \geq 1} f(e, t - \tau(e)) - \sum_{e \in \delta^{\rightarrow}(v) \wedge t + \tau(e) \leq T} f(e, t) \\ &= \sum_{e \in \delta^{\leftarrow}(v) \wedge t - \tau(e) \geq 1} f(e, t - \tau(e)) - \sum_{e \in \delta^{\rightarrow}(v)} f(e, t) \\ &= \text{wert}_f(v, t) \end{aligned}$$

2 Dynamische Transshipments

Für die Knoten in $V \setminus (S^+)$ ist damit $\text{wert}_{x_f}^{\text{st}}(v^t) = \text{wert}_f(v, t)$, da an Knoten vom Typ v^t mit $v \in V \setminus (S^+)$ nur Kanten vom Typ e^t anliegen. Damit gilt Flussershaltung für alle $v \in V \setminus (S^+ \cup S^-)$:

$$\forall v \in V \setminus (S^+ \cup S^-), \quad \forall t \in \{1, \dots, T\} : \quad \text{wert}_{x_f}^{\text{st}}(v^t) = \text{wert}_f(v, t) = 0$$

Außerdem erfüllt der Flusswert an den Kopien der Senke die Aussage im Lemma:

$$\forall t \in \{1, \dots, T\} : \quad \text{wert}_{x_f}^{\text{st}}(t_*) = \text{wert}_f(t_*, t)$$

Für die Kopien der Quelle müssen wir noch zusätzliche Kanten berücksichtigen. An den jeweils ersten Kopien liegen dabei je eine Kante von der Supersenke und eine Holdover-Kante zur nächsten Kopie an. Wir sehen, dass sich die eingehenden und ausgehenden Flusswerte aufheben:

$$\begin{aligned} \forall s \in (S^+) : \\ \text{wert}_{x_f}^{\text{st}}(s^1) &= \text{wert}_f(s, 1) + x_f(s_*, s^1) - x_f(s^1, s^2) \\ &= \text{wert}_f(s, 1) - \text{wert}_f(s) + \sum_{t'=2}^T \text{wert}_f(s, t') \\ &= \text{wert}_f(s, 1) - \sum_{t'=1}^t \text{wert}_f(s, t') + \sum_{t'=2}^T \text{wert}_f(s, t') \\ &= \text{wert}_f(s, 1) - \text{wert}_f(s, 1) \\ &= 0 \end{aligned}$$

An Kopien in Zwischenzeitschritten liegen nur Holdover-Kanten an. Auch hier addieren sich die Flusswerte zu Null.

$$\begin{aligned} \forall s \in (S^+), \forall t \in \{2, \dots, T-1\} : \\ \text{wert}_{x_f}^{\text{st}}(s^t) &= \text{wert}_f(s, t) + x_f(s^{t-1}, s^t) - x_f(s^t, s^{t+1}) \\ &= \text{wert}_f(s, t) - \sum_{t'=t}^T \text{wert}_f(s, t') + \sum_{t'=t+1}^T \text{wert}_f(s, t') \\ &= \text{wert}_f(s, t) - \text{wert}_f(s, t) \\ &= 0 \end{aligned}$$

Abschließend überprüfen wir noch, dass sich die Flusswerte für die oberste Kopie jeder Quelle aufheben:

$$\begin{aligned} \forall s \in (S^+) : \\ \text{wert}_{x_f}^{\text{st}}(s^T) &= \text{wert}_f(s, T) + x_f(s^{T-1}, s^T) \\ &= \text{wert}_f(s, T) - \sum_{t'=T}^T \text{wert}_f(s, t') \\ &= \text{wert}_f(s, T) - \text{wert}_f(s, T) \\ &= 0 \end{aligned}$$

Damit respektiert x_f Flusserhaltung.

Wenn ein Fluss x in N^T gegeben ist, definieren wir $f_x : E \rightarrow \mathbb{R}_{\geq 0}$ durch

$$\begin{aligned} f_x(e, t) &:= x(e_t) \quad \forall e \in E, \forall t \in \{1, \dots, T - \tau(e)\} \\ f_x(e, t) &:= 0 \quad \forall e \in E, \forall t \in \{T - \tau(e) + 1, \dots, T\} \end{aligned}$$

Damit erfüllt f_x nach Definition bereits die Bedingung, dass nach Ablauf des Zeithorizonts kein Fluss mehr unterwegs sein darf. Außerdem hält f_x die Kapazitätsbedingungen ein, weil x die Kapazitäten einhält. Analog zur Rechnung oben können wir berechnen, dass der Flusswert von f_x zu einem bestimmten Zeitpunkt t direkt mit den Flusswerten übereinstimmt, die man erhält, wenn man in N^T nur die Kanten vom Typ e^t berücksichtigt:

$\forall v \in V, \quad \forall t \in \{1, \dots, T\} :$

$$\text{wert}_{f_x}(v, t) = \sum_{e \in \delta^-(v) \wedge t - \tau(e) \geq 1} x(e^{t - \tau(e)}) - \sum_{e \in \delta^-(v) \wedge t + \tau(e) \leq T} x(e^t) \quad (2.1)$$

Wie oben nutzen wir aus, dass die rechte Seite der Gleichung für Knoten $v \in V \setminus S^+$ gerade $\text{wert}_x(v^t)$ ist, da an Knoten v^t mit $v \in V \setminus S^+$ in N^T nur Kanten anliegen, die Kopien von Kanten in E sind. Damit erfüllt f_x für alle Knoten $v \in V \setminus (S^+ \cup S^-)$ $\text{wert}_{f_x}(v, t) = \text{wert}_x(v^t) = 0$, da x Flusserhaltung erfüllt. Für die Senke erhalten wir die gewünschte Gleichung $\text{wert}_{f_x}(t_*, t) = \text{wert}_x(t_*^t)$. Wir müssen noch zeigen, dass f_x ausgleichend ist. Dafür nutzen wir erneut aus, dass für $\text{wert}_x(s^t)$ zwar nicht dieselbe Gleichung gilt wie für die übrigen Knoten der Form v^t , dass man den Wert aber unter Berücksichtigung der zusätzlichen Kanten auf ähnliche Weise ausdrücken kann:

$\forall s \in S^+ :$

$$\begin{aligned} \text{wert}_x(s^1) &= \sum_{e \in \delta^-(v) \wedge t - \tau(e) \geq 1} x(e^{t - \tau(e)}) - \sum_{e \in \delta^-(v) \wedge t + \tau(e) \leq T} x(e^t) + x(s_*, s^1) - x(s^1, s^2) \\ \text{wert}_x(s^T) &= \sum_{e \in \delta^-(v) \wedge t - \tau(e) \geq 1} x(e^{t - \tau(e)}) - \sum_{e \in \delta^-(v) \wedge t + \tau(e) \leq T} x(e^t) + x(s^{T-1}, s^T) \end{aligned}$$

$\forall s \in S^+, \forall t \in \{2, \dots, T - 1\} :$

$$\text{wert}_x(s^t) = \sum_{e \in \delta^-(v) \wedge t - \tau(e) \geq 1} x(e^{t - \tau(e)}) - \sum_{e \in \delta^-(v) \wedge t + \tau(e) \leq T} x(e^t) + x(s^{t-1}, s^t) - x(s^t, s^{t+1})$$

2 Dynamische Transshipments

Zusammen mit Gleichung (2.1) erhalten wir so eine Gleichung für $\text{wert}_{f_x}(s, t)$, die wir nun verwenden, um $\text{wert}_{f_x}(s)$ zu bestimmen.

$$\begin{aligned}
 \text{wert}_{f_x}(s) &= \sum_{t=1}^T \text{wert}_{f_x}(s, t) \\
 &= \text{wert}_x(s^1) - (x(s_*, s^1) - x(s^1, s^2)) \\
 &\quad + \sum_{t=2}^{T-1} [\text{wert}_x(s^t) - (x(s^{t-1}, s^t) - x(s^t, s^{t+1}))] + \text{wert}_x(s^T) - x(s^{T-1}, s^T) \\
 &= \underbrace{\left[\sum_{t=1}^T \text{wert}_x(s^t) \right]}_{=0} - x(s_*, s^1) \\
 &= -x(s_*, s^1)
 \end{aligned}$$

Damit gilt für alle $s \in S^+$ $|\text{wert}_{f_x}(s)| = x(s_*, s^1) = u(s_*, s^1) \leq |p(s)|$. Für t_* ist $p(t_*) = \sum_{s \in S^+} p(s)$, da p zulässig ist. Außerdem ist aufgrund von Lemma 20 $|\text{wert}_{f_x}(t_*)| = \sum_{s \in S^+} |\text{wert}_{f_x}(s)| \leq \sum_{s \in S^+} |p(s)| = p(t_*)$. Also ist f_x ein ausgleichender Fluss. \square

Als nächstes gehen wir auf die Arbeit von Minięka ein, der in [24] die Existenz von *lexikographisch maximalen Flüssen* zeigt. Der Begriff *lexikographisch maximal* bezieht sich dabei auf den Vektor, den man erhält, wenn man die Flusswerte einiger Quellen oder Senken in einer festen Reihenfolge notiert. Wir legen in dieser Beschreibung fest, dass wir den Vektor zu einer Menge $(S^+)'$ von Senken bilden, da wir nur dies für unseren späteren Beweis benötigen. Minięka geht auf die Beschreibung für Quellen und Senken gleichermaßen ein, Hoppe [18] verwendet auch Vektoren, die Flusswerte von Quellen *und* Senken enthalten.

Die Reihenfolge der Senken bei der Erstellung des Vektors stellen eine Art *Priorität* dar: Für eine feste Reihenfolge werden die vorderen Senken gegenüber späteren Senken bevorzugt, wenn man den Fluss lexikographisch maximiert. Minięka zeigt jedoch, dass es immer einen (statischen) Fluss gibt, der trotz des Maximierens der Summe der Flusswerte der ersten k Quellen auch die Summe der Flusswerte der ersten $k + 1$ Quellen maximiert – und das für alle $k \leq (S^+)' - 1$. Diese Flüsse nennt er *lexikographisch maximal* (bzgl. der Senken).

Dafür verwendet er folgende Definition, die wir hier übernehmen wollen (obwohl sie zu unserer bisherigen Schreibweise von Flusswerten nicht ganz konsistent ist):

Definition 33. Sei $N = (G, u, -, -, S^+, S^-)$ ein gerichtetes Netzwerk. Mit $V(T')$ bezeichnen wir die maximale Anzahl an Flusseinheiten, die in einem statischen Fluss in die Senken in T' fließen kann:

$$V(T') := \max \left\{ \sum_{t' \in T'} \text{wert}_x^{st}(t') \mid \begin{array}{l} f \text{ ist ein zulässiger, statischer Fluss in } N, \\ \text{der Flusserhaltung respektiert.} \end{array} \right\}$$

Der entscheidende Satz sagt nun aus, dass es immer einen statischen Fluss gibt, der $V(T')$ für eine Folge von ineinander enthaltenen Teilmengen der Senkenmenge gleichzeitig realisiert.

Satz 34 (Minieka, 1973). Sei $N = (G, u, -, -, S^+, S^-)$ ein gerichtetes Netzwerk und sei eine Folge $S_1^- \subseteq S_2^- \subseteq \dots \subseteq S_n^- \subseteq S^-$ von Teilmengen von S^- gegeben. Dann gibt es einen zulässigen, statischen Fluss in N , der Flusserhaltung respektiert und

$$\text{wert}_x^{st}(S_i^-) = V(S_i^-) \quad \forall i = 1, \dots, n$$

erfüllt.

Beweis. [24, Theorem 2] □

Minieka verwendet diesen Satz selbst dazu, um (unter anderem) die Existenz von Earliest Arrival Flüssen für Netzwerke mit einer Quelle und einer Senke zu beweisen. In [2] wird erwähnt, dass sich seine Konstruktion auch auf Netzwerke mit mehreren Quellen und einer Senke erweitern lässt. In der Tat stellen wir nun fest, dass unser zeitexpandiertes Netzwerk genau so definiert ist, dass die Existenz von Earliest Arrival Flüssen einfach zu sehen ist:

Korollar 35. Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk mit $S^- = \{t_*\}$. Dann gibt es ein Earliest Arrival Transshipment in N .

Beweis. Sei N^T das zeitexpandierte Netzwerk zu N wie in Definition 31. Wir möchten zeigen, dass es in N einen ausgleichenden Fluss gibt, der für jeden Zeitpunkt $t \in \{1, \dots, T\}$ den maximalen Flusswert $w_{\max}^+(t)$ erreicht. Aus Lemma 32 wissen wir, dass es eine Bijektion zwischen ausgleichenden Flüssen in N und zulässigen, statischen Flüssen in N^T , die Flusserhaltung respektieren, gibt, und dass dabei der Flusswert des dynamischen Flusses für jedes $t \in \{1, \dots, T\}$ mit dem Flusswert des statischen Flusses am Knoten t_*^t übereinstimmt. Daher reicht es aus, einen zulässigen statischen Fluss in N^T zu finden, der Flusserhaltung respektiert und die Summe der Flusswerte an $t_*^1, t_*^2, \dots, t_*^t$ für jedes t

2 Dynamische Transshipments

simultan maximiert, um zu zeigen, dass es einen ausgleichenden Fluss in N gibt, der zu jedem Zeitpunkt den Flusswert $w_{\max}^+(t)$ erreicht.

Jetzt definieren wir S_i^- für $i = 1, \dots, T$ durch

$$S_i^- := \{t_*^1, \dots, t_*^i\} \subseteq S^-.$$

Dadurch ist $S_i^- \subseteq S_{i+1}^- \forall i = 1, \dots, T-1$ erfüllt. Laut Satz 34 gibt es einen zulässigen statischen Fluss x , der Flusserhaltung respektiert und $\text{wert}_x^{\text{st}}(S_i^-) = V(S_i^-) \forall i = 1, \dots, n$ erfüllt. Außerdem ist $V(S_i^-)$ gerade der maximale Wert, den $\sum_{t=1}^i t_*^t$ erreichen kann. Damit entspricht x einem ausgleichenden Fluss f_x in N , der die Earliest Arrival Eigenschaft erfüllt. \square

Es wird auch deutlich, dass sich dieser Beweis auf Netzwerke mit mehreren Senken *nicht* übertragen lässt. In diesem Fall müsste im zeitexpandierten Netzwerk zusätzlich sichergestellt werden, dass der gesamte Fluss, der in eine Senke fließt, deren Bedarf nicht übersteigt. Das ist in der obigen Konstruktion nicht notwendig, da es nur eine Senke gibt – da p zulässig ist und die Angebote der Quellen eingehalten werden, kann der Bedarf von t von einem zulässigen Fluss, der Flusserhaltung respektiert, nicht überschritten werden. Wenn man dies selbst sicherstellen will, muss man die Flüsse in die gleiche Senke über alle Zeitschritte hinweg sammeln – dies widerspricht aber der Idee, jede Kopie einer Senke als eigene Senke aufzufassen (und den dort eintreffenden Fluss auch dort verschwinden zu lassen) und auf diese Weise Satz 34 anwenden zu können.

2.3 Algorithmen zur Berechnung dynamischer Transshipments (Überblick)

Minieka [24] beweist die Existenz von lexikographisch maximalen Flüssen konstruktiv, indem er den Algorithmus von Ford und Fulkerson zur Berechnung maximaler (statischer) Netzwerkflüsse geeignet abwandelt. Dadurch kann man Earliest Arrival Transshipments berechnen, indem man das zeitexpandierte Netzwerk wie oben aufbaut und darin lexikographisch maximale Flüsse sucht. Unabhängig von der Laufzeit des Ford-Fulkerson-Algorithmus ist die Laufzeit bei einem solchen Vorgehen niemals polynomiell, da das zeitexpandierte Netzwerk $\Theta(T \cdot |V|)$ Knoten sowie $\Theta(T \cdot |E|)$ Kanten besitzt und damit nur pseudopolynomielle Größe hat. Außerdem (und für praktische Zwecke noch ungünstiger) ist auch der Speicherbedarf exponentiell.

Im Folgenden ziehen wir uns vorübergehend auf Netzwerke mit einer einzigen Quelle s und einer einzigen Senke t zurück. Für diesen Fall bietet der Shortest-Path-Algorithmus eine

2.3 Algorithmen zur Berechnung dynamischer Transshipments (Überblick)

praktisch wesentlich geeignetere, aber (in der Laufzeit) immer noch pseudopolynomielle Methode. Dieser geht ebenfalls auf Minieka [24] sowie (unabhängig davon) auf Wilkinson [29] zurück. Beim Shortest-Path-Algorithmus wird ausgenutzt, dass die Klasse der *verallgemeinerten zeitlich wiederholten Flüsse* immer einen Earliest Arrival Fluss enthält. Einen *zeitlich wiederholten Fluss* erhält man auf folgende Weise: Zuerst wählt man eine Menge von Pfaden \mathcal{P} von s nach t . Jedem dieser Pfade $p \in \mathcal{P}$ weist man einen Flusswert x_p zu, so dass der statische Fluss, den man auf diese Weise induziert, zulässig ist (Flusserhaltung respektiert er bereits aufgrund der Konstruktion). Aus dem statischen Fluss gewinnt man nun einen dynamischen *zeitlich wiederholten Fluss*, indem man in jeden Pfad p in jedem Zeitschritt von $t = 1$ bis $t = (T - \tau(p))$ x_p Flusseinheiten hineinschickt, dabei bezeichnen wir mit $\tau(p)$ die Summe der Fahrzeiten der Kanten von p . Da wir für jeden Pfad rechtzeitig aufhören, Fluss zu schicken, ist nach $t = T$ kein Fluss mehr im Netzwerk. Außerdem ist der so erhaltene dynamische Fluss zulässig, weil zu keinem Zeitpunkt auf irgendeiner Kante insgesamt mehr Fluss verschickt wird als im statischen Fluss, der zulässig ist. Flusserhaltung ist zudem erfüllt, weil Fluss immer entlang von Pfaden unterwegs ist. Nicht jeder dynamische Fluss ist ein zeitlich wiederholter Fluss, Ford und Fulkerson haben aber bereits in [11] gezeigt, dass es immer einen maximalen Fluss gibt, der ein zeitlich wiederholter Fluss ist.

Für Earliest Arrival Flüsse gilt dies im Allgemeinen nicht, hier benötigen wir *verallgemeinerte zeitlich wiederholte Flüsse*. Diese unterscheiden sich von zeitlich wiederholten Flüssen dadurch, dass man in der Menge von Pfaden \mathcal{P} auch Pfade zulässt, die Kanten *rückwärts* benutzen. Ein Pfad, der eine Kante rückwärts verwendet, nimmt damit Fluss auf der Kante zurück. Damit dieses Vorgehen sinnvoll ist, muss sichergestellt werden, dass nur Kanten rückwärts verwendet werden, denen von einem anderen Pfad mindestens genauso viel Fluss zugewiesen wird wie wir zurücknehmen möchten – und zwar genau in den Zeitschritten, in denen wir die Kante rückwärts benutzen. Der Successive-Shortest-Path-Algorithmus berechnet iterativ Pfade so, dass diese Bedingung eingehalten wird und der erzeugte verallgemeinerte zeitlich wiederholte Fluss die Earliest Arrival Eigenschaft erfüllt⁴.

⁴Der Beweis dieser beiden Tatsachen ist in der Tat schwieriger als die Beschreibung des Algorithmus selbst.

Die Berechnung der Pfade besteht daraus, kürzeste Wege im sog. *Residualnetzwerk* zu suchen – diese enthalten Kanten nur dann rückwärts, wenn es andere Pfade gibt, die die Kante vorwärts benutzen. Es ist aber nicht direkt klar, warum sich dieses auf die zeitliche Wiederholung überträgt. Bei der Implementierung des Successive-Shortest-Path-Algorithmus ist der komplizierteste Teil das Aufbauen des dynamischen Flusses aus den vorgegebenen Pfaden und Flusswerten, da man die Rückwärtskanten passend verarbeiten muss.

2 Dynamische Transshipments

Leider ist die Anzahl der Pfadberechnungen, die der Successive-Shortest-Path-Algorithmus durchführt, nur pseudopolynomiell, und es gibt eine Klasse von Netzwerken (siehe [30]), für die die Anzahl der Iterationen tatsächlich exponentiell ist.

Bisher haben wir uns immer im diskreten Zeitmodell bewegt. Im kontinuierlichen Zeitmodell dürfen Flüsse den Flusswert auf jeder Kante zu jedem kontinuierlichen Zeitpunkt ändern, der Fluss auf einer speziellen Kante wird dann als Lebesgue-integrierbare Funktion modelliert. Außerdem ist die Earliest Arrival Eigenschaft nur erfüllt, wenn die Flussmenge eines Flusses für jeden kontinuierlichen Zeitpunkt maximal ist. Trotzdem gilt die Existenz von Earliest Arrival Flüssen bei einer Senke weiterhin (siehe [25]). Eine kontinuierliche Variante des Successive-Shortest-Path-Algorithmus wird von Fleischer und Tardos in [9] vorgestellt. Hoppe und Tardos [19] beschreiben einen Algorithmus, der Earliest Arrival Flüsse in Netzwerken mit einer Quelle und einer Senke auf folgende Weise approximiert: Für ein gegebenes $\epsilon > 0$ wird ein Fluss berechnet, dessen Flusswert für jeden (kontinuierlichen) Zeitpunkt t mindestens ein $(1 - \epsilon)$ -faches des maximalen Wertes ist, den ein (kontinuierlicher) dynamischer Fluss zum Zeitpunkt t erreichen kann⁵.

Wir kehren jetzt zu dem allgemeineren Fall von Netzwerken mit mehreren Quellen und einer Senke zurück. Earliest Arrival Transshipments existieren auch hier nicht nur im diskreten, sondern auch im kontinuierlichen Zeitmodell. Baumann und Skutella [2] geben hierfür einen Beweis, der nicht auf den diskreten Fall zurückgreift und im Gegensatz zu dem oben zitierten Beweis nicht auf Zeitexpansion beruht. Es handelt sich um einen konstruktiven Beweis, der einen Algorithmus für die Berechnung von Earliest Arrival Transshipments liefert, dessen Laufzeit (stark) polynomiell in der Summe von Eingabe- und Ausgabegröße ist. Dieser beruht u.a. auf einem Algorithmus von Hoppe [18] zur Berechnung (diskreter) dynamischer Transshipments in Netzwerken mit mehreren Quellen und Senken⁶. Bereits diese sehr viel eingeschränkte Fragestellung stellt (im Gegensatz zum statischen Fall) ein zwar polynomiell lösbares, aber dennoch sehr kniffliges Problem dar. Der Algorithmus in [18] ist der erste (stark) polynomielle Algorithmus zur Berechnung dynamischer Transshipments, und er verwendet die Minimierung submodularer Funktionen, so dass er für praktische Zwecke nicht relevant ist. Für die Berechnung eines Quickest Transshipments in (stark) polynomieller Zeit bettet Hoppe seinen Algorithmus in das Framework von Megiddo zur parametrischen Suche [22, 23] ein.

Fleischer und Skutella geben in [8] einen Algorithmus an, der Earliest Arrival Transshipment für mehrere Quellen und eine Senke auf andere Weise approximiert als der Algorithmus

⁵Die Laufzeit des Algorithmus ist polynomiell in der Eingabegröße *und* in $1/\epsilon$, es handelt sich also um einen sog. FPTAS (Fully Polynomiell Approximation Scheme).

⁶Der Algorithmus kann in das kontinuierliche Zeitmodell übertragen werden, siehe [9].

mus von Hoppe und Tardos⁷: Für ein gegebenes $\epsilon > 0$ wird ein Fluss berechnet, dessen Flusswert für jeden (kontinuierlichen) Zeitpunkt t mindestens dem maximalen Flusswert zum Zeitpunkt $(1 + \epsilon) \cdot t$ entspricht. Ein erwähnenswerter (pseudopolynomieller) Algorithmus wird von Tjandra in [28] beschrieben. Dieser beschäftigt sich mit einer größeren Klasse von dynamischen Flüssen, in der sich die Kapazitäten und Fahrzeiten auf den einzelnen Kanten mit der Zeit ändern können. Solche zeitabhängigen Kapazitäten und Fahrzeiten sind natürlich noch besser in der Lage, reale Abläufe zu modellieren. Interessant ist der Algorithmus aber auch deshalb, weil er ein implizites zeitexpandiertes Netzwerk verwendet und dadurch praktisch relevant ist. In einer abgewandelten Form für konstante Kapazitäten und Fahrzeiten wurde der Algorithmus innerhalb der bereits in Abbildung 1.1 erwähnten Evakuierungssoftware ZET implementiert [14].

2.4 Ein LP-basierter Existenztest

Wir entwickeln hier einen einfachen (pseudopolynomiellen) Existenztest für Earliest Arrival Transshipments, der auf der Lösung eines *linearen Programms* beruht. Lineare Programme können auf verschiedene Weisen definiert werden, wir entscheiden uns für eine relativ umfangreiche Version, die wir direkt gut auf unser Problem anwenden können.

Definition 36. Ein lineares Programm P besteht aus zwei Matrizen $A = (a)_{ij} \in \mathbb{R}^{k' \times l}$ und $B = (b)_{ij} \in \mathbb{R}^{k'' \times l}$ sowie drei Vektoren $c \in \mathbb{R}^{k'}$, $d \in \mathbb{R}^{k''}$ und $e \in \mathbb{R}^l$. Eine zulässige Lösung eines linearen Programms P ist ein Vektor $x \in \mathbb{R}_{\geq 0}^l$, der die Ungleichungen $a_{i1}x_1 + a_{i2}x_2 + \dots + a_{il}x_l \leq c_i \forall i = 1, \dots, k'$ und die Gleichungen $b_{i1}x_1 + b_{i2}x_2 + \dots + b_{il}x_l = d_i \forall i = 1, \dots, k''$ erfüllt. Eine optimale Lösung eines linearen Programms P ist eine zulässige Lösung, für die der Wert $e_1x_1 + e_2x_2 + \dots + e_lx_l$ maximal ist, d.h. für jede zulässige Lösung x' von P gilt $e_1x'_1 + e_2x'_2 + \dots + e_lx'_l \leq e_1x_1 + e_2x_2 + \dots + e_lx_l$. Ein lineares Programm P schreiben wir abgekürzt auch als

$$\begin{aligned}
 (P) \quad & \text{Maximiere} \quad e^T x \\
 & \text{s.t.} \quad Ax \leq c \\
 & \quad \quad Bx = d \\
 & \quad \quad x \geq 0
 \end{aligned}$$

Das Problem, eine zulässige Lösung für ein lineares Programm zu finden, ist polynomiell lösbar. Dieses wurde zuerst von Hačijan [16] gezeigt, der die *Ellipsoidmethode* auf die Lösung linearer Programme übertragen hat. Zur praktischen Lösung werden i.d.R. Varianten

⁷Es handelt sich auch hier um einen FPTAS.

2 Dynamische Transshipments

der von Dantzig [3] vorgestellten *Simplex-Methode* verwendet, die zwar im schlechtesten Fall exponentielle Laufzeit hat, aber in vielen Fällen praktisch effizient ist. Wir benötigen im Folgenden nur einen Algorithmus, der eine zulässige Lösung zu einem linearen Programm findet. Dieses Problem ist zum Finden einer optimalen Lösung äquivalent [15].

Um unser Problem als lineares Programm zu formulieren, möchten wir es zuerst auf ein statisches Flussproblem zurückführen, da sich statische Flussprobleme gut als lineares Programm formulieren lassen. Dafür verwenden wir ein zeitexpandiertes Netzwerk ähnlich zu Definition 31, müssen es jedoch, da wir potentiell mehr als eine Senke haben, um Holdover-Kanten für die Senken sowie eine Supersenke t_* erweitern.

Definition und Lemma 37. Sei $N = (G, u, \tau^0, p, S^+, S^-)$ ein gerichtetes Netzwerk mit $S^- = \{t_*\}$ (also $|S^-| = 1$) und $T < \infty$ ein Zeithorizont. Sei $G^T = (V^T, E^T)$ der zeitexpandierte Graph zu G bei Zeithorizont T (siehe Definition 30). Wir definieren das zeitexpandierte Netzwerk $\tilde{N}^T = (\tilde{G}^T, \tilde{u}^T, -, -, (\tilde{S}^+)^T, (\tilde{S}^-)^T)$ zu N für einen Zeithorizont T durch

$$\begin{aligned}
 \tilde{G}^T &:= (\tilde{V}^T, \tilde{E}^T) \\
 \tilde{V}^T &:= V^T \cup \{s_*, t_*\} && (s_*, t_* \notin V^T) \\
 \tilde{E}^T &:= E^T \\
 &\cup \{(s_*, s^1) \mid s \in S^+\} \cup \{(z^1, t_*) \mid z \in S^-\} \\
 &\cup \{(s^t, s^{t+1}) \mid s \in S^+, t \in \{1, \dots, T-1\}\} \\
 &\cup \{(z^t, z^{t+1}) \mid z \in S^-, t \in \{1, \dots, T-1\}\} \\
 \tilde{u}^T(e^t) &:= u(e) && \forall e \in E, \forall t \in \{1, \dots, T - \tau(e)\} \\
 \tilde{u}^T(s_*, s^1) &:= p(s) && \forall s \in S^+ \\
 \tilde{u}^T(z^T, t_*) &:= -p(z) && \forall z \in S^- \\
 \tilde{u}^T(s^t, s^{t+1}) &:= \infty && \forall s \in S^+, \forall t \in \{1, \dots, T-1\} \\
 \tilde{u}^T(z^t, z^{t+1}) &:= \infty && \forall z \in S^-, \forall t \in \{1, \dots, T-1\} \\
 (\tilde{S}^+)^T &:= \{s_*\} \\
 (\tilde{S}^-)^T &:= \{t_*\}
 \end{aligned}$$

Dann induziert jeder ausgleichende Fluss in N einen zulässigen statischen Fluss in \tilde{N}^T , der Flusserkhaltung respektiert, und umgekehrt. Der Flusswert des dynamischen Flusses zu einem Zeitpunkt t für $2 \leq t \leq T-1$ an einer Senke $z \in S^-$ entspricht dabei der Differenz zwischen dem Flusswert des statischen Flusses auf der ausgehenden Holdover-Kante (z^t, z^{t+1}) und dem Flusswert des statischen Flusses auf der eingehenden Holdover-Kante (z^{t-1}, z^t) . Für $t = 1$ ist der Flusswert des dynamischen Flusses gleich dem Flusswert des

statischen Flusses auf der ausgehenden Holdover-Kante (z^1, z^2) , und für $t = T$ entspricht der dynamische Flusswert an z der Differenz des statischen Flusswertes auf der Kante zur Supersenke (z^T, t_*) und dem Flusswert des statischen Flusses auf der eingehenden Holdover-Kante (z^{T-1}, z^T) .

Beweis. Der Beweis verläuft analog zum Beweis von Lemma 32 unter Einbeziehung der zusätzlichen Holdover-Kanten an den Senken. Sei $f : E \times \{1, \dots, T\} \rightarrow \mathbb{R}_{\geq 0}$ ein ausgleichender Fluss in \tilde{N}^T . Wir definieren $x_f : E^T \rightarrow \mathbb{R}_{\geq 0}$ durch

$$\begin{aligned} x_f(e_t) &:= f(e, t) && \forall e \in E, \forall t \in \{1, \dots, T - \tau(e)\} \\ x_f(s_*, s^1) &:= |\text{wert}_f(s)| && \forall s \in S^+ \\ x_f(z^T, t_*) &:= |\text{wert}_f(z)| && \forall z \in S^- \\ x_f(s^t, s^{t+1}) &:= - \sum_{t'=t+1}^T \text{wert}_f(s, t') && \forall s \in S^+, \forall t \in \{1, \dots, T - 1\} \\ x_f(z^t, z^{t+1}) &:= \sum_{t'=1}^t \text{wert}_f(z, t') && \forall z \in S^-, \forall t \in \{1, \dots, T - 1\} \end{aligned}$$

Analog zu Lemma 32 zeigen wir, dass Kapazitäten und Flusserhaltung eingehalten werden (d.h. $x_f(e) \leq \tilde{u}(e) \forall e \in \tilde{E}$ sowie $\text{wert}_{x_f}(v) = 0 \forall v \in V \setminus \{s_*, t_*\}$). Dabei können wir die Rechnungen auf Seite 26 abwandeln, um $\text{wert}_{x_f}^{\text{st}}(z, t) = \text{wert}_f(z^t) + x_f(z^{t-1}, z^t) - x_f(z^t, z^{t+1})$ für alle $z \in S^-, \forall t \in \{2, \dots, T - 1\}$ zu zeigen, was aufgrund der Flusserhaltung (von x_f) an z^t äquivalent zu $\text{wert}_f(z^t) = x_f(z^t, z^{t+1}) - x_f(z^{t-1}, z^t)$ ist. Ebenso erhalten wir $\text{wert}_f(z^1) = x_f(z^1, z^2)$ und $\text{wert}_f(z^T) = x_f(z^T, t_*) - x_f(z^{T-1}, z^T)$.

Zu einem Fluss $x : E^T \rightarrow \mathbb{R}_{\geq 0}$ in \tilde{N} definieren wir $f_x : E \times \{1, \dots, T\} \rightarrow \mathbb{R}_{\geq 0}$ durch

$$\begin{aligned} f_x(e, t) &:= x(e_t) && \forall e \in E, \forall t \in \{1, \dots, T - \tau(e)\} \\ f_x(e, t) &:= 0 && \forall e \in E, \forall t \in \{T - \tau(e) + 1, \dots, T\} \end{aligned}$$

Analog zu Lemma 32 kann man zeigen, dass f_x Kapazitäten und Flusserhaltung erfüllt (d.h. $f_x(e, t) \leq u(e) \forall e \in E, \forall t \in \{1, \dots, T\}$ sowie $\text{wert}_{f_x}(v) = 0 \forall v \in V \setminus \{s_*, t_*\}$ und $\text{wert}_{f_x}(s_*) < 0$ sowie $\text{wert}_{f_x}(t_*) > 0$) und ausgleichend ist (d.h. $|\text{wert}_{f_x}(v)| \leq |p(v)| \forall v \in S^+ \cup S^-$). Außerdem kann man aus den abgewandelten Rechnungen folgern, dass $\text{wert}_{f_x}(z^t) = \text{wert}_x^{\text{st}}(z, t) - x_f(z^{t-1}, z^t) + x_f(z^t, z^{t+1}) = x_f(z^t, z^{t+1}) - x_f(z^{t-1}, z^t)$ für alle $z \in S^-, \forall t \in \{2, \dots, T - 1\}$ sowie $\text{wert}_{f_x}(z^1) = \text{wert}_x^{\text{st}}(z, t) + x_f(z^1, z^2) = x_f(z^1, z^2)$ und $\text{wert}_{f_x}(z^T) = \text{wert}_x^{\text{st}}(z, t) - x_f(z^{T-1}, z^T) + x_f(z^T, t_*) = x_f(z^T, t_*) - x_f(z^{T-1}, z^T)$ gilt. \square

Das zeitexpandierte Netzwerk \tilde{N}^T hat die gleiche asymptotische Größe wie das in Definition 31 definierte Netzwerk N^T , insbesondere ist es ebenfalls exponentiell groß, wenn T

2 Dynamische Transshipments

exponentiell groß ist. Wir können daher nicht auf einen polynomiellen Existenztest hoffen, wenn wir das zeitexpandierte Netzwerk vollständig verarbeiten. Aufgrund des großen Platzbedarfs ist der Test, den wir uns hier überlegen, auch nicht unbedingt praxisrelevant. Das ist auch nicht das Ziel dieses Abschnitts, vielmehr geht es darum, grundsätzlich zu überlegen, wie man die Existenz von Earliest Arrival Transshipments überprüfen kann. Dafür benötigen wir nun, dass die Werte $w_{\max}^+(t)$ für alle $t \geq 1$ bekannt sind. Da w_{\max}^+ maximal den (endlichen) Wert $p(S^-) := \sum_{t \in S^-} p(t)$ erreichen kann, gibt es einen minimalen Zeitpunkt T^* , zu dem sich w_{\max}^+ nicht mehr ändert. Ohne uns weiter Gedanken über eine sinnvolle Berechnung von w_{\max}^+ zu machen, können wir davon ausgehen, dass wir sowohl T^* sowie alle Werte $w_{\max}^+(t)$ für $t \in \{1, \dots, T^*\}$ ermitteln können und dieses in einer Laufzeit möglich ist, die polynomiell in der Größe von N und T^* ist⁸.

Jetzt gehen wir folgendermaßen vor: Wir stellen ein lineares Programm auf, dessen zulässige Lösungen gerade statische Flüsse in \tilde{N} darstellen, die in jedem Zeitschritt den Flusswert $w_{\max}^+(t)$ erreichen. Wie wir den Flusswert in \tilde{N} für jeden Zeitschritt ablesen, haben wir in Lemma 37 gesehen.

Satz 38. *Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk und sei $T^* < \infty$ der kleinste Zeithorizont, für den $w_{\max}^+(T^*) = w_{\max}^+(T') \forall T' \geq T^*$ gilt. Sei \tilde{N}^{T^*} das zeitexpandierte Netzwerk zu N bei Zeithorizont T^* . Dann gibt es in N genau dann ein Earliest Arrival Transshipment, wenn das folgende lineare Programm eine zulässige Lösung $x \in \mathbb{R}_{\geq 0}^m$ besitzt:*

$$\begin{aligned}
 (P) \quad & \text{maximiere} && 0 \\
 & \text{s.t.} && x(e) \leq u(e) && \forall e \in \tilde{E}^{T^*} \\
 & && \sum_{e \in \delta^-(v)} x(e) - \sum_{e \in \delta^+(v)} x(e) = 0 && \forall v \in \tilde{V} \setminus \{s_*, t_*\} \\
 & && \sum_{z \in S^-} x(z^t, z^{t+1}) = w_{\max}^+(t) && \forall t = 1, \dots, T^* - 1 \\
 & && \sum_{z \in S^-} x(z^T, t_*) = w_{\max}^+(T^*) \\
 & && x(e) \geq 0 && \forall e \in \tilde{E}^{T^*}
 \end{aligned}$$

Aus jeder zulässigen Lösung von (P) lässt sich ein Earliest Arrival Transshipment in N berechnen.

⁸Da die Berechnung eines maximalen dynamischen Flusses in polynomieller Zeit möglich ist, können wir schlimmstenfalls die Werte von $w_{\max}^+(t)$ nacheinander für aufsteigendes t berechnen. Wenn sich w_{\max}^+ „lange genug“ (z.B. $\sum_{e \in E} \tau(e)$ Zeitschritte lang) nicht geändert hat, haben wir T^* bereits überschritten und können aufhören.

Beweis. Jede zulässige Lösung x^* von (P) stellt einen statischen Fluss x in \tilde{N} dar, der zulässig ist und Flusserhaltung erfüllt. Zulässigkeit gilt, da $x(e) \leq u(e)$ für alle $e \in \tilde{E}$ erfüllt ist. Flusserhaltung ist erfüllt, da $\text{wert}_x(v) = \sum_{e \in \delta^-(v)} x(e) - \sum_{e \in \delta^+(v)} x(e) = 0$ für alle $v \in \tilde{V} \setminus \{s_*, t_*\}$ gilt ($\text{wert}_x(s_*) \leq 0$ und $\text{wert}_x(t_*) \geq 0$ müssen wir nicht überprüfen, da s_* keine eingehenden und t_* keine ausgehenden Kanten besitzt). Aus x^* können wir wie in Lemma 37 einen dynamischen Fluss f_{x^*} bilden, der innerhalb des ersten Zeitschritts an Senke $z \in S^-$ den Flusswert $\text{wert}_{f_{x^*}}(z, 1) \stackrel{\text{Lem. 37}}{=} x^*(z^1, z^2)$, innerhalb der ersten $2 \leq t \leq T^* - 1$ Zeitschritte den Flusswert $\sum_{t'=1}^t \text{wert}_{f_{x^*}}(z, t') \stackrel{\text{Lem. 37}}{=} \sum_{t'=1}^t x^*(z^{t'}, z^{t'+1}) - x^*(z^{t'-1}, z^{t'}) = x^*(z^t, z^{t+1})$ und innerhalb von T^* Zeitschritten den Flusswert $\sum_{t'=1}^t \text{wert}_{f_{x^*}}(z, t') = x^*(z^t, z^{t+1}) + \text{wert}_{f_{x^*}}(T^*) \stackrel{\text{Lem. 37}}{=} x^*(z^t, z^{t+1}) + x^*(z^{T^*}, t_*) - x^*(z^{T^*-1}, z^{T^*}) = x^*(z^t, t_*)$ erreicht. Da x^* zulässige Lösung von (P) ist, erreicht f_{x^*} damit im ersten Zeitschritt den Flusswert $\sum_{z \in S^-} x^*(z^1, z^2) = w_{\max}^+(1)$, in den ersten $2 \leq t \leq T^* - 1$ Zeitschritten den Flusswert $\sum_{z \in S^-} x^*(z^t, z^{t+1}) = w_{\max}^+(t)$ und in T^* Zeitschritten den Flusswert $\sum_{z \in S^-} x^*(z^{T^*}, t_*) = w_{\max}^+(T^*)$.

Wenn es ein Earliest Arrival Transshipment f in N gibt, können wir einen statischen, zulässigen Fluss x_f , der Flusserhaltung erfüllt, wie in Lemma 37 definieren und auf ähnliche Weise ausrechnen, dass x_f eine zulässige Lösung von (P) ist. \square

2 *Dynamische Transshipments*

3 Dynamische Transshipments mit Nullfahrzeiten

3.1 Einleitung

Wir wollen jetzt untersuchen, warum und in welchen Fällen Netzwerke mit Nullfahrzeiten kein Earliest Arrival Transshipment besitzen. Dabei wissen wir bereits, dass die Existenz eines Earliest Arrival Transshipments gesichert ist, wenn es nur eine Senke gibt, weil es in diesem Fall sogar für beliebige Fahrzeiten gilt. Außerdem gibt es immer ein Earliest Arrival Transshipment, wenn nur eine Quelle vorhanden ist. Dieses folgt sofort aus Lemma 43, das wir auf Seite 48 formulieren werden. Dass man für ein Netzwerk mit Nullfahrzeiten, das kein Earliest Arrival Transshipment besitzt, mindestens zwei Quellen und Senken benötigt, kann man sich aber auch intuitiv gut erklären: In Netzwerken mit Nullfahrzeiten gibt es zwischen verschiedenen Zeitschritten keine Abhängigkeiten aufgrund von Fahrzeiten, sondern nur aufgrund von Angeboten und Bedarfen. Angebote und Bedarfe machen erst bei mindestens drei Terminalen Sinn. Nehmen wir an, wir haben mindestens zwei Quellen. Durch die Angebote der Quellen kann nun eine Abhängigkeit entstehen: Wenn beide in die gleiche Senke Fluss schicken können, dann kann eine der Quellen dadurch behindert werden, dass die andere Quelle den Bedarf der Senke bereits teilweise oder ganz erfüllt hat. Das kann aber nur geschehen, wenn es mindestens zwei Senken gibt, da bei einer Senke der Bedarf dieser immer groß genug ist, um das Angebot aller Quellen aufzunehmen. Deshalb ist es bei mehreren Quellen und Senken (und nur dann) wichtig, welche Quelle Fluss zu welcher Senke schickt. Ein Beispiel dafür ist in Abbildung 3.1 zu sehen. In dem dort abgebildeten Netzwerk mit zwei Quellen und zwei Senken gibt es zwei ausgleichende Flüsse, die zu verschiedenen Zeitpunkten den Flusswert maximieren: Der obere Fluss versendet im ersten Zeitschritt einen maximalen Fluss, kann dafür aber in den ersten beiden Zeitschritten zusammen nur fünf Flusseinheiten verschicken. Der unten abgebildete Fluss versendet im ersten Zeitschritt nur drei Flusseinheiten, schafft es aber dafür, in zwei Zeitschritten alle sechs Flusseinheiten zu verschicken. Lisa Fleischer schreibt dazu in [7]:

3 Dynamische Transshipments mit Nullfahrzeiten

„The problem with multiple sinks is that, in the rush to send flow, some source may send flow into the wrong sink. This is not a problem when there is only one sink.“

Es kann also vorkommen, dass ein maximaler Flusswert in einem Zeitschritt nur erreicht werden kann, wenn einige Quellen dabei Fluss in die „falschen“ Senken schicken. In solchen Fällen gibt es kein Earliest Arrival Transshipment.

Das Beispiel aus Abbildung 3.1 lässt sich auf andere Kapazitäten verallgemeinern: In Abbildung 3.2 sehen wir, wie man die Potentiale im Netzwerk abhängig von den Kapazitäten der Kanten wählen kann, um eine Situation zu erzeugen, in der es kein Earliest Arrival Transshipment gibt. Dabei sehen wir in Teilabbildung (a) das Netzwerk und in Teilabbildung (b) die von uns gewählten Angebote und Bedarfe. Anschließend zeigen die Teilabbildungen (c) und (d), wie man für die gegebenen Potentiale den zweiten Zeitschritt maximiert, die Teilabbildungen (e) und (f) zeigen, wie man im ersten Zeitschritt einen maximalen Flusswert erreicht, dabei aber keine Optimalität im zweiten Zeitschritt erreichen kann. Wir werden später auf dieses verallgemeinerte Beispiel zurückkommen. Ein weiteres Beispiel, dass wir später noch benötigen, ist in Abbildung 3.3 abgebildet. In Teilabbildung (a) wird das Netzwerk abgebildet und erklärt, welche Bedingungen die Kapazitäten erfüllen müssen. Wir können nämlich nicht für beliebige Kapazitäten eine Situation erzeugen, in der es ein Earliest Arrival Transshipment gibt: Die mittlere Kante muss eine niedrigere Kapazität haben als die beiden äußeren, damit uns das gelingt.

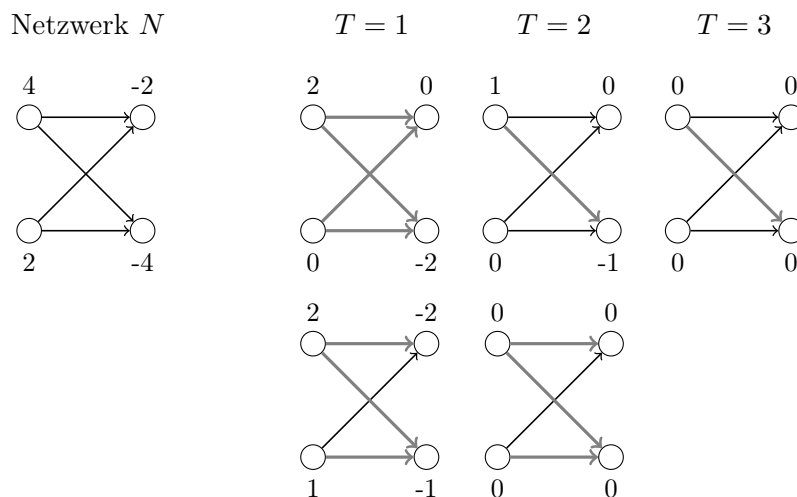
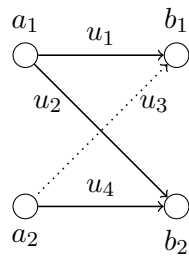
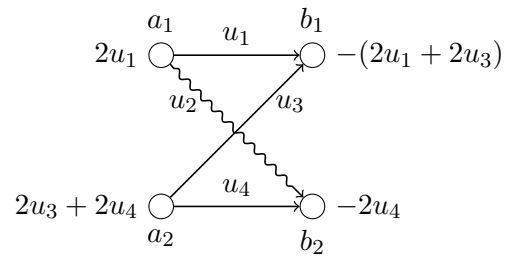


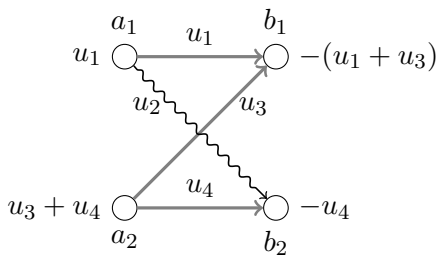
Abbildung 3.1: Ein Beispielnetzwerk mit Nullfahrzeiten und Einheitskapazitäten, in dem kein Earliest Arrival Transshipment existiert. Das Netzwerk ist aus [7].



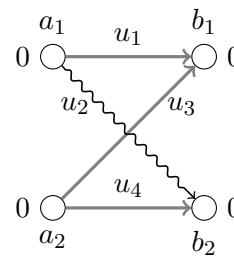
(a) Gegeben ist das hier abgebildete Netzwerk. Die Werte von u_1, u_2 und u_4 sind dabei echt positiv. Die gepunktete Kante gibt es evtl. nicht, was wir dadurch modellieren, dass u_3 nicht zwingend ein echt positiver Wert ist, sondern möglicherweise Null.



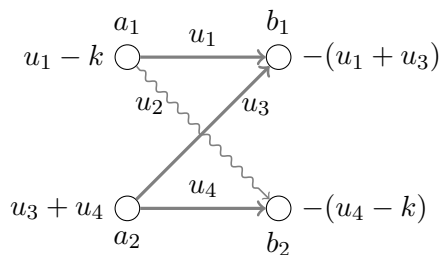
(b) Wir wählen die Bedarfe so, dass ein Fluss, der alle Angebote in zwei Zeitschritten aufbrauchen will, die geschlängelte Kante im ersten Zeitschritt nicht benutzen darf (wodurch im ersten Zeitschritt kein maximaler Fluss erreicht werden kann).



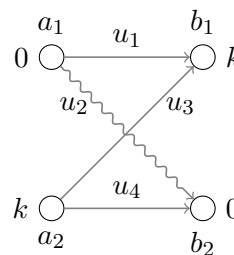
(c) Tatsächlich gibt es auch einen Fluss, der alle Angebote in zwei Zeitschritten aufbraucht. Dieser schickt im ersten Zeitschritt keinerlei Fluss über die geschlängelte Kante, lastet aber die anderen voll aus und erreicht damit einen Flusswert von $u_1 + u_3 + u_4$.



(d) Im zweiten Zeitschritt wird nun der gleiche Fluss erneut versandt, so dass insgesamt ein Flusswert von $2u_1 + 2u_3 + 2u_4$ erreicht wird und alle Angebote aufgebraucht sind.



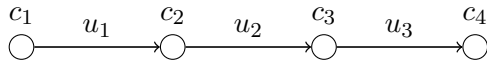
(e) Andererseits kann man den Flusswert maximieren, wenn man zusätzlich Fluss über die geschlängelte Kante schickt. Dort kann mindestens eine Einheit versendet werden. Wie viel verschickt werden kann, hängt von u_1, u_2, u_3 und u_4 ab. Wir gehen davon aus, dass ein maximaler (zulässiger) Wert von $k \geq 1$ Einheiten verschickt wird, so dass ein (maximaler) Flusswert von $u_1 + u_3 + u_4 + k$ erreicht wird.



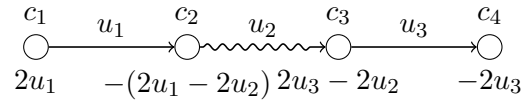
(f) Im zweiten Zeitschritt erhalten wir folgende Situation: Von a_1 können maximal $u_1 - k$ Einheiten ausgehen (unabhängig davon, über welche Kante), ebenso können nach b_2 maximal $u_4 - k$ Einheiten fließen. Zusätzlich können maximal u_3 Einheiten von a_2 nach b_1 fließen, so dass wir einen Flusswert von $u_1 + u_3 + u_4 - k$ erreichen. Insgesamt erhalten wir nur $2u_1 + 2u_3 + 2u_4 - k$ Flusseinheiten und damit weniger als möglich.

Abbildung 3.2: Diese Abbildung zeigt eine Verallgemeinerung von Beispiel 3.1.

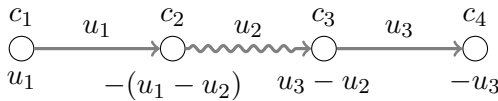
3 Dynamische Transshipments mit Nullfahrzeiten



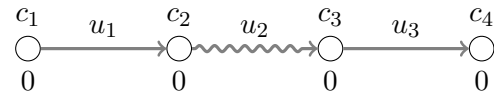
- (a) Gegeben ist das hier abgebildete Netzwerk. Im Gegensatz zum ersten Beispiel stellen wir weitergehendere Bedingungen an die gegebenen Kapazitäten: u_1, u_2 und u_3 müssen nicht nur echt positiv sein, zusätzlich soll $u_2 < u_1$ sowie $u_2 < u_3$ gelten.



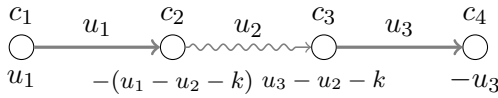
- (b) Hier wählen wir die Angebote so, dass ein Fluss, der alle Angebote in zwei Zeitschritten erfüllen will, im ersten Zeitschritt auch die geschlängelte Kante benutzen muss, obwohl dies (im ersten Zeitschritt) den Flusswert reduziert.



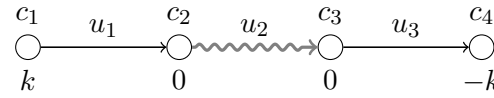
- (c) Der Fluss, der nach zwei Zeiteinheiten alle Angebote erfüllt, versendet im ersten Zeitschritt u_2 Flusseinheiten von c_1 nach c_4 . Die verbleibenden Kapazitäten nutzt er, um $u_1 - u_2 > 0$ Flusseinheiten von c_1 nach c_2 sowie $u_3 - u_2 > 0$ Flusseinheiten von c_3 nach c_4 zu transportieren.



- (d) Im zweiten Zeitschritt wird dies wiederholt, so dass erneut ein Flusswert von $u_1 - u_2 + u_3$ und insgesamt ein Flusswert von $2u_1 - 2u_2 + 2u_3$ verschickt wird.



- (e) Andererseits können wir den Flusswert im ersten Zeitschritt maximieren, wenn wir über die geschlängelte Kante weniger verschicken und stattdessen Fluss von c_1 nach c_2 sowie von c_3 nach c_4 senden. Wie viel Fluss wir auf diese Weise verschieben können, hängt von u_1, u_2 und u_3 ab, es ist aber mindestens 1 Einheit. Wir gehen davon aus, dass wir den maximalen Wert von $k \geq 1$ Einheiten verschieben und auf diese Weise einen Fluss mit Wert $u_1 - u_2 + u_3 + k$ erhalten.



- (f) Im zweiten Schritt können wegen des verbleibenden Angebots nur noch maximal $u_1 - u_2 - k$ Einheiten von c_1 nach c_2 sowie maximal $u_3 - u_2 - k$ Einheiten von c_3 nach c_4 verschickt werden. Von c_1 nach c_4 können maximal u_2 Einheiten gesendet werden, so dass wir im zweiten Schritt $u_1 - u_2 + u_3 - 2k$ und insgesamt $2u_1 - 2u_2 + 2u_3 - k$ Einheiten erhalten, was nicht optimal ist.

Abbildung 3.3: Diese Abbildung beschreibt für ein weiteres Netzwerk, wie man die Potentiale wählen kann, damit es kein Earliest Arrival Transshipment gibt.

In Teilabbildung (b) sieht man, wie man die Potentiale wählen kann, um in eine Situation ohne Earliest Arrival Transshipment zu geraten. Die Teilabbildungen (c) und (d) zeigen, wie man den zweiten Zeitschritt in dieser Situation maximiert, die Teilabbildungen (e) und (f) stellen dar, wie man im ersten Zeitschritt maximal viel verschickt, dafür den maximalen Wert im zweiten Zeitschritt dann nicht erreichen kann.

Im Folgenden wollen wir weitere Netzwerke und Klassen von Netzwerken bzgl. der Existenz von Earliest Arrival Transshipments analysieren. Wir haben in Abschnitt 2.4 bereits einen einfachen Test gesehen, mit dem man (in pseudopolynomieller Zeit) überprüfen kann, ob ein Netzwerk mit gegebenen Potentialen und Kapazitäten ein Earliest Arrival Transshipment besitzt. Dieser funktioniert natürlich auch für Netzwerke mit Nullfahrzeiten. Andererseits sind die Potentiale in einer tatsächlichen Anwendung möglicherweise im Vorhinein gar nicht bekannt. Es ist daher ebenfalls spannend, zu untersuchen, ob es Netzwerke gibt, die für beliebige Potentiale, beliebige Kapazitäten und sogar beliebige Wahlen von Quellen- und Senkenmengen ein Earliest Arrival Transshipment besitzen. Auf diese Frage gehen wir in Abschnitt 3.3 ein.

Es macht keinen Sinn, Graphen darauf zu untersuchen, ob sie *manchmal* (für einige Bedarfe oder Kapazitäten) ein Earliest Arrival Transshipment besitzen, da dies immer der Fall ist: Sei ein Graph mit beliebigen Kapazitäten gegeben. Dann fügt man vorübergehend eine Superquelle und -senke ein und berechnet einen maximalen Fluss. Anschließend gibt man jeder Quelle als Angebot den Fluss, der im maximalen Fluss auf der Kante von der Supersenke zu dieser Quelle fließt, für Senken verfährt man analog. Wenn man einen Fluss mit mehr als einem Zeitschritt wünscht, kann man die Bedarfe auch noch mit einer beliebigen Zahl multiplizieren - man erhält auf jeden Fall eine Situation, in der ein Earliest Arrival Transshipment existiert.

Bevor wir bestimmte Graphstrukturen auf die Existenz von Earliest Arrival Transshipments untersuchen, überlegen wir uns nun zunächst in Abschnitt 3.2, wie sich die Betrachtung von Nullfahrzeiten auf die verschiedenen dynamischen Transshipments auswirkt.

3.2 Netzwerke mit Nullfahrzeiten

In diesem Abschnitt wollen wir einige einfache Beobachtungen machen, die aus dieser Einschränkung der Fahrzeitenfunktion folgen und uns ansehen, wie sich dynamische Nullfahrzeitenflüsse verhalten.

Notation 39. Mit τ^0 ist die Nullfahrzeitenfunktion $\tau^0 : E \rightarrow \mathbb{Z}_{\geq 0}$ mit $\tau \equiv 0$ gemeint.

3 Dynamische Transshipments mit Nullfahrzeiten

Die wichtigste Beobachtung ist die folgende: Durch Nullfahrzeiten sind Flusseinheiten nicht mehr über mehrere Zeitschritte unterwegs, sondern erreichen ihr Ziel in dem Zeitschritt, in dem sie gestartet sind. Der dynamische Fluss zerfällt dadurch in eine Abfolge von Flüssen, bestehend aus allen Flusseinheiten, die im gleichen Zeitschritt unterwegs sind.

Lemma und Definition 40. Sei $N = (G, u, \tau^0, -, S^+, S^-)$ ein gerichtetes Netzwerk und sei f ein dynamischer Nullfahrzeitenfluss in N mit Zeithorizont T . Für festes $t \in \{1, \dots, T\}$ ist die Funktion $f_t : E \rightarrow \mathbb{Z}_{\geq 0}$ mit $f_t(e) := f(e, t)$ ein zulässiger statischer Fluss in N , der Flusserhaltung respektiert. Wir nennen f_t den t -ten Teilfluss von f .

Beweis. Die Kapazitätsbedingungen werden von f_t eingehalten, da f sie in jedem Zeitschritt einhält. Außerdem vereinfacht sich die Berechnung des Flusswertes an einem Knoten v zur Zeit t durch die Nullfahrzeiten:

$\forall v \in V, t = 1, \dots, T :$

$$\begin{aligned} \text{wert}_f(v, t) &= \sum_{e \in \delta^-(v) \wedge t - \tau(e) \geq 1} f(e, t - \tau(e)) - \sum_{e \in \delta^+(v)} f(e, t) \\ &= \sum_{e \in \delta^-(v)} f(e, t) - \sum_{e \in \delta^+(v)} f(e, t) \\ &= \text{wert}_{f_t}^{\text{st}}(v). \end{aligned}$$

Da f Flusserhaltung erfüllt, gilt insbesondere für alle $v \in V \setminus (S^+ \cup S^-)$ $\text{wert}_f(v, t) = 0$. Dadurch ist auch $\text{wert}_{f_t}^{\text{st}}(v) = 0$ für alle $v \in V \setminus (S^+ \cup S^-)$ erfüllt, d.h. f_t respektiert Flusserhaltung. \square

Alternativ zur Vorstellung von dynamischen Nullfahrzeitenflüssen als eingeschränkte Klasse dynamischer Flüsse kann man sie so auch als Erweiterung statischer Flüsse auffassen: Nehmen wir an, wir beschäftigen uns mit einem Problem mit vorgegebenen Flussmengen, zum Beispiel dem Transport eines bestimmten Guts von bestimmten Herstellern zu bestimmten Abnehmern mit Hilfe von Transportfahrzeugen. Jeder Hersteller stellt eine bestimmte Menge des Guts zum Transport bereit, jeder Abnehmer möchte eine bestimmte Menge erhalten. Das Straßennetz ist als Netzwerk gegeben, und die Transportwege zwischen Herstellern und Empfängern sind alle ungefähr gleich lang. Wir modellieren den Güterfluss daher als statischen Fluss. Dabei stellen wir fest, dass das Netzwerk nicht alle Güter gleichzeitig transportieren kann – wir müssen die Güter in mehreren Schritten verschicken. Da alle Fahrzeuge zuerst zurückfahren müssen, bevor der nächste Teil der Ware transportiert werden kann, kommt es nun vor allem auf die Anzahl der Schritte an, die benötigt werden, um alle Güter zu versenden. Gesucht wird also ein Quickest Transshipment in einem Netzwerk mit Nullfahrzeiten.

Neben der Steuerung von Verkehr können dynamische Nullfahrzeitenflüsse z.B. auch für spezielle Fälle von Routing in Datennetzwerken nützlich sein. Das ist besonders interessant, weil sich dynamische Flussfahrprobleme in Netzwerken mit Nullfahrzeiten stark vereinfachen, wie wir als nächstes sehen werden.

Flussprobleme

Aufbauend auf Lemma 40 gehen wir jetzt darauf ein, wie sich die von uns betrachteten Netzwerkflüsse in Netzwerken mit Nullfahrzeiten verhalten und richten uns dabei nach Ideen aus [7].

Die zentrale Erkenntnis, die wir benötigen, ist die Tatsache, dass wir dynamische Transshipments mit Nullfahrzeiten auf einfache Weise durch die Berechnung eines statischen maximalen Flusses in einem erweiterten Netzwerkes finden können.

Definition und Lemma 41. Sei $N = (G, u, \tau^0, p, S^+, S^-)$ ein gerichtetes Netzwerk. Wir definieren das Netzwerk $N^{\times T} := (G^{\times T}, u^{\times T}, -, (S^+)^{\times T}, (S^-)^{\times T})$ durch:

$$\begin{aligned}
 G^{\times T} &:= (V^{\times T}, E^{\times T}) \\
 V^{\times T} &:= V \cup \{s_*, t_*\} \\
 E^{\times T} &:= E \cup \{(s_*, s) \mid \forall s \in S^+\} \cup \{(t, t_*) \mid t \in S^-\} \\
 u^{\times T}(e) &:= T \cdot u(e) && \forall e \in E \\
 u^{\times T}(s_*, s) &:= |p(s)| && \forall s \in S^+ \\
 u^{\times T}(t, t_*) &:= |p(t)| && \forall t \in S^- \\
 (S^+)^{\times T} &:= \{s_*\} \\
 (S^-)^{\times T} &:= \{t_*\}
 \end{aligned}$$

Dann induziert jeder statische Fluss in $N^{\times T}$, der zulässig ist und Flusserhaltung erfüllt, einen ausgleichenden Nullfahrzeitenfluss in N mit Zeithorizont T mit gleichem Flusswert, und umgekehrt. Insbesondere stimmt der Wert eines maximalen zulässigen Flusses in $N^{\times T}$, der Flusserhaltung erfüllt, mit dem maximalen Wert $w_{\max}^+(t)$, den ein ausgleichender Nullfahrzeitenfluss bis zum Zeitpunkt T erreichen kann, überein.

Beweis. Sei f ein (dynamischer) ausgleichender Nullfahrzeitenfluss. Wir bilden den statischen Fluss $f^+ : E^{\times T} \rightarrow \mathbb{Z}_{\geq 0}$ durch $f^+(e) := \sum_{t=1}^T f(e, t) \forall e \in E$ sowie $f^+(s_*, s) := -\text{wert}_f(s) \forall s \in S^+$ und $f^+(t, t_*) := \text{wert}_f(t) \forall t \in S^-$. f^+ erfüllt die Kapazitätsbedingungen bzgl. $u^{\times T}$ für $e \in E$, da f die Kapazitäten u in jedem der T Zeitschritte einhält und die

3 Dynamische Transshipments mit Nullfahrzeiten

Kapazitäten bzgl $u^{\times T}$ T -mal so groß sind wie bzgl. u . Für $e \in \{(s_*, s) \mid \forall s \in S^+\}$ werden die Kapazitäten eingehalten, da $|\text{wert}_f(s)| \leq |p(s)| = u(s_*, s)$ gilt, für $e \in \{(t, t_*) \mid \forall t \in S^+\}$, weil $|\text{wert}_f(t)| \leq |p(t)| = u(t, t_*)$ gilt. Außerdem erfüllt f^+ Flusserhaltung für alle $v \in V$, da wir f^+ durch Addition statischer Flüsse erhalten, die Flusserhaltung erfüllten. An $s \in S^+$ gilt Flusserhaltung, da $\text{wert}_{f^+}(s) = \text{wert}_f(s) + f(s_*, s) = \text{wert}_f(s) - \text{wert}_f(s) = 0$ ist, in $t \in S^-$ gilt Flusserhaltung wegen $\text{wert}_f(f^+)(t) = \text{wert}_f(t) - f(t, t_*) = \text{wert}_f(t) - \text{wert}_f(t) = 0$. Außerdem stimmen die Werte der beiden Flüsse aufgrund von $\text{wert}_{f^+}^{\text{st}} = \text{wert}_{f^+}^{\text{st}}(t_*) = \sum_{t \in S^-} f^+(t, t_*) = \sum_{t \in S^-} \text{wert}_f(t) = \text{wert}_f$ überein.

Wenn x ein zulässiger Fluss in $N^{\times T}$ ist, der Flusserhaltung respektiert, dann definieren wir $x^{/T} : E \times \{1, \dots, T\} \rightarrow \mathbb{Z}_{\geq 0}$ durch $x^{/T}(e, t) := \frac{x(e)}{T}$. $x^{/T}$ erfüllt für alle $e \in E$ und alle $t \in \{1, \dots, T\}$ die Kapazitätsbedingungen bzgl. u , da x die Kapazitäten $u^{\times T}$ einhält und $u(e) = \frac{u^{\times T}}{T}$ ist. Außerdem überträgt sich auch die Flusserhaltung von x auf $x^{/T}$. $x^{/T}$ ist ausgleichend, da $|\text{wert}_{x^{/T}}(s)| = \left| \sum_{t'=1}^T \text{wert}_{x^{/T}}(s, t') \right| = \left| \sum_{t'=1}^T \frac{\text{wert}_x^{\text{st}}(s) - x(s_*, s)}{T} \right| = \left| T \frac{\text{wert}_x^{\text{st}}(t) s x(s_*, s)}{T} \right| = |\text{wert}_x^{\text{st}}(t) s x(s_*, s)| = |0 - x(s_*, s)| \leq |p(t)|$ und analog $|\text{wert}_{x^{/T}}(t)| = \sum_{t'=1}^T \text{wert}_{x^{/T}}(t, t') = \sum_{t'=1}^T \frac{\text{wert}_x^{\text{st}}(t) + x(t, t_*)}{T} = T \frac{\text{wert}_x^{\text{st}}(t) + x(t, t_*)}{T} = \text{wert}_x^{\text{st}}(t) + x(t, t_*) = 0 + x(t, t_*) \leq |p(t)|$ gilt. Die gleiche Rechnung hilft uns zu zeigen, dass die Werte der beiden Flüsse übereinstimmen: Es gilt $\text{wert}_{x^{/T}} = \sum_{t \in S^-} \text{wert}_{x^{/T}}(t) = \sum_{t \in S^-} x(t, t_*) = \text{wert}_x$. \square

Dynamic Transshipment Lemma 41 bedeutet, dass es in einem Netzwerk N mit Nullfahrzeiten genau dann ein dynamisches Transshipment mit Zeithorizont T gibt, wenn es in $N^{\times T}$ einen maximalen Fluss gibt, der den Flusswert $p(S^-) := \sum_{t \in S^-}$ erreicht. Ist dies der Fall, so können wir auch direkt ein dynamisches Transshipment angeben, indem wir in jedem der T Zeitschritte ein T -tel des Flusses in $N^{\times T}$ versenden. Dieses dynamische Transshipment hat den Nachteil, dass es nicht notwendigerweise ganzzahlig ist, auch dann nicht, wenn es ein ganzzahliges Transshipment gibt. Die Berechnung eines ganzzahligen dynamischen Transshipments ist komplizierter. Fleischer entwickelt in [7] einen Algorithmus für dieses Problem, der eine polynomielle Anzahl von Berechnungen maximaler Flüsse durchführt.

Quickest Transshipment Den minimalen Zeithorizont T^* kann man z.B. finden, indem man den gerade beschriebenen in eine binäre Suche einbettet und dabei für jeden zu testenden Zeithorizont jeweils einen maximalen Fluss in $N^{\times T}$ berechnet. Für eine streng polynomielle Laufzeit verwendet man statt der binären Suche das Framework zur parametrisierten Suche von Meggido [22, 23]. Zum optimalen Zeithorizont kann mit Hilfe des

Algorithmus von Fleischer auch ein ganzzahliges Quickest Transshipment berechnet werden.

Earliest Arrival Transshipments Die Earliest Arrival Eigenschaft lässt sich für Netzwerke mit Nullfahrzeiten jetzt recht einfach fassen: Ein Earliest Arrival Transshipment muss in Zeitschritt t so viel Fluss verschicken, wie ein statischer Fluss in $N^{\times t}$ maximal verschicken kann, d.h.

$$w_{\max}^+(t) = \max\{\text{wert}_x^{\text{st}} \mid x \text{ ist zulässiger Fluss in } N^{\times t} \text{ und respektiert Flusserhaltung}\}.$$

Die Bestimmung eines Earliest Arrival Transshipments ist jedoch nicht so einfach. Zwar können wir für jeden Zeitpunkt t mit Hilfe von $N^{\times T}$ einen Fluss mit Zeithorizont t berechnen, der nach t Zeitschritten tatsächlich $w_{\max}^+(t)$ erreicht, wir können aber dabei nicht sicherstellen, dass der Fluss auch für die Zeitschritte vor t optimal ist. Und obwohl es auf den ersten Blick so aussieht, als könnte man mit etwas mehr Geschick doch immer ein Earliest Arrival Transshipment berechnen, wissen wir bereits, dass dies (bei Netzwerken mit mehreren Quellen und Senken) im Allgemeinen nicht möglich ist, weil ein solches gar nicht immer existiert. Gerade dadurch wird die Betrachtung von Nullfahrzeitenflüssen für uns so spannend.

Für Netzwerke mit nur einer Senke wissen wir jedoch bereits aus Abschnitt 2.2, dass die Existenz sogar im allgemeinen Fall gesichert ist. Für solche Netzwerke geben Hajek und Ogier in [17] einen Algorithmus an, der mit Hilfe von $O(n)$ Berechnungen maximaler Flüsse ein Earliest Arrival Transshipment berechnet. Fleischer präsentiert in [7] einen Algorithmus, dessen Laufzeit $O(mn \log(n^2/m))$ ist.

3.3 Existenzaussagen

In der Einleitung haben wir in Abbildung 1.2 gesehen, dass Earliest Arrival Transshipments in Netzwerken mit mehreren Quellen und einer Senke bei allgemeinen Fahrzeiten nicht unbedingt existieren – und das, obwohl sie in Netzwerken mit einer Quelle und mehreren Senken immer existieren. Im Fall von Nullfahrzeiten ist die Situation hingegen symmetrisch. Um das zu sehen, definieren zuerst wir zu einem Netzwerk N das Netzwerk, das man durch Umdrehen aller Kanten erhält.

Definition 42. Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk. Wir definieren \overleftarrow{G} als den Graphen mit Knotenmenge V und Kantenmenge $\overleftarrow{E} := \{(w, v) \mid v, w \in V, (v, w) \in E\}$, außerdem sei $\overleftarrow{u} : E \rightarrow \mathbb{Z}_{\geq 0}$ definiert durch $\overleftarrow{u}(w, v) = u(v, w) \forall (v, w) \in E$ und $\overleftarrow{p} : V \rightarrow \mathbb{Z}_{\geq 0}$ definiert durch $\overleftarrow{p}(v) = -p(v) \forall v \in V$.

3 Dynamische Transshipments mit Nullfahrzeiten

Nullfahrzeitenflüsse in N können leicht in entsprechende Flüsse in \overleftarrow{N} überführt werden:

Lemma 43. *Sei $N = (G, u, \tau, p, S^+, S^-)$ ein gerichtetes Netzwerk. Dann existiert in N genau dann ein Earliest Arrival Transshipment, wenn in $\overleftarrow{N} = (\overleftarrow{G}, \overleftarrow{u}, \tau^0, \overleftarrow{p}, S^-, S^+)$ ein Earliest Arrival Transshipment existiert.*

Beweis. Zu einer Kante $e = (v, w) \in E$ notieren wir durch \overleftarrow{e} die Kante $(w, v) \in \overleftarrow{E}$. Jeder ausgleichende Fluss f entspricht einem ausgleichenden Fluss \overleftarrow{f} in \overleftarrow{N} , der durch $\overleftarrow{f}(e, t) := f(\overleftarrow{e}, t) \forall (v, w) \in E$ definiert ist. Ebenso entspricht jedem ausgleichenden Fluss \overleftarrow{f}' in \overleftarrow{N} ein ausgleichender Fluss f' in N , der durch $f'(e, t) := \overleftarrow{f}'(\overleftarrow{e}, t) \forall (v, w) \in E$ definiert ist. Die Kapazitätsbedingungen werden dabei jeweils aufgrund der Definition von \overleftarrow{u} eingehalten und die Flusserhaltung überträgt sich für Knoten $v \in V \setminus (S^+ \cup S^-)$, da die eingehenden und ausgehenden Kanten an einem Knoten nur die Rollen tauschen, was keinen Unterschied macht, wenn die Summe der Flusswerte auf den eingehenden Kanten der Summe der Flusswerte auf den ausgehenden Kanten entspricht. An den Knoten in $(S^+ \cup S^-)$ wechselt das Vorzeichen des Flusswertes, passend dazu, dass die Quellen- und Senkenmenge die Rollen tauschen. Dadurch ist die Flusserhaltung an diesen Knoten ebenfalls erfüllt. Die Flusswerte der Flüsse stimmen aufgrund von Lemma 20 in jedem Zeitschritt überein. \square

Aufgrund von Lemma 43 existiert in einem Netzwerk N mit Nullfahrzeiten und einer Quelle ein Earliest Arrival Transshipment, weil es in \overleftarrow{N} ein Earliest Arrival Transshipment gibt, da \overleftarrow{N} nur eine Senke hat.

Wenn man einen dynamischen Fluss f mit allgemeinen Fahrzeiten und Zeithorizont T von N nach \overleftarrow{N} zu übertragen will, kann man nicht $\overleftarrow{f}(e, t) = f(\overleftarrow{e}, t)$ verwenden, da man zur Wahrung der Flusserhaltung die Fahrzeiten berücksichtigen muss. Ohne Beweis merken wir an, dass $\overleftarrow{f}(e, t) = f(\overleftarrow{e}, T - t + 1)$ eine mögliche Wahl ist, für die die Flusserhaltung erhalten bleibt. Durch diesen Zusammenhang zwischen den Flüssen in N und \overleftarrow{N} bleibt die Earliest Arrival Eigenschaft jedoch nicht erhalten. Stattdessen erfüllt der Fluss in \overleftarrow{f} die *Latest Departure Eigenschaft*: Zu jedem Zeitpunkt t ist die Flussmenge, die die Quelle in den Zeitschritten $t + 1$ bis T verlässt, maximal (Latest Departure Flüsse werden z.B. in [27, 24] definiert). Tatsächlich kann man das zeitexpandierte Netzwerk aus Abschnitt 2.2 für Netzwerke mit einer Quelle und einen vorgegebenen Zeithorizont T analog aufbauen und auf diese Weise zeigen, dass in solchen Netzwerken immer ein ausgleichender Fluss existiert, der die Latest Departure Eigenschaft erfüllt. Mit dieser Thematik beschäftigen wir uns aber hier nicht näher.

Im Folgenden sehen wir uns zwei spezielle Klassen von Netzwerken an, in denen es immer ein Earliest Arrival Transshipment gibt. In Abschnitt 3.3.1 geht es um Netzwerke, die

einen ausgezeichneten Knoten v besitzen, der eine Engstelle darstellt: Alle Wege von Quellen zu Senken führen durch v . Diese Netzwerke sind deshalb interessant, weil der Fluss sich im Knoten v bündeln muss. Dadurch übernimmt v für das Teilnetzwerk, das die Quellen enthält, die Funktion einer Senke, und für das Teilnetzwerk, das die Senken enthält, die Funktion einer Quelle. Wir werden diese Tatsache ausnutzen und zeigen, dass es in Netzwerken von diesem Typ für beliebige Kapazitäten und Potentiale immer ein Earliest Arrival Transshipment gibt. Im Anschluss werden wir in Abschnitt 3.3.2 In- und Out-Trees mit beschränkter Tiefe betrachten. Diese besitzen nicht nur für beliebige Kapazitäten und Potentiale, sondern auch unabhängig von der Verteilung der Quellen und Senken im Netzwerk immer ein Earliest Arrival Transshipment.

Mit Hilfe der Erkenntnisse aus Abschnitt 3.3.1 und 3.3.2 werden wir schließlich in 3.3.3 vollständig charakterisieren, in welchen gerichteten Netzwerken für beliebige Kapazitäten, Potentiale und Wahlen der Quellen- und Senkenmenge immer ein Earliest Arrival Transshipment existiert.

3.3.1 Netzwerke mit einer Engstelle

Wir betrachten jetzt eine spezielle Netzwerkkategorie, die Netzwerke enthält, die einen Knoten als Engstelle besitzen. Alle Pfade von Quellen zu Senken verlaufen durch diesen ausgezeichneten Knoten (siehe auch Abbildung 3.4).

Definition 44. Ein Netzwerk $N_v = (G, -, -, -, S^+, S^-)$ gehört zur Klasse \mathcal{N}_\bullet , wenn es einen ausgezeichneten Knoten $v \in G$ gibt, so dass der induzierte ungerichtete Graph G_U zu G durch Herausnahme von v und allen zu v adjazenten Kanten in genau zwei Zusammenhangskomponenten $G_U^L = (V_U^L, E_U^L)$ und $G_U^R = (V_U^R, E_U^R)$ mit $S^+ \subseteq V_U^L$ und $S^- \subseteq V_U^R$ zerfällt. Außerdem bezeichnen wir mit G_L den von $V_U^L \cup \{v\}$ induzierte Teilgraphen von G und mit G_R den von $V_U^R \cup \{v\}$ induzierten Teilgraphen von G .

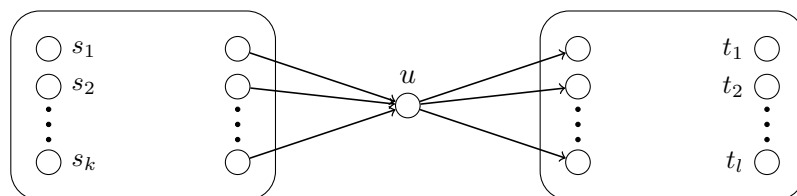


Abbildung 3.4: Schematische Darstellung eines Netzwerks aus \mathcal{N}_\bullet .

In jedem Netzwerk $N \in \mathcal{N}_\bullet$ gibt es für beliebige Kapazitätsfunktionen und Potentiale immer ein Earliest Arrival Transshipment.

3 Dynamische Transshipments mit Nullfahrzeiten

Um dies zu zeigen, beweisen wir zuerst Lemma 46, das uns erlaubt, Fluss in begrenztem Maße zwischen verschiedenen Teilflüssen eines Nullfahrzeitentransshipments zu verschieben. Für den Beweis von Lemma 46 benötigen wir die Aussage von Lemma 45.

Lemma 45. *Sei $N = (G, u, -, -, S^+, S^-)$ ein Netzwerk und seien x_1 und x_2 zwei zulässige statische Flüsse in N , die Flusserhaltung erfüllen. Sei $\alpha \in [0, 1]$. Dann ist der Fluss $x_\alpha : E \rightarrow \mathbb{Z}_{\geq 0}$, definiert durch $x_\alpha(e) := \alpha x_1(e) + (1 - \alpha)x_2(e)$, ebenfalls zulässig in N und erfüllt Flusserhaltung.*

Beweis. Der Fluss x_α hält die Kapazitätsbedingungen in N ein, da $x_\alpha(e) = \alpha x_1(e) + (1 - \alpha)x_2(e) \leq \alpha u(e) + (1 - \alpha)u(e) = u(e)$ gilt. Außerdem gilt $\text{wert}_{x_\alpha}(v) = \alpha \text{wert}_{x_1}(e) + (1 - \alpha) \text{wert}_{x_2}(e) \forall v \in V$, so dass sich auch die Flusserhaltung überträgt. \square

Lemma 46. *Sei $N = (G, u, \tau^0, p, S^+, S^-)$ ein gerichtetes Netzwerk und sei f ein ausgleichender Nullfahrzeitenfluss mit Zeithorizont T in N . Außerdem seien f_i und f_j zwei Teilflüsse von f mit $i, j \in \{1, \dots, T\}$ und $\text{wert}_{f_i}^{st} > \text{wert}_{f_j}^{st}$ und sei $a \in \mathbb{R}$ eine beliebige Zahl im Intervall $\left[\frac{\text{wert}_{f_i}^{st} + \text{wert}_{f_j}^{st}}{2}, \text{wert}_{f_i}^{st} \right]$.*

Dann gibt es zwei (statische) Flüsse f'_i und f'_j mit $\text{wert}_{f'_i}^{st} = a$ und $\text{wert}_{f'_j}^{st} = \frac{\text{wert}_{f_j}^{st} + \text{wert}_{f_i}^{st}}{2} - a$, so dass man durch Austauschen von f_i durch f'_i und f_j durch f'_j in f wieder einen ausgleichenden Nullfahrzeitenfluss in N mit gleichem Flusswert erhält.

Beweis. Zu zwei statischen Flüssen $x_1 : E \rightarrow \mathbb{Z}_{\geq 0}$ und $x_2 : E \rightarrow \mathbb{Z}_{\geq 0}$ definieren wir $x_1 + x_2 : E \rightarrow \mathbb{Z}_{\geq 0}$ durch $(x_1 + x_2)(e) = x_1(e) + x_2(e) \forall e \in E$. Zu einem statischen Fluss $x : E \rightarrow \mathbb{Z}_{\geq 0}$ und einer reellen Zahl $\alpha \in \mathbb{R}$ definieren wir $\alpha x : E \rightarrow \mathbb{Z}_{\geq 0}$ durch $(\alpha x)(e) = \alpha \cdot x(e) \forall e \in E$. Wir beobachten, dass man f_i und f_j ohne Schwierigkeiten durch f'_i und f'_j austauschen kann, wenn diese in N zulässig sind sowie Flusserhaltung erfüllen und wenn zusätzlich $(f'_i + f'_j)(e) = (f_i + f_j)(e) \forall e \in E$ gilt. In diesem Fall erreichen f'_i und f'_j zusammen nämlich an allen Quellen und Senken dieselben Flusswerte wie f_i und f_j und können die Eigenschaft von f , ausgleichend zu sein, nicht gefährden.

Sei $N^{\times 2}$ das Netzwerk, das man durch Verdoppeln aller Kapazitäten in N und anschließendes Einfügen von Superquelle und Supersenke erhält (siehe Definition und Lemma 41). $f_i + f_j$ ist ein zulässiger statischer Fluss in $N^{\times 2}$, der Flusserhaltung erfüllt. Aus dem Beweis von Lemma 41 wissen wir, dass der dynamische Fluss, der zweimal hintereinander den Teilfluss $f_h : E \rightarrow \mathbb{Z}_{\geq 0}$, definiert durch $f_h(e) = \frac{f_i(e) + f_j(e)}{2}$, verschickt, ein ausgleichender Fluss in N ist. Insbesondere ist f_h daher ein zulässiger statischer Fluss in N , der Flusserhaltung respektiert. Außerdem erfüllt f_h die Gleichung $(f_h + f_h)(e) = (f_i + f_j)(e) \forall e \in E$. Es ist also möglich, f_i und f_j in f durch zweimaliges Versenden von f_h zu ersetzen. Das ist

auch die von uns gewünschte Lösung, falls $a = \frac{\text{wert}_{f_i}^{\text{st}} + \text{wert}_{f_j}^{\text{st}}}{2}$ ist, da $\text{wert}_{f_h} = \frac{\text{wert}_{f_i}^{\text{st}} + \text{wert}_{f_j}^{\text{st}}}{2}$ gilt.

Andernfalls setze $\alpha := \frac{a - \text{wert}_{f_h}^{\text{st}}}{\text{wert}_{f_h}^{\text{st}} - \text{wert}_{f_i}^{\text{st}}} \in [0, 1]$, d.h. es gilt $\alpha \cdot \text{wert}_{f_i}^{\text{st}} + (1 - \alpha)(\text{wert}_{f_h}^{\text{st}}) = a$. Aufbauend darauf definieren wir $f'_i = \alpha f_i + (1 - \alpha)f_h$ sowie $f'_j = (1 - \alpha)f_i + \alpha f_h$. Beide sind aufgrund von Lemma 45 in N zulässig und erfüllen Flusserhaltung, da dies für f_i und für f_h gilt. Außerdem ist

$$f'_i + f'_j = \alpha f_i + (1 - \alpha)f_h + (1 - \alpha)f_i + \alpha f_h = f_i + f_j$$

Wir können also f_i und f_j in f durch f'_i und f'_j austauschen und erfüllen damit die Bedingung des Lemmas, weil $\text{wert}_{f'_i}^{\text{st}} = \alpha \text{wert}_{f_i}^{\text{st}} + (1 - \alpha) \text{wert}_{f_h}^{\text{st}} = a$ sowie $\text{wert}_{f'_j}^{\text{st}} = \text{wert}_{f_j}^{\text{st}} + \text{wert}_{f_i}^{\text{st}} - a$ gilt. \square

Satz 47. Sei $N'_v = (G, u, -, -, S^+, S^-) \in \mathcal{N}_\bullet$. Sei $p : V \rightarrow \mathbb{Z}_{\geq 0}$ eine beliebige zulässige Potentialfunktion. Dann gibt es in $N_v = (G, u, \tau^0, p, S^+, S^-)$ ein Earliest Arrival Transshipment.

Beweis. Sei $v \in V$ der ausgezeichnete Knoten in N'_v und seien G_L und G_R definiert wie in Definition 44. Dann ist $N_L = (G, u, \tau^0, p, S^+, \{v\})$ ein Netzwerk mit einer Senke und enthält deshalb ein Earliest Arrival Transshipment f_L , während $N_R = (G, u, \tau^0, p, \{v\}, S^-)$ ein Netzwerk mit einer Quelle ist und deshalb ein Earliest Arrival Transshipment f_R enthält. Wir definieren die akkumulierten Flusswerte für f_L und f_R durch

$$w_{f_L}^+ := \sum_{t'=1}^t \text{wert}_{f_L}(t')$$

$$w_{f_R}^+ := \sum_{t'=1}^t \text{wert}_{f_R}(t').$$

Behauptung: Für jedes $t \geq 1$ gilt, dass ein ausgleichender Fluss in N in den ersten t Zeitschritten maximal $w^+(t) := \min \left\{ w_{f_L}^+(t), w_{f_R}^+(t) \right\}$ Flusseinheiten verschicken kann. Es gibt ein Earliest Arrival Transshipment in N , dass diesen Flusswert tatsächlich erreicht.

Wir setzen $w(t) = w^+(t) - w^+(t-1) \forall t \geq 2$ und $w(1) = w^+(1)$. Dann ist das Erreichen von $w^+(t)$ Flusseinheiten innerhalb der ersten t Zeitschritte äquivalent dazu, in den Zeitschritten $t' = 1$ bis $t' = t$ jeweils den Flusswert $w(t')$ zu verschicken.

Als erstes beobachten wir, dass $w^+(t)$ eine obere Schranke für den maximalen Flusswert nach t Zeitschritten ist, da der maximale Flusswert nach t Zeitschritten in mindestens einer der beiden Netzwerkhälften durch $w^+(t)$ beschränkt ist.

3 Dynamische Transshipments mit Nullfahrzeiten

Die Schwierigkeit besteht darin, ein Earliest Arrival Transshipment anzugeben, also einen Fluss, der in jedem Zeitschritt $t \geq 1$ den Wert $w(t)$ erreicht. Wären $w_{f_L}^+(t)$ und $w_{f_R}^+(t)$ für alle Zeitpunkte identisch, so könnten wir den gewünschten Fluss für jeden Zeitschritt zusammensetzen, indem wir den t -ten Teilfluss von f_L verwenden, um $w^+(t)$ Flusseinheiten von S^+ nach v zu transportieren und dann den t -ten Teilfluss in f_R verwenden, um sie von v nach S^- zu bringen.

Im Allgemeinen sind die Flusswerte von f_L und f_R in einem Zeitschritt jedoch nicht identisch. In diesem Fall „verschieben“ wir Fluss zwischen den einzelnen Zeiteinheiten, um sowohl in N_L als auch in N_R ein Transshipment zu finden, das den Flusswert $w(t)$ in jedem Zeitschritt t erreicht.

Wir betrachten jetzt den Fluss f_L in N_L und passen ihn schrittweise an, bis wir einen Fluss erhalten, der in jedem Zeitschritt $t \geq 1$ den Flusswert $w(t)$ erreicht. Die dabei entstehenden Zwischenflüsse bezeichnen wir weiterhin mit f_L , ebenso verwenden wir $w_{f_L}^+$ weiterhin für den akkumulierten Flusswert des veränderten Flusses.

Solange $w_{f_L}^+(t) = w^+(t)$ gilt, ist auch $\text{wert}_{f_L}(t) = w(t)$. Sei t' der erste Zeitpunkt, zu dem $w_{f_L}^+(t') > w^+(t')$ und damit auch $\text{wert}_{f_L}(t') > w(t')$ gilt. Mit t'' bezeichnen wir den ersten Zeitpunkt nach t' , zu dem $\text{wert}_{f_L}(t'') < w(t'')$ gilt (diesen Zeitpunkt muss es geben, da $w^+(t)$ und $w_{f_L}^+(t)$ ab irgendeinem t den gleichen (maximalen) Wert annehmen).

An dieser Stelle benötigen wir, dass $w(t)$ monoton fallend ist. Wir zeigen diese Aussage im Anschluss an den Beweis in Lemma 48.

Da $w(t)$ monoton fallend ist, gilt $w(t'') \leq w(t')$ und somit auch $\text{wert}_{f_L}(t') > w(t') \geq w(t'') > \text{wert}_{f_L}(t'')$. Wir setzen $b := \min\{\text{wert}_{f_L}(t') - w(t'), w(t'') - \text{wert}_{f_L}(t'')\}$, d.h. $b \geq 0$ und $b \leq \text{wert}_{f_L}(t') - w(t')$, $b \leq w(t'') - \text{wert}_{f_L}(t'')$. Daraus folgt $2b \leq \text{wert}_{f_L}(t') - w(t') + w(t'') - \text{wert}_{f_L}(t'') = \text{wert}_{f_L}(t') - \text{wert}_{f_L}(t'') + w(t'') - w(t') \leq \text{wert}_{f_L}(t') - \text{wert}_{f_L}(t'')$, da $w(t)$ monoton fallend ist. Wir setzen jetzt $a := \text{wert}_{f_L}(t') - b$ und wissen, dass a wegen $0 \leq b \leq \frac{\text{wert}_{f_L}(t') - \text{wert}_{f_L}(t'')}{2}$ im Intervall $[\frac{\text{wert}_{f_L}(t') + \text{wert}_{f_L}(t'')}{2}, \text{wert}_{f_L}(t')]$ liegt. Wir können daher Lemma 46 benutzen, um den Fluss in den Zeitschritten t' und t'' in f_L so umzuverteilen, dass zum Zeitpunkt t' der Flusswert a und zum Zeitpunkt t'' der Flusswert $\text{wert}_{f_L}(t'') + \text{wert}_{f_L}(t') - a$ versendet wird. Damit erreichen wir, dass wert_{f_L} zu mindestens einem der beiden Zeitpunkte mit w übereinstimmt. Die Flusswerte zu Zeitpunkten t''' mit $t' \neq t'''$ und $t'' \neq t'''$ haben wir nicht verändert.

Wir bemerken, dass $w_{f_L}^+(t)$ weiterhin zu jedem Zeitpunkt mindestens den Wert $w^+(t)$ hat. Dadurch können wir analog das nächste Paar von Zeitschritten $t' < t''$ mit $\text{wert}_{f_L}(t') > w(t')$ und $\text{wert}_{f_L}(t'') < w(t'')$ finden und f_L so verändern, dass $w_{f_L}^+$ und w für mindestens einen der beiden Zeitpunkte übereinstimmen. Auf diese Weise fahren wir fort, solange

$w_{f_L}^+$ und w nicht vollständig übereinstimmen. Da wir in jedem Schritt mindestens einen Zeitschritt in Übereinstimmung bringen (und sich w^+ nach endlich vielen Zeitschritten nicht mehr ändert), ist dieses Verfahren endlich.

Anschließend verfahren wir mit dem Fluss f_R in N_R analog, so dass wir am Ende beide Flüsse zusammensetzen können und einen Fluss erhalten, der $w^+(t)$ zu jedem Zeitpunkt $t \geq 1$ realisiert.

□

Lemma 48. *Im Beweis von Satz 47 gilt, dass $w(t)$ monoton fallend ist.*

Beweis. Die Werte $\text{wert}_{f_L}(t)$ und $\text{wert}_{f_R}(t)$ sind für steigendes t monoton fallend, da der maximale Fluss in N_L und N_R in einem späteren Zeitpunkt nicht größer werden kann. Das bedeutet, dass die Zunahme von $w_{f_L}^+(t)$ bzw. $w_{f_R}^+(t)$ monoton fallend ist. Anders ausgedrückt: Man kann $w_{f_L}^+(t)$ bzw. $w_{f_R}^+(t)$ jeweils passend (z.B. durch lineare Interpolation) zu einer kontinuierlichen konkaven Funktion erweitern. Das Minimum der beiden konkaven Funktionen ist dann konkav, und es stimmt an den diskreten Stellen $t \geq 1$ mit $w^+(t)$ überein. Also ist die Zunahme von $w^+(t)$ abnehmend, d.h. w ist monoton fallend. □

3.3.2 In-Trees und Out-Trees

In- und Out-Trees besitzen im Allgemeinen kein Earliest Arrival Transshipment, wie man an dem Beispiel aus Abbildung 3.3 sieht. Das Netzwerk in dieser Abbildung ist ein gerichteter Pfad aus drei Kanten und somit sowohl ein In- als auch ein Out-Tree, und es besitzt kein Earliest Arrival Transshipment. Wenn der längste Pfad allerdings maximal aus zwei Kanten besteht, gibt es in einem In- oder Out-Tree immer ein Earliest Arrival Transshipment.

Lemma 49. *Sei $B = (V, E)$ ein Out-Tree mit maximaler Tiefe 2. Außerdem sei $u : E \rightarrow \mathbb{Z}_{\geq 0}$ eine beliebige Kapazitätsfunktion, $p : V \rightarrow \mathbb{Z}_{\geq 0}$ eine beliebige zulässige Potentialfunktion und sei $(S^+ \subseteq V, S^- \subseteq V)$ die zu p konsistente Wahl der Quellen- und Senkenmenge. Dann gibt es in $N_B = (B, u, \tau^0, S^+, S^-)$ ein Earliest Arrival Transshipment.*

Beweis. Zunächst machen wir einige Beobachtungen über die Verteilung von Quellen und Senken innerhalb von V . Wenn der Wurzelknoten keine Quelle ist, so kann in einem ausgleichenden Fluss niemals Fluss durch den Wurzelknoten oder eine der aus dem Wurzelknoten ausgehenden Kanten fließen. Dadurch reduziert sich die Frage, ob der gegebene Out-Tree ein EAT hat, darauf, ob jeder der durch Herausnahme des Wurzelknotens entstehenden

3 Dynamische Transshipments mit Nullfahrzeiten

Teilbäume für sich betrachtet ein EAT hat. Dieses ist aber immer der Fall, da diese Teilbäume nur einen Knoten besitzen, aus dem Fluss herausfließen kann, so dass sie immer maximal eine Quelle haben. Wir gehen also davon aus, dass der Wurzelknoten des Out-Trees positives Potential besitzt. Außerdem kann in die Blätter des Out-Trees nur Fluss hinein- und nicht hinausfließen. Wenn ein Blatt kein echt negatives Potential hat, dann weist jeder ausgleichende Fluss allen Kanten in dieses Blatt denn Flusswert Null zu. Solche Blätter können also von der Betrachtung ausgeschlossen werden. Wir nehmen deshalb an, dass Blätter echt negatives Potential haben.

Entsprechend dieser Beobachtungen benennen wir nun die Knoten im Baum: Den Wurzelknoten nennen wir s . Auf mittlerer Ebene gebe es k Quellenknoten, die wir v_1, \dots, v_k nennen. Weiterhin gebe es dort l Knoten mit Potential Null, die v_{k+1}, \dots, v_{k+l} heißen sollen und m Senken $v_{k+l+1}, \dots, v_{k+l+m}$. Die Anzahl der Kinder eines Knotens v_i (für $1 \leq i \leq k+l+m$) bezeichnen wir mit n_{v_i} , die Kinder selbst, welche ja in jedem Fall Senken sind, nennen wir $t_{v_i,1}, \dots, t_{v_i,n_{v_i}}$.

Es sei $T_i = (V_i, E_i)$ der von $V_i := \{v_i, t_{v_i,1}, \dots, t_{v_i,n_{v_i}}\}$ induzierte Teilgraph von B und es sei $U_i = (V_i \cup \{s\}, E_i \cup \{(s, v_i)\})$. Wir definieren $\text{wert}_f(V')$ für eine Knotenteilmenge $V' \subseteq V$ durch $\text{wert}_f(V') := \sum_{v' \in V' \wedge p(v') < 0} \text{wert}_f(v')$. Dann kann man den Flusswert eines ausgleichenden Flusses f in N_B folgendermaßen schreiben:

$$\begin{aligned} \text{wert}_f &= \sum_{i=1}^{k+l} \sum_{j=1}^{n_{v_i}} \text{wert}_f(t_{v_i,j}) + \sum_{i=k+l+1}^{k+l+m} \left(\text{wert}_f(v_i) + \sum_{j=1}^{n_{v_i}} \text{wert}_f(t_{v_i,j}) \right) \\ &= \sum_{i=1}^{k+l+m} \text{wert}_f(V_i) \end{aligned}$$

Außerdem können sich Flüsse, die in verschiedene Teilbäume fließen, nicht gegenseitig behindern – insbesondere sind also die einzelnen Werte von $\text{wert}_f(V_i)$ für verschiedene i voneinander unabhängig. Der Flusswert ist deshalb genau dann insgesamt maximal, wenn jeder der Werte $\text{wert}_f(V_i)$ maximal ist. Deshalb möchten wir die Existenz eines EATs in B gerne auf die Existenz von EATs in allen U_i zurückführen. Das einzige, was dabei eventuell noch stören könnte, ist die Notwendigkeit, s für jedes U_i ein neues Potential zuzuweisen. Wir dürfen den Flusswert, der aus s in U_i hineinfließt, nicht unnötig einschränken – andererseits dürfen wir nicht mehr Angebot verteilen als s insgesamt besitzt. Zur Lösung überlegen wir uns das Folgende: Um alle Potentiale in einem Teilbaum T_i ausgleichen zu können, muss s nach T_i genau $p_{U_i} := \left| p(v_i) + \sum_{j=1}^{n_{v_i}} p(t_{v_i,j}) \right|$ Flusseinheiten schicken. Außerdem kann kein ausgleichender Fluss (auch keiner, der nicht zu einem Transshipment erweiterbar ist) jemals mehr als p_{U_i} Flusseinheiten von s in den Teilbaum T_i schicken, da diese nicht abfließen können. Die richtige Wahl für das Potential von s in U_i ist also p_{U_i} (diese addieren sich insgesamt zu $p(s)$ auf, wenn p eine zulässige Potentialfunktion ist).

Wir definieren deshalb $p_i : \{V_i \cup s\} \rightarrow \mathbb{Z}_{\geq 0}$ durch $p_i(v) = p(s) \forall v \in V_i$ sowie $p_i(s) = p_{U_i}$. Mit u_i bezeichnen wir die auf $V_i \cup \{s\}$ eingeschränkte Kapazitätsfunktion u . (S_i^+, S_i^-) ist die zu p_i konsistente Wahl von Quellen- und Senkenmenge in U_i . Dann gibt es in N_B genau dann ein Earliest Arrival Transshipment, wenn es in allen gerichteten Netzwerken $N_i := (U_i, u_i, \tau^0, p_i, S_i^+, S_i^-)$ ein Earliest Arrival Transshipment gibt. Abschließend stellen wir fest, dass N_i zur Netzwerkkategorie \mathcal{N}_\bullet gehört und damit laut Satz 47 ein Earliest Arrival Transshipment besitzt.

Für In-Trees folgt die Behauptung jetzt aus Lemma 43. □

3.3.3 Charakterisierung gerichteter Graphen

Zum Abschluss charakterisieren wir gerichtete Netzwerke mit der schönen Eigenschaft, für alle Kapazitäten, Potentiale und Wahlen von Quellen- und Senkenmengen ein Earliest Arrival Transshipment zu besitzen. In- und Out-Trees gehören zu diesen Netzwerken, und Satz 52 wird zeigen, dass das bereits die größte Klasse gerichteter Netzwerke ist, die diese Eigenschaft besitzt.

Aufgrund der Beispiele in den Abbildungen 3.2 und 3.3 wissen wir, dass ein gerichtetes Netzwerk, das immer ein Earliest Arrival Transshipment enthalten soll, die dort abgebildeten Graphen nicht enthalten darf. Das liegt daran, dass wir durch die Wahl der Kapazitäten einen Graphen auf einen beliebigen, nicht notwendigerweise induzierten Teilgraphen einschränken können. Wenn wir einen Graphen auf einen der Beispielgraphen aus Abbildung 3.2 oder 3.3 einschränken können, können wir durch eine passende Wahl der Potentiale und der Kapazitäten im eingeschränkten Graphen eine Situation erzeugen, in der es kein Earliest Arrival Transshipment gibt. Da die beiden Beispielgraphen von so entscheidender Bedeutung sind, führen wir Bezeichnungen für sie ein.

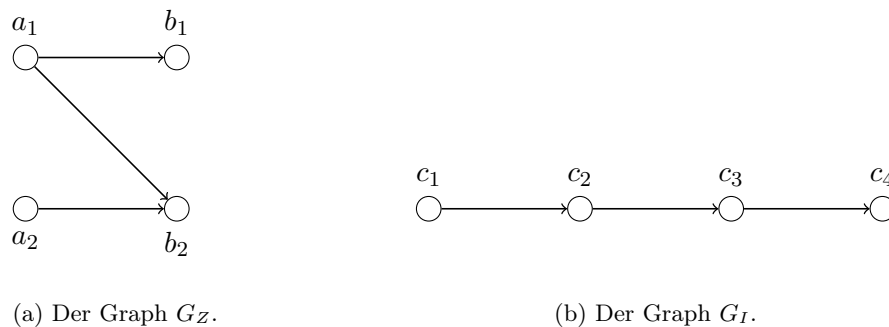


Abbildung 3.5: Die Graphen G_Z und G_I .

3 Dynamische Transshipments mit Nullfahrzeiten

Definition 50. Mit G_Z bezeichnen wir den Graphen mit der Knotenmenge $V = \{a_1, a_2, b_1, b_2\}$ und den Kanten $E = \{(a_1, b_1), (a_1, b_2), (a_2, b_2)\}$ (siehe Abbildung 3.5(a)). Mit G_I bezeichnen wir den Graphen mit der Knotenmenge $V = \{c_1, c_2, c_3, c_4\}$ und den Kanten $E = \{(c_1, c_2), (c_2, c_3), (c_3, c_4)\}$ (siehe Abbildung 3.5(b)).

Jetzt charakterisieren wir gerichtete Netzwerke, die immer ein Earliest Arrival Transshipment besitzen. Das Ausschließen von Unterteilungen der Graphen G_Z und G_I als Teilgraphen genügt dafür bereits.

Definition 51. Sei G ein gerichteter Graph. Eine Unterteilung von G ist ein Graph, der durch beliebig viele Einfügeoperationen eines Knotens auf einer Kante aus G erzeugt werden kann. Eine solche Einfügeoperation ersetzt eine Kante $(v, w) \in E$ durch einen Pfad aus zwei Kanten (v, x) und (x, w) , wobei $x \notin V$ ist. Das Ergebnis einer einzelnen Einfügeoperation ist also ein Graph $G' = (V \cup \{x\}, (E \setminus \{(v, w)\}) \cup \{(v, x), (x, w)\})$.

Satz 52. Sei $N = (G, -, -, -, -, -)$ ein gerichtetes Netzwerk. N besitzt genau dann für alle Wahlen von Kapazitätsfunktion, Potentialen und Quellen- und Senkenmengen ein Earliest Arrival Transshipment, wenn es keine Unterteilung von G_Z oder G_I als Teilgraph enthält.

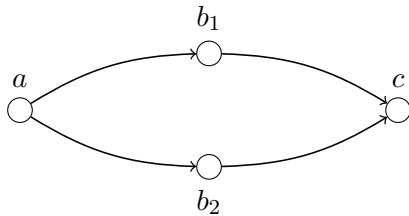
Beweis. Wir gehen davon aus, dass der von G induzierte ungerichtete Graph G_U zusammenhängend ist. Sollte dies nicht der Fall sein, kann man die von den in G_U definierten Zusammenhangskomponenten induzierten Teilgraphen von G einzeln darauf untersuchen, ob sie immer ein Earliest Arrival Transshipment enthalten. Der gesamte Graph enthält genau dann für alle Kapazitäten und Bedarfe ein Earliest Arrival Transshipment, wenn dies für jede der Zusammenhangskomponenten gilt.

i) Sei in G entweder eine Unterteilung von G_Z oder G_I als Teilgraph enthalten. Wir müssen eine Situation konstruieren, in der N kein Earliest Arrival Transshipment enthält. Wir beobachten zunächst, dass wir Unterteilungen von Graphen dabei genauso behandeln können wie die Graphen selbst, da wir abzweigende Kanten an Zwischenknoten mit Hilfe der Kapazitätsfunktion ausschalten können und sich aufeinanderfolgende Kanten mit gleicher Kapazität dann nicht anders verhalten als eine einzelne Kante. Wenn eine Unterteilung von G_Z in G enthalten ist, so setzen wir die Kapazitäten für alle Kanten der Unterteilung auf Eins und wählen die Potentiale entsprechend der in Abbildung 3.2 angegebenen Vorgehensweise. Wenn eine Unterteilung von G_I in G enthalten ist, dann ist G_I selbst Teilgraph von G . Wir wählen dann Kapazitäten so, dass die mittlere Kante des enthaltenen G_I eine kleinere Kapazität hat als die beiden äußeren, und vergeben Potentiale entsprechend der Erklärung in Abbildung 3.3. In beiden Fällen gibt es kein Earliest Arrival Transshipment.

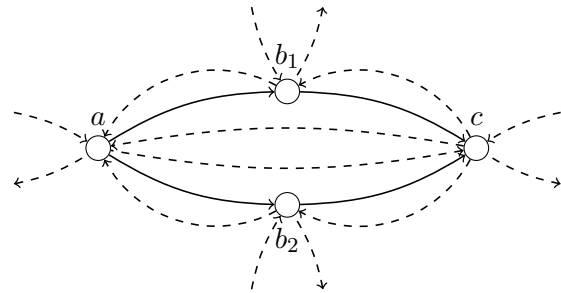
ii) Jetzt ist ein Graph G gegeben, der keine Unterteilung der Graphen G_Z und G_I enthält. Zu zeigen ist, dass N ein Earliest Arrival Transshipment enthält. Da G_I ein gerichteter Pfad aus drei Kanten ist, kann G nur Pfade aus maximal zwei Kanten enthalten. G ist also bereits stark eingeschränkt. Im Folgenden werden wir zeigen, dass G entweder sehr klein ist oder eine schöne Struktur aufweist. Dazu gehen wir verschiedene Fälle durch.

Fall 1 Zunächst behandeln wir den Fall, dass G zwei parallele Pfade zwischen zwei Knoten $a \in V$ und $b \in V$ enthält, d.h. es gibt Knoten $c_1 \in V$ und $c_2 \in V$ mit $(a, c_1), (a, c_2), (c_1, b), (c_2, b) \in E$. Diese Situation wird in Abbildung 3.6(a) gezeigt. Die Einschränkung, auf Unterteilungen von G_Z und G_I zu verzichten, stellt sich in diesem Fall als sehr mächtig heraus: Es ist nicht möglich, dass G noch weitere Kanten enthält (somit auch keine weiteren Knoten, da wir von einem zusammenhängenden Graphen ausgehen). Jede mögliche weitere Kante, die entweder zu einem vorhandenen Knoten zeigt oder von einem solchen ausgeht, formt mit den vorhandenen Kanten entweder eine Unterteilung von G_Z oder von G_I . Die einzelnen Möglichkeiten sind in Abbildung 3.6(b) bis 3.6(d) dokumentiert. G kann den Teilgraphen in Abbildung 3.6(a) also nur enthalten, wenn er mit ihm übereinstimmt. Wenn es in diesem Graphen keine zwei Quellen und Senken gibt, enthält er immer ein Earliest Arrival Transshipment. Außerdem ist die Existenz gesichert, wenn a keine Quelle oder c keine Senke ist. Wir gehen o.B.d.A. davon aus, dass b_1 eine Quelle und b_2 eine Senke ist und konstruieren unter dieser Bedingung ein Earliest Arrival Transshipment: Solange dies möglich ist, lasten wir die Kanten (a, b_1) und (b_2, c) voll aus. Damit ist der Flusswert maximal, unabhängig von dem Fluss auf den anderen beiden Kanten. Auch auf (b, c_1) und (a, b_2) versenden wir aber den maximal möglichen Flusswert, um sicherzustellen, dass die Potentiale von b_1 und b_2 so spät wie möglich ausgeglichen werden. Ab dem Zeitpunkt, zu dem dieses für b_1 oder b_2 geschieht (falls es überhaupt eintritt), versenden wir so viel wie möglich über den Pfad aus den Kanten (a, b_1) und (b_1, c) bzw. (a, b_2) und (b_2, c) . Unser Flusswert bleibt damit auch für die späteren Zeitpunkte optimal. In dem Graphen in Abbildung 3.6(a) gibt es also immer ein Earliest Arrival Transshipment.

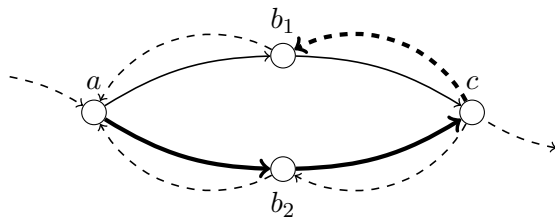
Eine leichte Abwandlung der Situation erhalten wir, wenn nur einer der beiden Pfade vorhanden ist, zusätzlich aber von a direkt eine Kante nach c verläuft. Wie in Abbildung 3.7 dokumentiert wird, sind hier zwar zusätzliche Kanten möglich, aber nur zwischen den drei bereits vorhandenen Knoten - eine Kante zu einem neuen Knoten führt zu einer Unterteilung von G_I oder G_Z . Enthält G also einen Pfad aus zwei Kanten und eine parallele Kante, so besitzt G nur drei Knoten und damit immer ein Earliest Arrival Transshipment (da es maximal eine Quelle oder maximal eine Senke gibt).



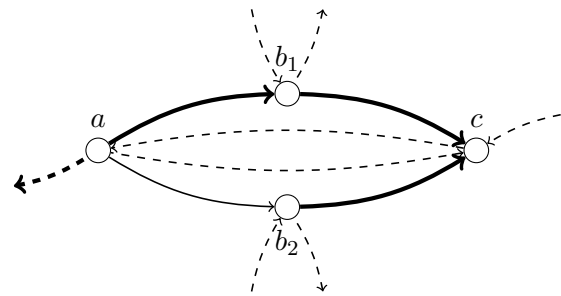
(a) Zwei parallele Pfade mit jeweils zwei Kanten.



(b) Die gestrichelten Kanten stellen alle Möglichkeiten dar, wie an a , b_1 , b_2 oder c weitere Kanten anliegen könnten.

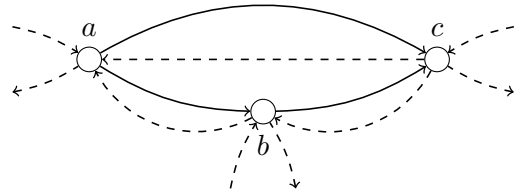
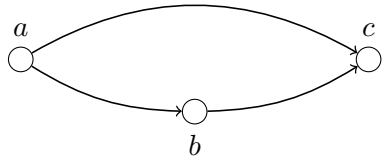


(c) Keine der hier eingezeichneten Kanten kann in G enthalten sein, weil dadurch eine Unterteilung von G_I entsteht. Diese ist beispielhaft für eine der gestrichelten Kanten durch dick gezeichnete Kanten markiert.

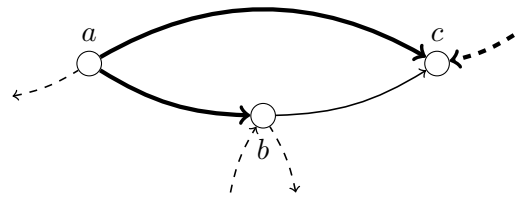
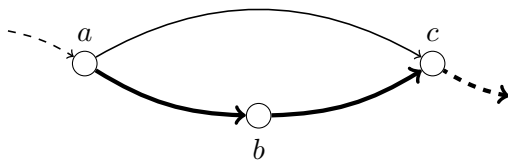


(d) Keine der hier eingezeichneten Kanten kann in G enthalten sein, weil dadurch eine Unterteilung von G_Z entsteht. Diese ist beispielhaft für eine der gestrichelten Kanten durch dick gezeichnete Kanten markiert.

Abbildung 3.6: Wenn G zwei parallele Pfade aus je zwei Kanten enthält, dann können keine weiteren Kanten enthalten sein, da sonst eine Unterteilung von G_I oder G_Z entsteht. Abbildung (b) zeigt alle Möglichkeiten für zusätzliche Kanten auf, die an einem der vorhandenen Knoten anliegen. Diese Möglichkeiten unterscheiden sich darin, ob durch sie eine Unterteilung von G_I oder von G_Z entsteht. Erstgenannte sind in Abbildung (c) abgebildet, die anderen in Abbildung (d).



- (a) Ein Pfad mit zwei Kanten und eine parallel verlaufende Kante. (b) Die gestrichelten Kanten stellen alle Möglichkeiten dar, wie an a , b , oder c weitere Kanten anliegen könnten.



- (c) Die hier eingezeichneten Kanten können nicht in G enthalten sein, da sonst ein G_I enthalten ist. Dieser ist für eine der gestrichelten Kanten beispielhaft durch dick gezeichnete Kanten markiert. (d) Die hier eingezeichneten Kanten können nicht in G enthalten sein, da sonst ein G_Z enthalten ist. Dieser ist für eine der gestrichelten Kanten beispielhaft durch dick gezeichnete Kanten markiert.

Abbildung 3.7: Wenn G einen Pfad aus zwei Kanten und eine parallele Kante enthält, können keine weiteren Knoten vorhanden sein, da sonst eine Unterteilung von G_I oder G_Z entsteht. Im Gegensatz zu dem in Abbildung 3.6 gezeigten Fall können zwar zusätzliche Kanten enthalten sein (und zwar (b, a) , (c, a) und (c, b)), aber keine der Kanten, die zu zusätzlichen Knoten führen. Abbildung (c) zeigt die Kanten, durch die eine Unterteilung des G_I entsteht, Abbildung (d) zeigt jene, durch die eine Unterteilung von G_Z entsteht.

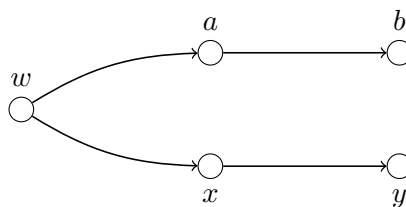
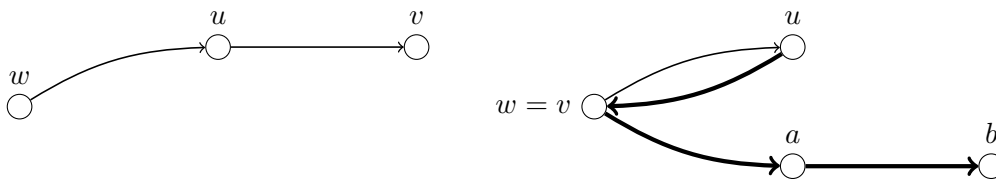
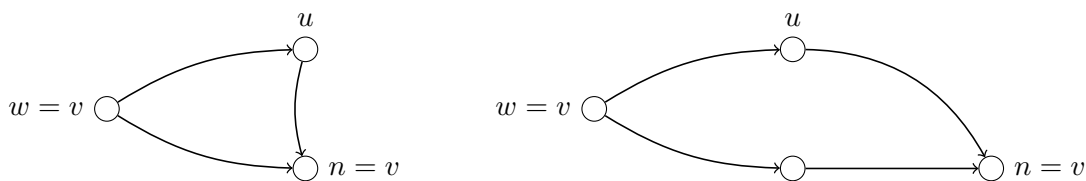


Abbildung 3.8: Zwei Pfade aus zwei Kanten, die im gleichen Knoten starten (aber nicht enden).

3 Dynamische Transshipments mit Nullfahrzeiten

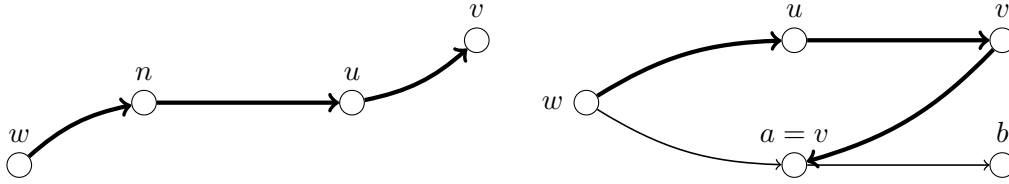


- (a) Wenn v nicht in V_B liegt, wird die Out-Tree-Eigenschaft nicht gestört, außerdem ist die Tiefe weiterhin maximal 2. B wäre also nicht maximal gewesen.
- (b) Wenn $w = v$ gilt, dann bilden (u, w) , (w, a) und (a, b) oder (u, w) , (w, x) und (x, y) einen G_I . Ein-gezeichnet ist der erste Fall.

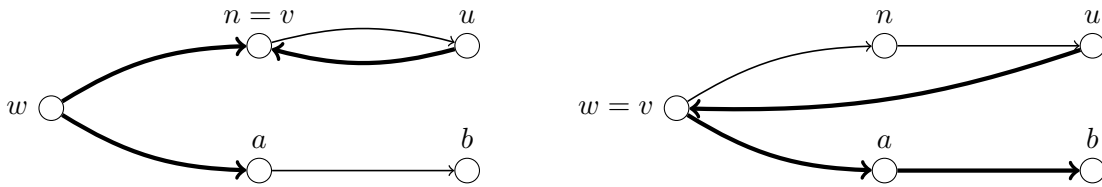


- (c) Wenn v ein Knoten n in V_B , aber nicht w ist, gelangen wir in Fall 1. Hier ist die Situation eingezeichnet, wenn w und n direkt verbunden sind.
- (d) Wenn v ein Knoten n in V_B , aber nicht w ist, gelangen wir in Fall 1. Hier ist die Situation eingezeichnet, wenn w und n über einen weiteren Knoten verbunden sind.

Abbildung 3.9: Diese vier Abbildungen zeigen die verschiedenen Fälle, die auftreten können, wenn B die Kante (w, u) enthält und wir eine Kante (u, v) hinzufügen möchten.



(a) Wenn v nicht in V_B liegt, bilden w, n, u und v einen G_I . (b) Wenn v in V_B liegt, aber weder w noch n ist, bilden w, n, u und v ebenfalls einen G_I .



(c) Wenn v der Knoten n ist, erhalten wir einen G_Z , bestehend aus den Kanten $(u, n), (w, n)$ und (w, a) (falls $v = n = a$ gilt, ersetze (w, a) durch (w, x)). (d) Wenn $w = v$ gilt, dann bilden $(u, w), (w, a)$ und (a, b) oder $(u, w), (w, x)$ und (x, y) einen G_I . Ein-gezeichnet ist der erste Fall.

Abbildung 3.10: Diese vier Abbildungen zeigen die verschiedenen Fälle, die auftreten können, wenn B die Kanten (w, n) und (n, u) enthält und wir eine Kante (u, v) hinzufügen möchten.

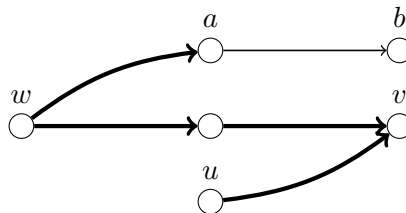


Abbildung 3.11: Diese Abbildung zeigt einen entstehenden G_Z , wenn $v \in V_B, u \notin V_B$ und $v = b$ gilt.

Fall 2 Der nächste Fall behandelt die Situation, dass G zwei Pfade aus zwei Kanten besitzt, die im gleichen Knoten starten (aber nicht im gleichen Knoten enden). Im Gegensatz zu Fall 1 kann G nun beliebig viele weitere Kanten besitzen, sofern diese „passend“ liegen. Wir werden aber zeigen, dass G ein Out-Tree mit maximaler Tiefe 2 sein muss. Damit besitzt G aufgrund von Lemma 49 für alle Potentiale, Kapazitäten und Wahlen von Quellen- und Senkenmenge ein Earliest Arrival Transshipment. Wir benennen die Knoten unserer beiden Pfade wie in Abbildung 3.8.

Zuerst bemerken wir, dass G keine Kanten enthalten kann, die in w enden – eine solche Kante bildet immer einen G_I mit einem der beiden Pfade (auch, wenn sie z.B. in y beginnt, in diesem Fall wird der G_I von y, w, a und b gebildet). Wir wollen zeigen, dass G ein Out-Tree ist, und wissen jetzt, dass dann w die Wurzel unseres Baums sein muss.

Sei nun $B = (V_B, E_B) \subseteq G$ ein Teilgraph von G mit den folgenden Eigenschaften: B ist ein Out-Tree mit Wurzel w , der insofern maximal ist, als dass für jede Kante $e = (u, v) \in E \setminus E_B$ mit $u \in V_B$ oder $v \in V_B$ (oder beides) gilt, dass $B' = (V_B \cup \{u, v\}, E_B \cup \{e\})$ kein Out-Tree ist. Man kann zu B also keine Kante hinzufügen, ohne die Out-Tree-Eigenschaft zu zerstören. Wir zeigen, dass es keine Kante $e = (u, v) \in E \setminus E_B$ geben kann, d.h. B ist G . Sei dafür eine solche Kante $e = (u, v) \in E \setminus E_B$ gegeben. Dann ist entweder $u \in V_B$ oder $v \in V_B$ (oder beides).

Zuerst betrachten wir, was geschieht, wenn $u \in V_B$ ist. Da B ein Out-Tree mit Wurzel w ist und $u \in V_B$ gilt, gibt es in B einen Weg von w nach u . Dieser Weg kann aus ein oder zwei Kanten bestehen (da G keine Unterteilung von G_I enthält, kann es keine längeren Pfade geben). Für beide Möglichkeiten müssen wir untersuchen, wie die Situation aussieht. Wenn der Weg aus einer Kante besteht, handelt es sich also um die Kante (w, u) . Wir stoßen auf vier (unmögliche) Möglichkeiten, wie v liegen kann, die wir im Folgenden kurz nennen und die auch durch Abbildung 3.9 veranschaulicht werden. Es ist nicht möglich, dass v außerhalb von V_B liegt, weil $B' = (V_B \cup \{v\}, E_B \cup \{(u, v)\})$ sonst ein Out-Tree mit maximaler Tiefe 2 ist, B wäre also nicht maximal. Es kann auch nicht sein, dass v der Wurzelknoten w ist (d.h. die Kante zeigt auf die Wurzel zurück), weil (v, w) entweder zusammen mit (w, a) , (a, b) oder mit (w, x) , (x, y) einen G_I bilden würde (v könnte zwar a oder x sein, aber nicht beides gleichzeitig). Es bleibt noch, dass v auf einen anderen Knoten $n \neq w$ in V_B zeigt. Dann bilden w, n und v aber einen der beiden Graphen aus Fall 1 (welchen hängt davon ab, ob w und n direkt oder über einen Zwischenknoten verbunden sind). Eine solche Konstruktion schließt aber, wie in Fall 1 bereits gezeigt, weitere Knoten aus – insbesondere kann sie also gar nicht gleichzeitig mit den Pfaden auftreten, von denen wir im aktuellen Fall ausgehen.

Jetzt betrachten wir die drei Fälle, die entstehen, wenn der Weg von w nach u über einen Zwischenknoten n verläuft. Diese werden in Abbildung 3.10 veranschaulicht. Wenn v nicht

einer der Knoten w und n ist, dann erhalten wir durch (w, n) , (n, u) und (u, v) einen G_I . Falls v gleich n ist, können wir einen G_Z entdecken: Er besteht aus den Kanten (u, n) , (w, n) und (w, a) , falls $n \neq a$, oder aus den Kanten (u, n) , (w, n) und (w, x) andernfalls. Wenn v der Wurzelknoten w ist, und u nicht b ist, dann bilden (u, w) , (w, a) und (a, b) einen G_I , wenn $b = u$ gilt, dann erhalten wir den G_I durch die Kanten (u, w) , (w, x) und (a, y) . Damit ist der Fall, dass $u \in V_B$ liegt, abgeschlossen. Wir halten außerdem fest, dass insbesondere der Fall, dass u und v beide in V_B liegen, bereits abgehandelt ist.

Es bleibt also der Fall, dass $v \in V_B$ gilt, aber $u \notin V_B$ ist. Da B ein Out-Tree mit Wurzel w ist, gibt es einen Pfad p von w nach v . Außerdem gibt es (mindestens) eine ausgehende Kante e aus w , deren Zielknoten nicht auf p liegt. Dann bilden aber p , e und (u, v) einen G_Z . Dieses ist beispielhaft in Abbildung 3.11 zu sehen.

Wir haben nun gezeigt, dass es keine Kante außerhalb von E_B geben kann – B umfasst also ganz G , und G ist somit ein Out-Tree mit maximaler Tiefe 2.

Fall 3 Der nächste Fall dreht sich um die Situation, dass G zwei Pfade aus zwei Kanten besitzt, die im gleichen Knoten enden (aber nicht im gleichen Knoten starten). Dafür wenden wir Lemma 43 an, d.h. wir führen die Existenz von Earliest Arrival Transshipments in G auf die Existenz von Earliest Arrival Transshipments in \overleftarrow{G} zurück (für \overleftarrow{G} wie in Lemma 43 definiert). In \overleftarrow{G} gibt es aber zwei Pfade, die im gleichen Knoten beginnen (aber in verschiedenen enden), so dass wir aufgrund von Fall 2 wissen, dass immer ein Earliest Arrival Transshipment existiert.

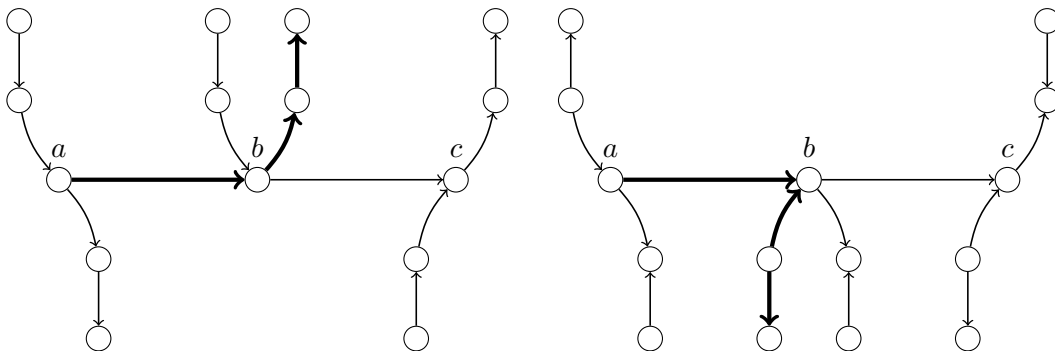
Fall 4 Wir haben uns bisher mit Fällen beschäftigt, in denen zwei Pfade aus zwei Kanten in G enthalten sind. Die restlichen Fälle enthalten entweder Pfade aus zwei Kanten, die nicht auf die bisher betrachteten Weisen zusammenhängen – oder sie enthalten gar keine Pfade aus zwei Kanten. Letzteren Fall behandeln wir hier. Sei G also ein Graph, der keinen Pfad aus zwei Kanten enthält (und außerdem keinen G_Z). Sei $v \in V$ ein Knoten mit mindestens zwei ausgehenden Kanten. v kann keine eingehenden Kanten besitzen (sonst gibt es einen Pfad der Länge 2). Die Endknoten der aus v ausgehenden Kanten können dann selbst weder eine ausgehende Kante enthalten (sonst gibt es einen Pfad der Länge 2) noch eine eingehende Kante (diese bildet zusammen mit der Kante von v aus und der anderen von v ausgehenden Kante einen G_Z). G enthält also nur Kanten von v zu anderen Knoten – und damit immer ein Earliest Arrival Transshipment, da v die einzige mögliche Quelle ist. Wenn kein Knoten mit mindestens zwei ausgehenden Kanten vorhanden ist, aber ein Knoten w mit zwei eingehenden Kanten, so kann man analog argumentieren, dass G nur Kanten besitzt, die in w enden. In diesem Fall gibt es immer ein Earliest

3 Dynamische Transshipments mit Nullfahrzeiten

Arrival Transshipment, weil es maximal eine Senke gibt. Gibt es weder einen Knoten mit zwei eingehenden noch einen mit zwei ausgehenden Kanten, dann besteht G entweder aus einem einzelnen Knoten, einer einzelnen Kante oder aus einer Kante zusammen mit ihrer Rückwärtskante.

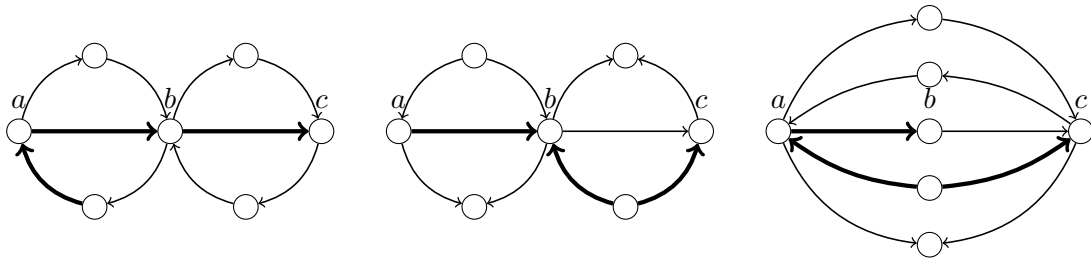
Fall 5 Im abschließenden Fall betrachten wir die Situation, dass G mindestens einen Pfad der Länge 2 besitzt (wir befinden uns also nicht in Fall 4), aber keine der in den Fällen 1-3 beschriebenen Situationen auftritt. Wir werden zeigen, dass die verbleibende Klasse von Graphen ebenfalls so weit eingeschränkt ist, dass es immer ein Earliest Arrival Transshipment gibt. Der Pfad soll aus den Kanten (a, b) und (b, c) bestehen.

Dafür grenzen wir zuerst die noch möglichen Kanten ein. Sei G_U der von G induzierte ungerichtete Graph. Behauptung: G kann keine Knoten (außer a und c) besitzen, die in G_U zu einem der Knoten a, b oder c den Abstand 2 haben, weil sonst ein G_I , ein G_Z oder einer der bereits behandelten Fälle auftritt. Für die Kombination der Ausrichtung der beiden



- (a) In dieser Abbildung werden die beiden ungerichteten Kanten mit der gleichen Ausrichtung versehen. Auf diese Weise erhalten wir sechs Fälle. Für die oberhalb von a, b und c eingezeichneten Pfade entsteht ein G_I , ein solcher ist beispielhaft eingezeichnet. Der linke der beiden unteren Pfade führt zu der Situation aus Fall 2, der rechte zu der Situation aus Fall 3.
- (b) Hier werden alle Fälle abgebildet, in denen die beiden neuen Kanten mit entgegengesetzter Ausrichtung versehen werden und so voneinander weg oder aufeinander zu zeigen. In den oberen zwei Fällen tritt wiederum ein G_I auf, in den unteren vier Fällen bildet sich ein G_Z . Ein solcher Fall ist beispielhaft durch dick gezeichnete Linien markiert.

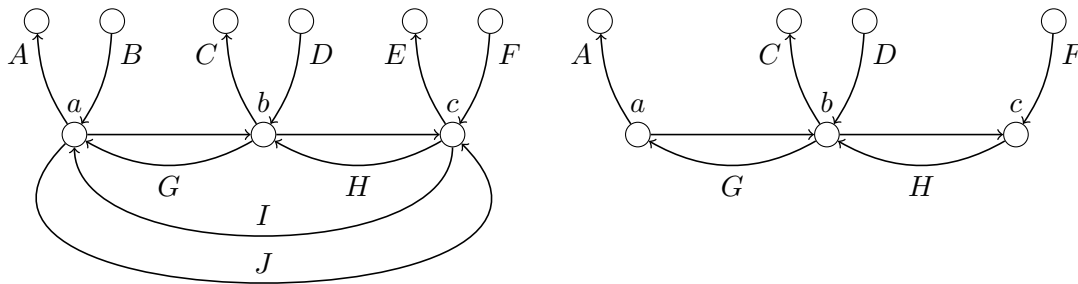
Abbildung 3.12: Hier werden die zwölf Möglichkeiten dokumentiert, einen Pfad aus zwei ungerichteten Kanten an a, b oder c anzuhängen und anschließend mit zwei Richtungen für die beiden Kanten zu versehen. In jedem der zwölf Fälle tritt ein G_I , ein G_Z oder einer der bereits behandelten Fälle 1-3 auf.



- (a) Diese Abbildung deckt die Fälle ab, bei denen a und b oder b und c durch einen ungerichteten Pfad verbunden werden, dessen Kanten anschließend in die gleiche Richtung ausgerichtet werden. Jeder dieser Pfade führt zu einem G_I , einer ist als Beispiel eingezeichnet.
- (b) In dieser Abbildung werden alle Möglichkeiten aufgezeigt, einen ungerichteten Pfad zwischen zwei der Knoten a , b und oder zwischen b und c einzufügen und die Kanten des Pfades verschieden auszurichten. Dabei entsteht für die beiden oben eingezeichneten Pfade ein G_I , bei den unteren ein G_Z . Einer der beiden G_Z ist exemplarisch durch dick gezeichnete Linien markiert.
- (c) In der letzten Abbildung werden die vier Fälle, a und c mit einem Pfad zu verbinden und die Kanten des Pfades auf alle vier möglichen Weisen auszurichten, abgebildet. Dabei erzeugen die oberen Pfade wiederum ein G_I und die unteren ein G_Z , und einer der beiden G_Z ist eingezeichnet.

Abbildung 3.13: Hier werden die zwölf Möglichkeiten dokumentiert, einen Pfad aus zwei ungerichteten Kanten zwischen zwei der Knoten a , b oder c verlaufen zu lassen und anschließend mit zwei Richtungen für die beiden Kanten zu versehen. In jedem der zwölf Fälle tritt ein G_I , ein G_Z oder einer der bereits behandelten Fälle 1-3 auf.

3 Dynamische Transshipments mit Nullfahrzeiten



(a) Alle Möglichkeiten für zusätzliche Kanten.

(b) Die verbleibenden Möglichkeiten.

Abbildung 3.14: Abbildung (a) zeigt alle Möglichkeiten, zusätzliche Kanten an den Knoten a , b oder c anzubringen, durchnummeriert von A bis J . Dabei bilden B und E einen G_I , bei J liegt eine der Situationen aus Fall 1 vor. Außerdem gibt es einen G_I , sobald die Kante I mit einer der Kanten vom Typ A , C , D oder F gemeinsam auftritt. Dadurch kann I ebenfalls aus der Betrachtung ausgeschlossen werden. Abbildung (b) zeigt die verbleibenden Möglichkeiten.

Kanten, die einen solchen Knoten mit a , b oder c verbinden, gibt es vier Möglichkeiten. Aufgrund der drei Anschlussmöglichkeiten gibt es daher zwölf Fälle, die überprüft werden müssen. Dieses geschieht in Abbildung 3.12. Weitere zwölf Fälle erhält man, wenn man (ungerichtete) Pfade aus zwei Kanten zwischen zwei der drei Knoten a , b und c betrachtet und diese auf jede mögliche Weise ausrichtet. In Abbildung 3.13 wird dokumentiert, dass wir in diesen Fällen immer einen G_I oder einen G_Z erhalten.

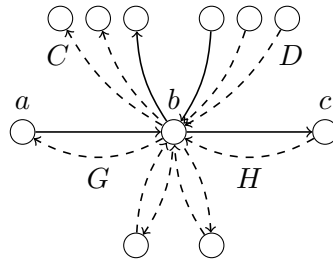
Dadurch wissen wir nun folgendes: Zusätzliche Kanten in G können nur an a , b oder c anschließen, und wenn zwei Kanten dabei an verschiedenen Knoten (aus a , b , c) anliegen, so sind die jeweils anderen Knoten der Kante verschieden (sonst ergibt sich, ungerichtet aufgefasst, einen Pfad zwischen den beiden Startknoten).

Um den Fall abzuschließen, testen wir nun systematisch, welche Typen von Kanten es gibt und welche Typen mit welchen Typen gemeinsam auftreten können. Dafür listen wir alle Möglichkeiten für zusätzliche Kanten in G in Abbildung 3.14(a) auf. Die Kanten A bis F stehen dabei für Kantentypen und können auch mehrfach vorkommen, die Kanten G , H und I können maximal einmal auftreten.

Man sieht, dass die Kanten B , E und J nicht auftreten können, sie sind nur zur besseren Übersicht eingezeichnet. Außerdem stellen wir fest, dass die Kante I nur vorkommen kann, wenn keine Kante vom Typ A , C , D oder F auftritt. Das bedeutet aber, dass G auf die drei Knoten a , b und c beschränkt ist und auf jeden Fall ein Earliest Arrival Transshipment



- (a) Wenn G außer (a, b) und (b, c) mindestens eine Kante vom Typ A enthält, so sind Kanten vom Typ D und F sowie die Kante H nicht mehr möglich und wir erhalten einen Graph, der aus der Kante (a, b) , evtl. der Kante G , mindestens einer A -Kante und mindestens einer C -Kante besteht (dafür fassen wir (b, c) als C -Kante auf).
- (b) Wenn G außer (a, b) und (b, c) mindestens eine Kante vom Typ F enthält, so sind Kanten vom Typ A und C sowie die Kante G nicht mehr möglich und wir erhalten einen Graph, der aus der Kante (b, c) , evtl. der Kante H , mindestens einer F -Kante und mindestens einer D -Kante besteht (dafür fassen wir (a, b) als D -Kante auf).



- (c) Wenn G außer (a, b) und (b, c) weder Kanten vom Typ A noch Kanten vom Typ F enthält, so handelt es sich um einen sternförmigen Graphen, der aus den vom Typ C und D besteht (alle vorhandenen und eventuell vorhandenen Kanten können als C - oder D -Kante aufgefasst werden). Es ist auch möglich, dass eine C -Kante den Startknoten einer D -Kante als Endknoten hat, so dass es Knoten gibt, zwischen denen sowohl Hin- als auch Rückkante existieren.

Abbildung 3.15: Wenn G den Pfad aus den beiden Kanten (a, b) und (b, c) , aber keinen G_I und keinen G_Z enthält und wenn keiner der Fälle 1-3 vorliegt, dann gibt es nur noch drei Graphenklassen, zu denen G gehören kann. Diese sind hier abgebildet.

3 Dynamische Transshipments mit Nullfahrzeiten

besitzt. In der weiteren Betrachtung können wir I nun ebenfalls ausschließen und verbleiben mit den Möglichkeiten aus Abbildung 3.14(b).

Wir stellen fest, dass Kanten vom Typ A nicht mit Kanten vom Typ D oder F oder mit der Kante H zusammen auftreten können, da sonst ein G_Z entsteht. Daher gelangen wir in eine Situation wie in Abbildung 3.15(a), wenn mindestens eine Kante vom Typ A vorhanden ist. Andererseits stellen wir fest, dass Kanten vom Typ F Kanten vom Typ A und C sowie die Kante G ausschließen, so dass das Vorhandensein von mindestens einer Kante vom Typ F eine Situation wie in Abbildung 3.15(b) herbeiführt. Der dritte Fall, der abschließend noch behandelt werden muss, entsteht, wenn weder Kanten vom Typ A noch vom Typ F vorhanden sind, so dass es nur Kanten vom Typ C und D und eventuell die Kanten G und H gibt. In diesem Fall, der in Abbildung 3.15(c) abgebildet ist, kann es auch Knotenpaare geben, zwischen denen Hin- und Rückkante existieren.

Den dritten Fall können wir am schnellsten erledigen: Hier handelt es sich um einen sternförmigen Graphen, der unabhängig von der Verteilung der Quellen und Senken zur Netzwerkkategorie \mathcal{N}_\bullet gehört. Damit hat G in diesem Fall aufgrund von Satz 47 immer ein Earliest Arrival Transshipment.

Wenn wir zeigen, dass es in der Situation aus Abbildung 3.15(a) immer ein Earliest Arrival Transshipment gibt, folgt dieses auch für die Situation aus Abbildung 3.15(b), da man diese auf das Szenario mit A - und C -Kanten zurückführen kann, indem man alle Kanten umdreht und anschließend Lemma 43 verwendet. Wir müssen also nur noch zeigen, dass es immer ein Earliest Arrival Transshipment gibt, wenn G aus der Kante (a, b) , evtl. der Kante (b, a) sowie mindestens einer A - und mindestens einer C -Kante besteht.

Wenn G diese Form hat, können nur a und b sinnvollerweise Quellen sein, da alle anderen Knoten keine ausgehenden Kanten besitzen. Wir gehen davon aus, dass beide eine Quelle sind (ansonsten gibt es auf jeden Fall ein Earliest Arrival Transshipment). Alle anderen Knoten sind Senken oder werden aus der Betrachtung ausgeschlossen. Es handelt sich bei G fast um einen Outtree: Nur die evtl. vorhandene Kante (b, a) stört. Wir beobachten aber, dass es immer (abhängig von den Potentialen) für eine der beiden Kanten (a, b) und (b, a) unsinnig ist, Fluss über diese zu verschicken. Dafür sei S_a die Menge der Senken, die direkt von a aus erreichbar sind, und S_b die Menge der Senken, die direkt von b aus erreichbar sind. Außerdem sei w_a die Summe der Potentiale der Senken in S_a und w_b die Summe der Potentiale der Senken in S_b . Wir wissen, dass $p(a) + p(b) + w_a + w_b = 0$ ist, weil der Potentialvektor sonst nicht zulässig wäre. Nehmen wir kurz an, $p(a)$ sei mindestens w_a . Fluss, der nach S_a fließt, muss immer über a laufen. Aus a können aber nach S_a nur w_a Flusseinheiten verschickt werden. Das zusätzliche Angebot $p(a) - w_a \geq 0$ muss den Knoten in einem ausgleichenden Fluss über die Kante (a, b) verlassen, wenn es überhaupt verschickt

wird. Ebenso muss für jede Flusseinheit, die a über die Kante (b, a) zusätzlich erreicht, eine Flusseinheit über (a, b) zurückgeschickt werden, wenn diese noch verschickt werden soll. Stattdessen kann in dem entsprechenden ausgleichenden Fluss aber auch auf das Versenden über (b, a) und das Rücksenden über (a, b) vollständig verzichtet werden, so dass man immer zu einem anderen ausgleichenden Fluss gelangt, der die Kante (b, a) überhaupt nicht nutzt. Wenn $p(a) \geq w_a$ gilt, können wir also auf die Betrachtung von (b, a) verzichten und erhalten einen Out-Tree, in dem immer ein Earliest Arrival Transshipment existiert. Analog können wir auf die Betrachtung der Kante (a, b) verzichten, wenn $p(b) \geq w_b$ gilt und erhalten für diesen Fall ebenfalls einen Out-Tree und damit immer ein Earliest Arrival Transshipment. Aufgrund von $p(a) + p(b) + w_a + w_b = 0$ ist immer einer der beiden Fälle gegeben, so dass die Situation in Abbildung 3.15(a) nun vollständig abgehandelt ist.

Dieses beendet unseren Beweis: Wir haben gesehen, dass G immer entweder nur aus wenigen Knoten besteht oder ein sternförmiger Graph oder ein Out- / In-Tree (mit einer potentiellen zusätzlichen Kante) ist. Insbesondere besitzt G immer ein Earliest Arrival Transshipment. \square

3 *Dynamische Transshipments mit Nullfahrzeiten*

4 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde die Existenz von Earliest Arrival Flüssen in Netzwerken mit mehreren Senken untersucht. In dieser Forschungsrichtung gibt es noch viele offene Fragen und interessante Problemstellungen, die hier in Zusammenhang mit den in der Arbeit vorgestellten Ergebnissen vorgestellt werden.

Für den Fall allgemeiner Fahrzeiten wurde in Abschnitt 2.4 ein einfacher Existenztest entwickelt, der gegebenenfalls auch einen Earliest Arrival berechnet. Der Test basiert auf der Lösung eines pseudopolynomiell großen linearen Programms (P) und ist damit nicht effizient und auch nicht praxisrelevant. Die Ursache der möglicherweise exponentiellen Größe des linearen Programms liegt in der Betrachtung aller einzelnen Zeitschritte durch das zeitexpandierte Netzwerk. Da sich der Wert, den ein Earliest Arrival Fluss in einem Netzwerk erreichen muss, nicht zwangsläufig in jedem Zeitschritt ändert, ist diese Betrachtung unnötig genau. Möglicherweise lässt sich ein deutlich effizienterer Test entwickeln, wenn man die Earliest Arrival Eigenschaft für mehrere Zeitschritte gleichzeitig überprüft.

Auch die Untersuchung des *dualen* linearen Programms (D) zu (P) ist vielversprechend. Das lineare Programm (P) hat genau dann keine Lösung, wenn (D) keine Lösung hat oder unbeschränkt ist. Stellt man (D) zu dem linearen Programm aus unserem Existenztest auf, stellt man fest, dass es immer die Nulllösung als zulässige Lösung besitzt. Das bedeutet, dass (P) genau dann keine Lösung besitzt, wenn (D) keine Lösung besitzt. Diese Erkenntnis könnte dazu beitragen, mit Hilfe des dualen Programms einen anderen Existenztest zu finden. Ein dritter Ansatz zur Entwicklung eines Existenztests für Earliest Arrival Flüsse besteht darin, sich die Vektoren aus den Flusswerten der Quellen und Senken von ausgleichenden Flüsse mit verschiedenem Zeithorizont anzusehen und das aus diesen Vektoren bestehende Polytop zu analysieren.

Nach dem Existenztest für Netzwerke mit allgemeinen Fahrzeiten wurden in Kapitel 3 Netzwerke mit Nullfahrzeiten untersucht. Dabei wurde eine Klasse von Netzwerken \mathcal{N}_\bullet präsentiert, für die die Existenz von Earliest Arrival Flüssen garantiert ist. Der Beweis für diese Aussage liefert auch einen Algorithmus, um aus zwei Earliest Arrival Flüssen in kleineren Netzwerken mit nur einer Quelle oder Senke einen Earliest Arrival Fluss im gesamten Netzwerk zu berechnen. Die Berechnung der Earliest Arrival Flüsse in den kleineren Netz-

4 Zusammenfassung und Ausblick

werken ist polynomiell möglich, die Verknüpfung zu einem Fluss im Gesamtnetzwerk ist jedoch pseudopolynomiell. Ist es möglich, die Vorgehensweise so zu verbessern, dass der gesamte Algorithmus effizient ist?

Außerdem ist es natürlich sehr interessant, welche weiteren Klassen von Netzwerken die Existenz von Earliest Arrival Flüssen immer garantieren. Dabei kann man danach unterscheiden, welche Informationen bereits gegeben sind und welche beliebig wählbar sein sollen. Gerichtete Netzwerke, die trotz der beliebige Wahl aller Parameter immer einen Earliest Arrival Fluss zulassen, wurden in Abschnitt 3.3.3 charakterisiert. In diese Klasse fallen zum Beispiel In- und Out-Trees mit maximaler Tiefe 2. Die Netzwerkklassse \mathcal{N}_\bullet gehört zu den Netzwerken, die bei fester Quellen- und Senkenmenge für beliebige Wahl der übrigen Parameter die Existenz eines Earliest Arrival Flusses garantieren. Die weitere Suche nach Netzwerkklassen dieser Art könnte möglicherweise zu einer Klassifizierung von Netzwerken mit vorgegebenen Quellen- und Senkenmengen, die immer einen Earliest Arrival Fluss besitzen, führen. Die vollständige Charakterisierung von Netzwerken mit einer Garantie für Earliest Arrival Flüsse wird noch schwieriger, wenn man zum Netzwerk auch die Kapazitäten hinzuzählt: In diesem Fall kann es sein, dass ein Netzwerk insgesamt für alle Wahlen der Potentialfunktion einen Earliest Arrival Fluss besitzt, obwohl es Teilgraphen enthält, für die es Wahlen der Potentialfunktion gibt, die einen Earliest Arrival Fluss verhindern.

Zum Abschluss seien noch zwei Ideen erwähnt, die in dieser Arbeit nicht mehr untersucht wurden. Der erste Gedankengang beschäftigt sich mit der Frage, ob es möglich ist, dynamische Nullfahrzeitenflüsse zu berechnen, die die Earliest Arrival Eigenschaft *fast* erfüllen. Hoffnung gibt in dieser Hinsicht die Beobachtung, dass das Versenden einer Flusseinheit in einem früheren Zeitschritt in einem späteren Zeitschritt nur das Versenden von zwei Flusseinheiten verhindern kann: Die Abhängigkeit zwischen Zeitschritten entsteht nur durch das Ausgleichen von Potentialen. Eine versendete Flusseinheit ändert das Potential maximal an einer Quelle und an einer Senke und kann so in einem späteren Zeitschritt nur zwei Flusseinheiten behindern. Möglicherweise ist sogar der Flusswert des dynamischen Flusses,

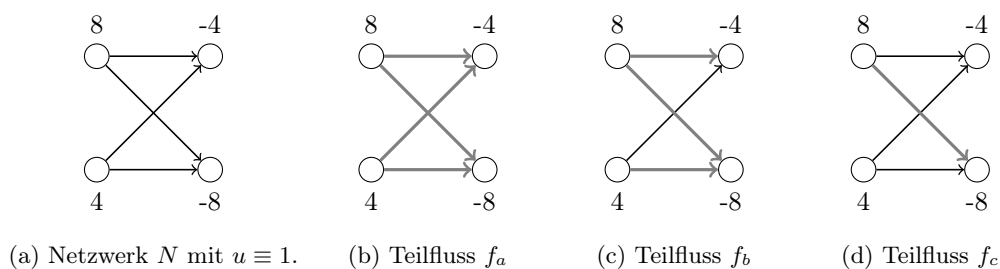


Abbildung 4.1: Ein Netzwerk N und drei verschiedene mögliche Teilflüsse in N .

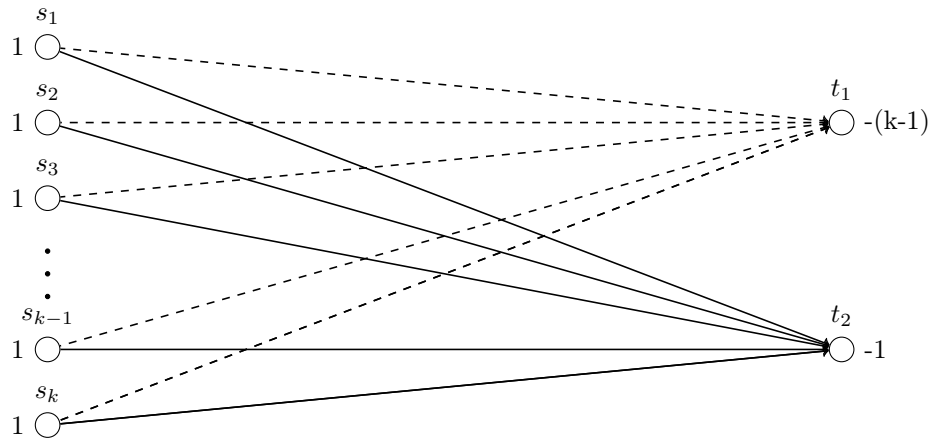


Abbildung 4.2: Ein Netzwerk N' mit k Quellen s_1, \dots, s_k , zwei Senken t_1 und t_2 sowie Einheitskapazitäten.

den man erhält, indem man iterativ maximale Flüsse im Netzwerk berechnet und die verbleibenden Potentiale entsprechend anpasst, nicht besonders weit von dem Wert entfernt, den ein Earliest Arrival Fluss erreichen müsste.

Die zweite Fragestellung beschäftigt sich damit, ob es möglich ist, dass es keinen Earliest Arrival Fluss gibt, aber für jedes Paar aus zwei Zeitschritten ein ausgleichender Fluss existiert, der zu diesen Zeitschritten den Wert erreicht, den ein Earliest Arrival Fluss erreichen müsste. Wenn dies nicht möglich ist, kann man die Existenz eines Earliest Arrival Flusses prüfen, indem man für alle Tupel aus zwei Zeitschritten testet, ob der Wert eines ausgleichenden Flusses für beide Zeitschritte gleichzeitig maximierbar ist. Es reicht aber zumindest nicht aus, Paare von aufeinanderfolgenden Zeitschritten zu überprüfen. In Abbildung 4.1(a) ist ein Netzwerk N mit zwei Quellen und zwei Senken, Nullfahrzeiten und Einheitskapazitäten zu sehen. Der maximale Flusswert $w_{\max}^+(t)$, den ein ausgleichender Fluss in N nach $t \in \{1, \dots, 4\}$ Zeitschritten erreichen haben kann, ist durch $w_{\max}^+(1) = 4$, $w_{\max}^+(2) = 8$, $w_{\max}^+(3) = 9$ und $w_{\max}^+(4) = 12$ gegeben. Es ist möglich, für zwei aufeinanderfolgende Zeitpunkte den maximalen Wert durch einen ausgleichenden Fluss zu erreichen: Um $w_{\max}^+(1) = 4$ und $w_{\max}^+(2) = 8$ zu erreichen, verschickt man in N zweimal den Teilfluss f_a , der in Teilabbildung (b) zu sehen ist. Um $w_{\max}^+(2) = 8$ und $w_{\max}^+(3) = 9$ zu erreichen, versendet man zweimal Teilfluss f_a und im dritten Zeitschritt Teilfluss f_c aus Teilabbildung (d). Damit man $w_{\max}^+(3) = 9$ und $w_{\max}^+(4) = 12$ erreicht, versendet man viermal Teilfluss f_b aus Teilabbildung (c). Es gibt aber zwei (nicht aufeinanderfolgende) Werte, die sich nicht gleichzeitig erreichen lassen, und zwar $w_{\max}^+(2)$ und $w_{\max}^+(4)$.

Es reicht also nicht aus, aufeinanderfolgende Zeitpunkte zu überprüfen. Im Fall allgemeiner Fahrzeiten können wir außerdem für gegebenes k ein Netzwerk angeben, in dem für jede

4 Zusammenfassung und Ausblick

Auswahl von k Zeitschritten alle Zeitschritte gleichzeitig maximierbar sind, $k + 1$ Zeitschritte jedoch nicht gleichzeitig maximierbar sind. Netzwerk N' in Abbildung 4.2 mit k Quellen s_1, \dots, s_k , zwei Senken t_1 und t_2 sowie Einheitskapazitäten erfüllt dieses, wenn man die Fahrzeiten $\tau(s_i, t_1) = i - 1$ sowie $\tau(s_i, t_2) = i$ wählt. Dadurch ist die Fahrzeit von jeder Quelle aus zu t_2 um eins länger als zu t_1 . Aufgrund der Potentiale gibt es genau eine Quelle, deren Flusseinheit trotzdem zu t_2 verschickt werden muss. Für $t \in \{1, \dots, k\}$ ist $w_{\max}^+(t) = t$. Dieser Wert wird von einem ausgleichenden Fluss, der die Flusseinheit aus Quelle s_i nach t_2 verschickt (und alle anderen Flusseinheiten nach t_1 verschickt), für alle t außer $t = i$ erreicht.

Index

- adjazent, 12
- Angebot von s , 19
- Ausgangsgrad, 12
- Baum
 - In-Tree, 14
 - Out-Tree, 14
 - ungerichtet, 14
- Bedarf von t , 19
- Blatt
 - in einem In-Tree, 14
 - in einem Out-Tree, 14
- dynamischer Fluss, 16
 - zulässig, 17
- dynamischer Netzwerkfluss, 16
- dynamischer Nullfahrzeitenfluss, 16
- dynamisches Nullfahrzeitentransshipment, 20
- dynamisches Transshipment mit Nullfahrzeiten, 20
- dynamisches Transshipment mit Zeithorizont T , 20
- Earliest Arrival Transshipment, 21
- Earliest Arrival Transshipment mit Nullfahrzeiten, 21
- Eingangsgrad, 12
- Fahrzeit, 13
- Fluss, 15
 - ausgleichend, 20
 - vollständig ausgleichend, 20
- Flusserhaltung, 19
 - dynamisch, 18
 - statisch, 15
- Flusswert, 17
- Graph, 12
 - gerichtet, 12
 - induzierter ungerichteter, 13
 - ungerichtet, 12
- induzierter Teilgraph, 12
- Kantenkapazität, 13
- konsistent, 19
- Kreis
 - gerichteter, 14
 - ungerichteter, 14
- Latest Departure Eigenschaft, 48
- lineares Programm, 33
- maximaler dynamischer Fluss, 18
- maximaler statischer Fluss, 16
- Netzwerk, 12
 - gerichtet, 12
 - ungerichtet, 12
- Pfad
 - gerichteter, 13
 - ungerichteter, 13
- Potential, 13
 - zulässig, 20
- Quelle, 19

Index

Quickest Transshipment, 21

Quickest Transshipment mit Nullfahrzei-
ten, 21

Senke, 19

statischer Fluss, 15

zulässig, 15

statischer Netzwerkfluss, 15

Teilfluss, 44

Teilgraph, 12

Tiefe, 15

Unterteilung, 56

Wurzel

in einem In-Tree, 14

in einem Out-Tree, 14

zeitexpandierte Netzwerk, 34

zeitexpandierter Graph, 22

zeitexpandiertes Netzwerk, 23

Zeitexpansion, 22

zeitlich wiederholter Fluss, 31

zusammenhängend, 14

ungerichtet, 14

Zusammenhangskomponente, 14

Literaturverzeichnis

- [1] AHUJA, RAVINDRA K., THOMAS L. MAGNANTI und JAMES B. ORLIN: *Network flows*. Prentice Hall Inc., 1993.
- [2] BAUMANN, NADINE und MARTIN SKUTELLA: *Solving evacuation problems efficiently: Earliest arrival flows with multiple sources*. Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, Seiten 399–408, 2006.
- [3] DANTZIG, GEORGE B.: *Maximization of a linear function of variables subject to linear inequalities*. In: *Activity Analysis of Production and Allocation*, Cowles Commission Monograph No. 13, Seiten 339–347. John Wiley & Sons Inc., New York, N. Y., 1951.
- [4] DINIC, E. A.: *An algorithm for the solution of the problem of maximal flow in a network with power estimation*. Doklady Akademii Nauk SSSR, 194:754–757, 1970.
- [5] DRESSLER, DANIEL, MARTIN GROSS, JAN-PHILIPP KAPPMEIER, TIMON KELTER, JOSCHA KULBATZKI, DANIEL PLÜMPE, GORDON SCHLECHTER, MELANIE SCHMIDT, MARTIN SKUTELLA und SYLVIE TEMME: *On the Use of Network Flow Techniques for Assigning Evacuees to Exits*. Eingereicht bei der ICEM 2009.
- [6] EDMONDS, JACK und RICHARD M. KARP: *Theoretical improvements in algorithmic efficiency for network flow problems*. Journal of the ACM, 19:248–264, 1972.
- [7] FLEISCHER, LISA K.: *Faster algorithms for the quickest transshipment problem*. SIAM Journal on Optimization, 12(1):18–35, 2001.
- [8] FLEISCHER, LISA K. und MARTIN SKUTELLA: *Quickest flows over time*. SIAM Journal of Computing, 36(6):1600–1630, 2007.
- [9] FLEISCHER, LISA K. und ÉVA TARDOS: *Efficient continuous-time dynamic network flow algorithms*. Operations Research Letters, 23(3-5):71–80, 1998.
- [10] FORD, JR., LESTER R. und DELBERT R. FULKERSON: *Maximal flow through a network*. Canadian Journal of Mathematics. Journal Canadien de Mathématiques, 8:399–404, 1956.

Literaturverzeichnis

- [11] FORD, JR., LESTER R. und DELBERT R. FULKERSON: *Flows in networks*. Princeton University Press, 1962.
- [12] GALE, DAVID: *Transient flows in networks*. Michigan Mathematical Journal, 6:59–63, 1959.
- [13] GOLDBERG, ANDREW V. und ROBERT E. TARJAN: *A new approach to the maximum-flow problem*. Journal of the Association for Computing Machinery, 35(4):921–940, 1988.
- [14] GROSS, MARTIN, MOUKARRAM KABBASH, JAN-PHILIPP KAPPMEIER, SOPHIA KARDUNG, TIMON KELTER, JOSCHA KULBATZKI, DANIEL PLÜMPE, MARCEL PREUSS, GORDON SCHLECHTER, MELANIE SCHMIDT, SYLVIE TEMME und MATTHIAS WOSTE: *Projektgruppe 517: Evakuierungsprobleme, Endbericht*. Technische Universität Dortmund, 2008.
- [15] GRÖTSCHEL, MARTIN: *Lineare Optimierung. Skriptum zur Vorlesung im Wintersemester 2003/2004*.
- [16] HAČIJAN, L. G.: *A polynomial algorithm in linear programming*. Doklady Akademii Nauk SSSR, 244(5):1093–1096, 1979.
- [17] HAJEK, BRUCE und RICHARD G. OGIER: *Optimal dynamic routing in communication networks with continuous traffic*. Networks, 14(3):457–487, 1984.
- [18] HOPPE, BRUCE: *Efficient dynamic network flow algorithms*. Doktorarbeit, Cornell University, 1995.
- [19] HOPPE, BRUCE und ÉVA TARDOS: *Polynomial time algorithms for some evacuation problems*. In: *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, Seiten 433–441. ACM, 1994.
- [20] KARZANOV, A. V.: *The problem of finding the maximal flow in a network by the method of preflows*. Doklady Akademii Nauk SSSR, 215:49–52, 1974.
- [21] KORTE, BERNHARD und JENS VYGEN: *Combinatorial Optimization*, Band 21 der Reihe *Algorithms and Combinatorics*. Springer, Berlin, 3. Auflage, 2006.
- [22] MEGIDDO, NIMROD: *Combinatorial optimization with rational objective functions*. Mathematics of Operations Research, 4(4):414–424, 1979.
- [23] MEGIDDO, NIMROD: *Applying parallel computation algorithms in the design of serial algorithms*. Journal of the Association for Computing Machinery, 30(4):852–865, 1983.

- [24] MINIEKA, EDWARD: *Maximal, lexicographic, and dynamic network flows*. Operations Research, 21:517–527, 1973.
- [25] PHILPOTT, ANDREW B.: *Continuous-time flows in networks*. Math. Oper. Res., 15(4):640–661, 1990.
- [26] SCHRIJVER, ALEXANDER: *Combinatorial optimization*, Band 24 der Reihe *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2003.
- [27] SKUTELLA, MARTIN: *An Introduction to Network Flows Over Time*. In: COOK, W., L. LOVÁSZ und J. VYGEN (Herausgeber): *Research Trends in Combinatorial Optimization*, Seiten 451–482. Springer, 2009.
- [28] TJANDRA, STEVANUS A.: *Dynamic Network Optimization with Application to the Evacuation Problem*. Doktorarbeit, Universität Kaiserslautern, 2003.
- [29] WILKINSON, W. L.: *An algorithm for universal maximal dynamic flows in a network*. Operations Research, 19:1602–1612, 1971.
- [30] ZADEH, NORMAN: *A bad network problem for the simplex method and other minimum cost flow algorithms*. Mathematical Programming, 5:255–266, 1973.

Literaturverzeichnis