8 Das Problem von Okamura und Seymour

Das Menger-Problem kann man als ein „Wegpackungsproblem“ auffassen, d.h. es sollen knotendisjunkte bzw. kantendisjunkte Wege zwischen vorgegebenen Knoten in einen Graph „gepackt“ werden. Ein allgemeineres kantendisjunktes Wegpackungsproblem kann folgendermaßen formuliert werden.

> Kantendisjunktes Wegpackungsproblem
> Gegeben sei ein Graph $G = (V, E)$ und Paare ausgezeichneter Knoten $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$, $s_i, t_i \in V$ (nicht notwendig paarweise verschieden). Finde paarweise kantendisjunkte s_i-t_i-Wege p_i in G, $1 \leq i \leq k$. Die s_i, t_i werden *Terminale* genannt, die Mengen $\{s_i, t_i\}$ heißen *Netze*.

Dieses Problem ist $\mathcal{NP}$-vollständig, auch falls G planar ist. Naheliegende Einschränkungen des Problems in planaren Graphen sind:

- Die Lage der s_i, t_i ist „eingeschränkt“, etwa alle s_i, t_i, $1 \leq i \leq k$ liegen auf dem Rand derselben Facette.
- Sei $D := \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ (Menge der Demand-Kanten), dann soll $G + D := (V, E \cup D)$ planar sein.

Ein wichtiges Kriterium für die Lösbarkeit solcher Probleme ist ähnlich wie beim kantendisjunkten Menger-Problem die „Kapazität“ des Graph.

Definition 8.1. *Sei* $G = (V, E)$, $X \subseteq V$. *Dann heißt*

$$\mathrm{cap}(X) := |\{\{u, v\} \in E : u \in X, v \in V \backslash X\}|$$

die Kapazität *von* X *(Größe des durch* X *induzierten Schnittes). Zu* $G = (V, E)$ *sei* $D = \{\{s_i, t_i\} : s_i, t_i \in V, 1 \leq i \leq k\}$ *gegeben. Dann heißt*

$$\mathrm{dens}(X) := |\{\{s_i, t_i\} \in D : |\{s_i, t_i\} \cap X| = 1\}|$$

die Dichte *von* X.

$$\mathrm{fcap}(X) := \mathrm{cap}(X) - \mathrm{dens}(X)$$

bezeichne die freie Kapazität *von* X. X *heißt* saturiert*, falls* $\mathrm{fcap}(X) = 0$ *und* übersaturiert*, falls* $\mathrm{fcap}(X) < 0$.

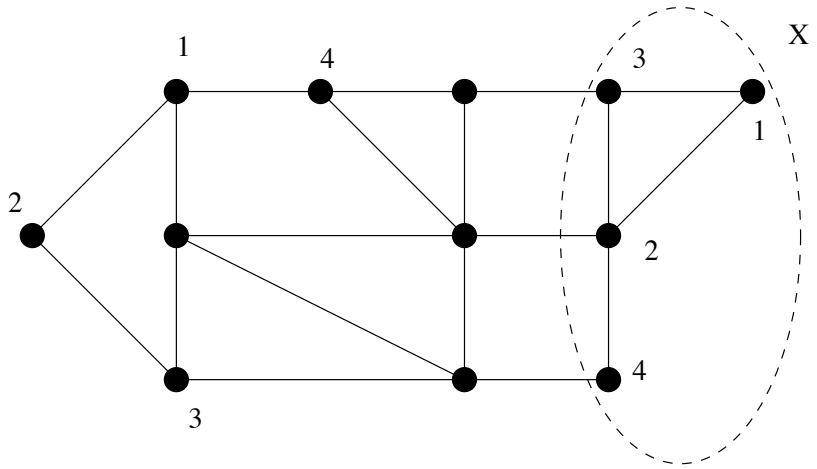

Abbildung 8.1: Ein Problembeispiel mit Schnitt X, für den $\mathrm{cap}(X) = 3$, $\mathrm{dens}(X) = 4$ und $\mathrm{fcap}(X) < 0$ gilt

Eine notwendige Bedingung für die Lösbarkeit des kantendisjunkten Wegpackungsproblems ist offensichtlich

$$\mathrm{fcap}(X) \geq 0 \quad \text{für alle} \quad X \subseteq V.$$

Wir nennen diese Bedingung *Kapazitätsbedingung*. Im allgemeinen ist die Kapazitätsbedingung keine hinreichende Bedingung für die Lösbarkeit des kantendisjunkten Wegpackungsproblems. Siehe Abb. 8.2.

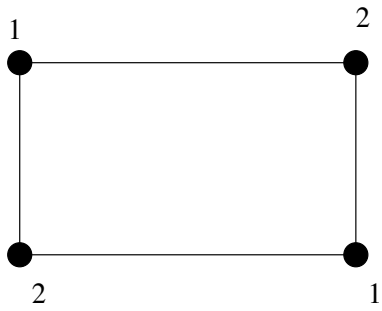

Abbildung 8.2: Ein Problembeispiel, das die Kapazitätsbedingung erfüllt, d.h. $\mathrm{cap}(X) \geq 2$ für alle $\emptyset \neq X \subset V$ und $\mathrm{dens}(X) \leq 2$, aber nicht lösbar ist.

Ein Graph $G = (V, E)$ mit $D = \{\{s_i, t_i\} : s_i, t_i \in V,\ 1 \leq i \leq k\}$ erfüllt die *Geradheitsbedingung* (auch *Euler-Bedingung* genannt), falls $\mathrm{fcap}(X)$ gerade ist für alle $X \subseteq V$.

Wir betrachten ab jetzt folgenden Spezialfall des kantendisjunkten Wegpackungsproblems:

Okamura & Seymour-Problem
Gegeben sei ein planarer Graph $G = (V, E)$ und Paare ausgezeichneter Knoten $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$, $s_i, t_i \in V$, wobei alle s_i, t_i auf dem Rand derselben Facette liegen. O.B.d.A. sei G so planar eingebettet, dass s_i, t_i, $1 \leq i \leq k$ auf der äußeren Facette liegen. Außerdem sei die Geradheitsbedingung erfüllt. Finde paarweise kantendisjunkte s_i-t_i-Wege p_i in G, $1 \leq i \leq k$.

Lemma 8.2. *Sei* $G = (V, E)$, $D := \{\{s_i, t_i\} : s_i, t_i \in V, 1 \leq i \leq k\}$. *Es gilt* $\mathrm{fcap}(X)$ *gerade für alle* $X \subseteq V$ *genau dann, wenn* $\mathrm{fcap}(v) := \mathrm{fcap}(\{v\})$ *gerade für alle* $v \in V$.

Beweis. „$\Longrightarrow$“ ist trivial.

„$\Longleftarrow$“ Sei also $\mathrm{fcap}(v)$ gerade für alle $v \in V$. Es gilt für $X \subseteq V$

$$\mathrm{cap}(X) = \sum_{v \in X} \mathrm{cap}(v) - 2 \cdot \left|\left\{\{u, v\} \in E : u, v \in X\right\}\right| \quad \text{und}$$
$$\mathrm{dens}(X) = \sum_{v \in X} \mathrm{dens}(v) - 2 \cdot \left|\left\{\{s_i, t_i\} \in D : s_i, t_i \in X\right\}\right|.$$

Dann ist

$$\begin{aligned} \mathrm{fcap}(X) &= \sum_{v \in X} \mathrm{cap}(v) - \sum_{v \in X} \mathrm{dens}(v) - 2 \cdot \left|\left\{\{u, v\} \in E : u, v \in X\right\}\right| \\ &\quad +2\left|\left\{\{s_i, t_i\} \in D : s_i, t_i \in X\right\}\right| \\ &= \sum_{v \in X} \mathrm{fcap}(v) - 2 \cdot \Big(\left|\left\{\{u, v\} \in E : u, v \in X\right\}\right| \\ &\quad +\left|\left\{\{s_i, t_i\} \in D : s_i, t_i \in X\right\}\right|\Big) \end{aligned}$$

Damit ist $\mathrm{fcap}(X)$ gerade, falls alle $\mathrm{fcap}(v)$ für $v \in X$ gerade sind. □

Satz 8.3 (Satz von Okamura & Seymour, 1981). *Gegeben sei ein planar eingebetteter Graph* $G = (V, E)$ *und* $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$, *wobei* $s_1, \ldots, s_k, t_1, \ldots, t_k$ *auf dem Rand der äußeren Facette von* G *liegen. Die Kapazitätsbedingung und die Geradheitsbedingung seien erfüllt. Dann existieren paarweise kantendisjunkte* s_i-t_i-*Wege in* G.

Der Beweis von Okamura & Seymour zu Satz 8.3 führt direkt zu einem $\mathcal{O}(n^5)$ Algorithmus zur Lösung des kantendisjunkten Wegpackungsproblems in Graphen, die die Bedingungen von Satz 8.3 erfüllen. Dieser kann auf $\mathcal{O}(n^2)$ verbessert werden (Becker & Mehlhorn 1986; Matsumoto, Nishizeki & Saito 1985). Dabei werden explizit Kapazitäten und Dichten von Schnitten untersucht unter Benutzung der Dualität zwischen Schnitten und Kreisen.

Wir werden hier einen Linearzeitalgorithmus behandeln, der wiederum auf einer Tiefensuche mit Right-First-Auswahlregel basiert.

Sei im folgenden $G = (V, E)$ planar eingebettet, so dass $s_1, t_1, \ldots, s_k, t_k$ auf dem Rand der äußeren Facette liegen und die Geradheitsbedingung sei erfüllt. O.B.d.A. sei G zweifach zusammenhängend; diese Annahme dient nur der Vereinfachung der Darstellung.

Der Algorithmus besteht aus zwei Phasen. Zunächst wird mittels einer Right-First-Tiefensuche eine Lösung für das kantendisjunkte Wegpackungsproblem für ein modifiziertes Problembeispiel mit einer „einfacheren“ Struktur bestimmt. Diese Lösung induziert einen gerichteten Hilfsgraphen $\overrightarrow{G}$, in dem dann wieder mittels Right-First-Tiefensuche das Ausgangsproblem gelöst wird.

Linearzeitalgorithmus für das Okamura & Seymour-Problem

Schritt 1: Konstruiere aus $G = (V, E)$ mit $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ ein Problem bestehend aus $G = (V, E)$ mit $D' = \{\{s'_1 t'_1\}, \ldots, \{s'_k, t'_k\}\}$ mit „einfacherer“ Struktur.

Schritt 2: Gegeben sei $G = (V, E)$ und $D' = \{\{s'_i, t'_i\}, \ldots, s'_k, t'_k\}\}$. Berechne in $\mathcal{O}(n)$ kantendisjunkte s'_i-t'_i-Wege mittels Right-First-Tiefensuche und Orientierung der entsprechenden Wege von s'_i nach t'_i. Diese induzieren einen gerichteten Graph $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$.

Schritt 3: Gegeben sei nun der durch Schritt 2 induzierte Graph $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$ und $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$. Berechne in $\mathcal{O}(n)$ kantendisjunkte gerichtete s_i-t_i-Wege p_i in $\overrightarrow{G}$, welche kantendisjunkte s_i-t_i-Wege in G induzieren.

Definition 8.4. $G = (V, E)$ *mit* $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ *hat* Klammerstruktur*, falls* $G + D$ *so planar eingebettet werden kann, dass die Kanten aus* D *kreuzungsfrei in die äußere Facette der entsprechenden Einbettung von* G *eingebettet sind. Siehe Abb. 8.3.*

Ausführung von Schritt 1: Konstruktion von D' mit Klammerstruktur.

Konstruiere aus $G = (V, E)$ mit $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ ein Problem bestehend aus $G = (V, E)$ mit $D' = \{\{s'_1 t'_1\}, \ldots, \{s'_k, t'_k\}\}$, so dass $\{s_1, \ldots, s_k, t_1, \ldots, t_k\} = \{s'_1, \ldots, s'_k, t'_1, \ldots, t'_k\}$ und die $\{s'_1, t'_1\}, \ldots, \{s'_k, t'_k\}$ *Klammerstruktur* haben.

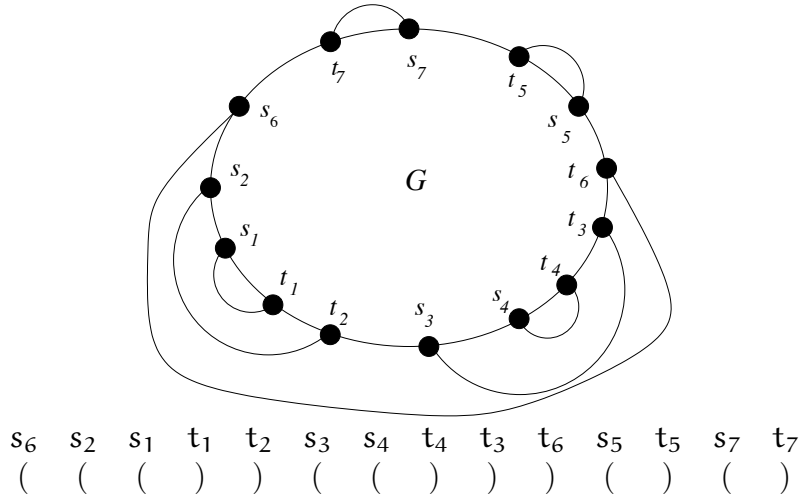

Abbildung 8.3: Beginnend mit einem beliebigen s_i, z.B. s_6 haben die Netze *Klammerstruktur* d.h. Paarung der $\{s_i, t_i\}$ entspricht einer korrekten Klammerung der entsprechenden Klammern.

Zu einem Problembeispiel $G = (V, E)$ mit $D = \{\{s_1, t_1, \}, \ldots, \{s_k, t_k\}\}$ (ohne Klammerstruktur) kann ein Problembeispiel G, D' mit Klammerstruktur leicht in $\mathcal{O}(n)$ konstruiert werden. Wähle dazu ein beliebiges Terminal (s_i oder t_i) als Startterminal s. Beginnend bei s gehe im Gegenuhrzeigersinn um die äußere Facette. Dem jeweils ersten Terminal eines $\{s_i, t_i\}$ ordne eine öffnende Klammer zu und dem jeweils zweiten eine schließende Klammer. Die korrekte Klammerung dieser Klammern beginnend mit der Klammer zu s induziert D' (Realisierung mit STACK siehe Übung).

Ausführung von Schritt 2: Berechnung der s'_i-t'_i-Wege $q_1, \ldots, q_k$

Gegeben sei $G = (V, E)$ und $D' = \{\{s'_i, t'_i\}, \ldots, s'_k, t'_k\}\}$ mit Klammerstruktur. Berechne in $\mathcal{O}(n)$ kantendisjunkte s'_i-t'_i-Wege mittels Right-First-Tiefensuche und eine Orientierung der entsprechenden Wege von s'_i nach t'_i.

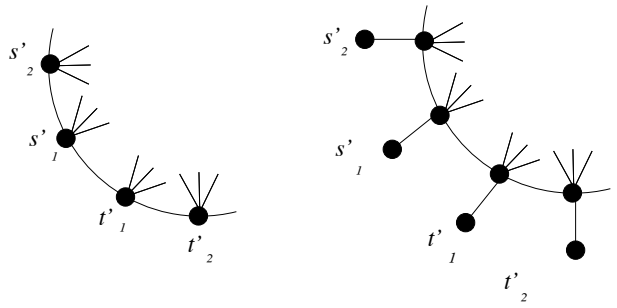

Abbildung 8.4: Aus technischen Gründen hinzugefügte „Dummy-Kanten“.

Die s'_i, t'_i seien, beginnend beim Startterminal s im Gegenuhrzeigersinn so angeordnet, dass jeweils s'_i vor t'_i und t'_i vor t'_{i+1}. Aus technischen Gründen füge jeweils Kanten wie in Abb. 8.4 hinzu.

RIGHT-FIRST-PROZEDUR ($G = (V, E)$, $D' = \{\{s'_i, t'_i\}$)

1: **Für** $i := 1$ bis k **führe aus**
2: Füge zu q_i die eindeutige zu s'_i inzidente Kante ausgehend aus s'_i orientiert als führende Kante hinzu.
3: Setze $v \leftarrow$ eindeutiger zu s'_i adjazenter Knoten.
4: **Solange** $v \notin \{s'_j, t'_j, 1 \leq j \leq k\}$ **führe aus**
5: Sei $\{v, w\}$ rechteste freie Kante bzgl. der führenden Kante von q_i.
6: Füge (v, w) zu q_i als führende Kante hinzu.
7: Setze $v \leftarrow w$.
8: **Ende** „**Solange**“
9: **Falls** $v \neq t'_i$ **dann**
10: gib „unlösbar“ aus.
11: **Ende** „**Falls**“
12: **Ende** „**Für**“
13: Gib $q_1, \ldots, q_k$ aus.

Offensichtlich endet wegen der Geradheitsbedingung jeder Durchlauf von 1. bei einem Terminalknoten.

Beobachtung. *Wegen der Right-First-Auswahlregel können an einem Knoten v Reihenfolgen von Kanten wie in Abb. 8.5 nicht vorkommen. Daher gilt für die Wege q_i:*

i. Keine zwei Wege q_i und q_j kreuzen sich.

ii. Kein Weg q_i kreuzt sich selbst.

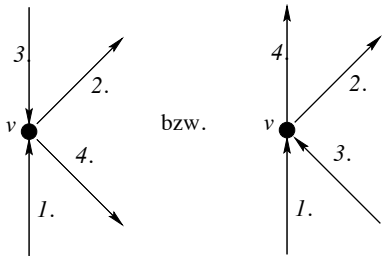

Abbildung 8.5: Reihenfolgen von Kanten, die wegen der Right-First-Auswahlregel nicht vorkommen können.

Sei $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$, $\overrightarrow{V} \subseteq V$, der Graph, der durch die von $q_1, \ldots, q_k$ belegten Kanten zusammen mit der Orientierung induziert wird. Dann gilt für $v \in \overrightarrow{V} \backslash \{s'_i, t'_i : 1 \leq i \leq k\}$, dass $d^{\leftarrow}(v) = d^{\rightarrow}(v)$ in $\overrightarrow{G}$.

Corollar 8.5. $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$ *enthält keinen Rechtskreis.*

Beweis. Wenn $\overrightarrow{C}$ Kanten eines Rechtskreises in $\overrightarrow{G}$ sind, so gehören nicht alle Kanten aus $\overrightarrow{C}$ zu dem selben Weg q_i. Seien q_i, q_j Wege, die Kanten aus $\overrightarrow{C}$ besetzen. Da q_i und q_j sich nicht kreuzen, müssen die Terminale s'_i, t'_i und s'_j, t'_j in der Reihenfolge s'_i, t'_j, s'_j, t'_i im Gegenuhrzeigersinn auf der äußeren Facette von G liegen. Dies ist ein Widerspruch. Siehe Abb. 8.6. □

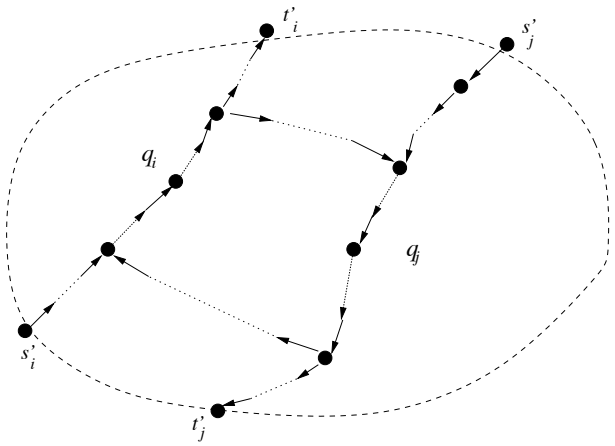

Abbildung 8.6: Durch einen Rechtskreis induzierte Reihenfolge der Terminale s'_i, t'_j, s'_j, t'_i.

Lemma 8.6. *Für ein lösbares Problem des Wegpackungsproblems* $G = (V, E)$ *und* $D' = \{\{s'_1, t'_1\}, \ldots, \{s'_k, t'_k\}\}$*, wobei* D' *Klammerstruktur hat,* s'_i, t'_i *auf dem Rand der äußeren Facette liegen und die Geradheitsbedingung erfüllt ist, berechnet die RIGHT-FIRST-PROZEDUR(*G, D'*) kantendisjunkte* s'_i-t'_i*-Wege* q_i*, für* $1 \leq i \leq k$.

Beweis. Betrachte eine beliebige kreuzungsfreie Lösung $q'_1, ..., q'_k$. Eine solche Lösung existiert immer! Dann können wir für jedes q'_i dessen linke und deren rechte Seite betrachten. Ebenso können wir für die von der RIGHT-FIRST-PROZEDUR konstruierten Wege q_i linke und rechte Seite betrachten.

Wenn $q'_1, ..., q'_k$ kreuzungsfrei ist, so gilt für jedes q'_i, dass es vollständig zur linken Seite aller $q'_1, \ldots, q'_{i-1}$ gehört. Induktiv über i, $1 \leq i \leq k$, gilt für jedes q_i, da es mittels Right-First-Auswahlregel und entsprechend der Klammerstruktur von D' bestimmt wurde, dass die linke Seite von q'_i ganz enthalten ist in der linken Seite von q_i. Daraus folgt insbesondere, dass q_i die Terminale s'_i und t'_i verbindet. □

Laufzeit: Offensichtlich ist Schritt 2 amortisiert in $\mathcal{O}(n)$ realisierbar, da die rechteste freie Kante bzgl. der führenden Kante immer die nächste Kante nach der führenden Kante in der (aktuellen) Adjazenzliste ist, und damit in konstanter Zeit gefunden werden kann.

Ausführung von Schritt 3: Berechnung der s_i-t_i-Wege $p_1, \ldots, p_k$

Gegeben sei nun der durch Schritt 2 induzierte Graph $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$ und $D = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$, wobei ausgehend vom Startterminal s aus Schritt 1 im Gegenuhrzeigersinn s_i vor t_i liegt, und o.B.d.A. die Indizierung so ist, dass t_i vor t_{i+1}. Berechne in $\mathcal{O}(n)$ kantendisjunkte s_i-t_i-Wege p_i in $\overrightarrow{G}$ mittels gerichteter Right-First-Tiefensuche, und zwar der Reihe nach entsprechend der Indizierung der $\{s_i', t_i'\}$ (also entsprechend dem Auftreten der t_i ausgehend von s im Gegenuhrzeigersinn).

RIGHT-FIRST-PROZEDUR ($\overrightarrow{G}$, D)

1: **Für** $i = 1$ bis k **führe aus**
2: Füge zu p_i die eindeutige aus s_i herausführende Kante von $\overrightarrow{G}$ als führende Kante hinzu.
3: Setze $v \leftarrow$ Einlaufknoten dieser Kante.
4: **Solange** $v \notin \{s_j, t_j : 1 \leq j \leq k\}$ **führe aus**
5: Sei (v, w) die rechteste freie Kante, die aus v herausführt bzgl. der führenden Kante von p_i, dann füge (v, w) zu p_i als führende Kante hinzu.
6: Setze $v \leftarrow w$.
7: **Ende** „**Solange**“
8: **Falls** $v \neq t_i$ **dann**
9: gib aus „unlösbar“.
10: **Ende** „**Falls**“
11: **Ende** „**Für**“
12: Gib $p_1, \ldots, , p_k$ aus.

Korrektheit: Um zu beweisen, dass Schritt 3 korrekte s_i-t_i-Wege p_i konstruiert für ein lösbares Problem geben wir einen Linearzeitalgorithmus an, der für jedes p_i einen Weg p angibt mit der Eigenschaft:

i. Falls p_i korrekt s_i mit t_i verbindet, so induziert p einen saturierten Schnitt in G.

ii. Falls p_i Terminal s_i nicht mit t_i verbindet, aber jedes p_j, $1 \leq j \leq i$ korrekt s_j mit t_j verbindet, so induziert p einen übersaturierten Schnitt in G.

Mit ii. ist im Fall, dass ein p_i nicht s_i mit t_i verbindet, die Kapazitätsbedingung, und damit eine notwendige Bedingung für Lösbarkeit, verletzt.

Sei p_i der i-te Weg, der konstruiert wurde, p_j, $1 \leq j < i$ seien die zuvor konstruierten Wege und verbinden jeweils s_j und t_j. Weg p_i ende bei Knoten t. Dann ist $t \in \{t_i, \ldots, t_k\}$. Folgende Prozedur besteht aus einer „Left-First-Tiefensuche“ rückwärts und berechnet einen Weg p, der einen Schnitt mit den gewünschten Eigenschaften induziert.

SCHNITT-PROZEDUR $(p_1, \ldots, p_i)$

1: p sei die eindeutige in t einlaufende Kante
2: $v \leftarrow$ Auslaufknoten dieser Kante.
3: **Solange** $v \neq \{s_1, \ldots, s_k\}$ **führe aus**
4: Sei (v, x) die letzte zu p hinzugefügte Kante und (u, v) die im Uhrzeigersinn erste Kante nach (v, x), die nicht durch p besetzt ist,
5: dann füge (u, v) zu p hinzu.
6: Setze $v := u$.
7: **Ende** „**Solange**“
8: Gib p aus.

Offensichtlich kann die SCHNITT-PROZEDUR in $\mathcal{O}(n)$ realisiert werden. Wir beweisen, dass die Menge der Kanten $\{u, v\}$ aus G, für die u auf p und v rechts von p liegt, einen saturierten bzw. übersaturierten Schnitt in G, D induzieren. Dazu muss natürlich die rechte Seite von p wohldefiniert sein.

Lemma 8.7. *Der Weg p, der von der SCHNITT-PROZEDUR konstruiert wird, enthält keine Kreuzung. Insbesondere sind seine linke und rechte Seite wohldefiniert.*

Beweis. Angenommen p würde sich selbst kreuzen. Dann gibt es einen einfachen Rechtskreis oder einen einfachen Linkskreis auf p mit Kreuzung in einem Knoten v. Da $\overrightarrow{G}$ nach Korollar 8.5 keinen Rechtskreis enthält, gibt es in v einen einfachen Linkskreis, wobei die entsprechenden Kanten von p, die zu v inzident sind, die in Abb. 8.7 gezeigte Konstellation bilden.

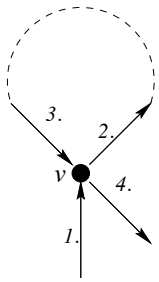

Abbildung 8.7: Illustration zu Lemma 8.7.

Dies ist ein Widerspruch zur Vorgehensweise in Schritt 2 der SCHNITT-PROZEDUR.□

Lemma 8.8. *Sei A die Menge der Kanten $\{u, v\}$ aus G mit u auf p und v rechts von p. Jede Kante $\{u, v\} \in A$ mit u auf p gehört zu $\overrightarrow{G}$, und zwar in der Orientierung (u, v) genau dann, wenn sie von einem der p_j, $1 \leq j < i$, besetzt ist.*

Beweis. Sei $\{u, v\} \in A$ mit u auf p. Falls $\{u, v\}$ von einem der p_j, $1 \leq j < i$ besetzt ist, so ist deren Orientierung (u, v) in $\overrightarrow{G}$, nach Konstruktion von p.

Fall 1: Angenommen $\{u, v\} \in A$ mit u auf p habe Orientierung (u, v) in $\overrightarrow{G}$,und sei nicht durch eines der p_i besetzt.

Betrachte die zu p gehörenden Kanten (x, u), (u, y) inzident zu p, für die $\{u, v\}$ rechts von (x, u) und (u, y) liegt. Die Kante, die bei der Berechnung der $p_1, \ldots, p_i$ unmittelbar vor (u, y) gewählt wurde, ist dann (x, u) oder liegt links von (x, u) und (u, y). Dann ist aber die Wahl von (u, y) ein Widerspruch zur Right-First-Auswahlregel. Siehe Abb. 8.8.

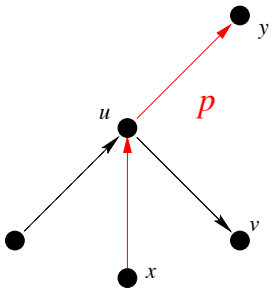

Abbildung 8.8: Illustration zu Fall 1 aus Lemma 8.8.

Fall 2: Betrachte nun eine Kante $\{u, v\} \in A$, mit u auf p, die nicht zu $\overrightarrow{G}$ gehört. Dann können wegen der Right-First Auswahlregel die Kanten (x, u), (u, y) von p, für die $\{u, v\}$ rechts liegt, nicht beide zu dem selben Weg q'_i aus $\overrightarrow{G}$ gehören. Gehöre (x, u) also zu q'_j und (u, y) zu q'_l, $j \neq l$. Dann liegt die Vorgängerkante von (u, y) auf q'_l rechts von $\{u, v\}$ und (u, y), und kann daher von keinem der $p_1, \ldots, p_i$ besetzt sein. Dann muss es eine weitere Kante (z, u) in $\overrightarrow{G}$ geben, die von einem der $p_1, \ldots, p_i$ besetzt ist und links von (x, u) und (u, y) liegt, und eine weitere Kante (u, w), die Nachfolgerkante von (z, u) auf dem entsprechenden Weg q'_r in $\overrightarrow{G}$ ist, und auch links von (x, u), (u, y) liegt. Dann liegt aber die Kante $\{u, v\}$ rechts von q'_r im Widerspruch zur Right-First-Auswahlregel. Siehe Abb. 8.9.

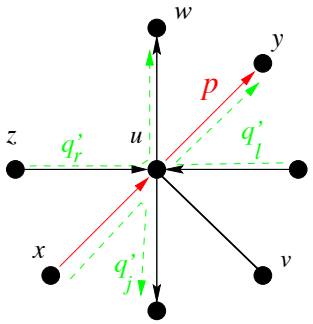

Abbildung 8.9: Illustration zu Fall 2 aus Lemma 8.8.

Lemma 8.9. *Sei $X \subseteq V$ Menge der Knoten rechts von p. Falls p_i ein s_i-t_i-Weg ist, ist X saturiert, ansonsten ist X übersaturiert.*

Beweis. Alle Kanten $\{u, v\}$ mit u auf p und $v \in X$ gehören entweder zu einem Weg p_j, $1 \leq j < i$, mit $s_j \in V \backslash X$ und $t_j \in X$, oder zu einem Weg q'_j, mit $s'_j \in X$ und $t'_j \in V \backslash X$. Wenn p_i ein s_i-t_iWeg ist, dann ist damit

$$\begin{aligned} \operatorname{cap}(X) &= \left| \left\{ \{s_j, t_j\} \colon s_j \in V \backslash X,\ t_j \in X,\ 1 \leq j \leq i \right\} \right| \\ &\quad + \left| \left\{ \{s'_j, t'_j\} \colon s'_j \in X \text{ und } t'_j \in V \backslash X,\ \{s'_j, t'_j\} \notin \left\{ \{s_1, t_1\}, \ldots, \{s_i, t_i\} \right\} \right\} \right| \\ &= \operatorname{dens}(X) \ . \end{aligned}$$

Wenn p_i kein s_i-t_i-Weg ist, so verbindet p_i das Terminal s_i mit einem Terminal $t \in \{t_j;\ i < j \leq k\}$. Da $s_i \in V \backslash X$ und $t_i \in X$ ist, dann gilt

$$\operatorname{cap}(X) \ \leq \ \operatorname{dens}(X) - 1 \ . \qquad \square$$

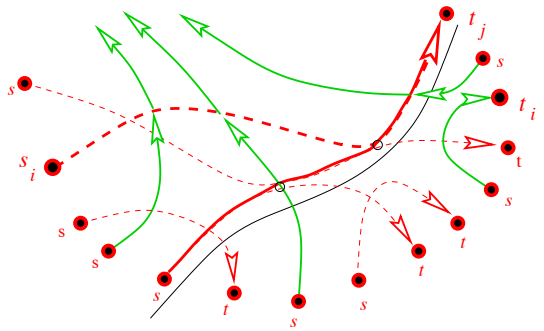

Abbildung 8.10: Illustration der SCHNITT-PROZEDUR. Der Weg p (rot durchgezogen), der von der SCHNITT-PROZEDUR berechnet wird, darunter der durch p induzierte Schnitt, der in diesem Fall übersaturiert ist, da er zusätzlich zu bereits verbundenen Terminalpaaren noch $\{s_i, t_i\}$ trennt. Wege p_i sind rot gestrichelt, Wege q_j grün.

Laufzeit: Mit UNION-FIND kann die RIGHT-FIRST-PROZEDUR zu Schritt 3 in $\mathcal{O}(n)$ realisiert werden.

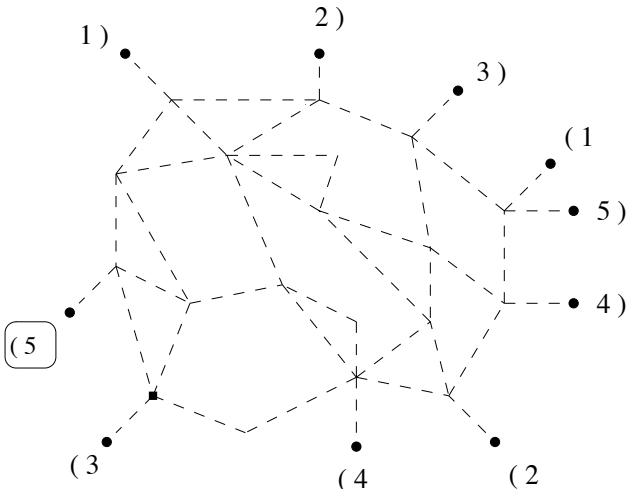

Abbildung 8.11: Ein Problembeispiel für das Problem von Okamura & Seymour und zugehöriges Problembeispiel mit Klammerstruktur bei Wahl von 5 als Startterminal.

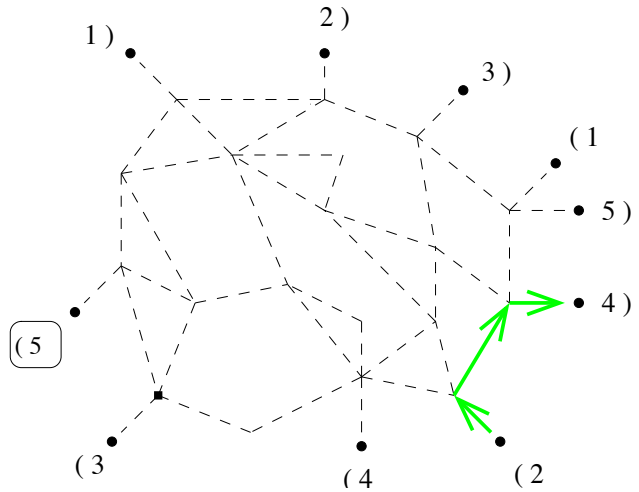

Abbildung 8.12: Der erste Weg, den die RIGHT-FIRST-PROZEDUR für das Problembeispiel mit Klammerstruktur berechnet.

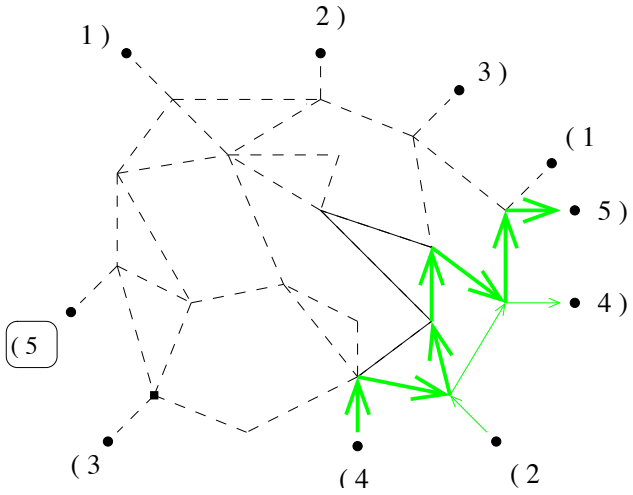

Abbildung 8.13: Der zweite Weg, den die RIGHT-FIRST-PROZEDUR für das Problembeispiel mit Klammerstruktur berechnet.

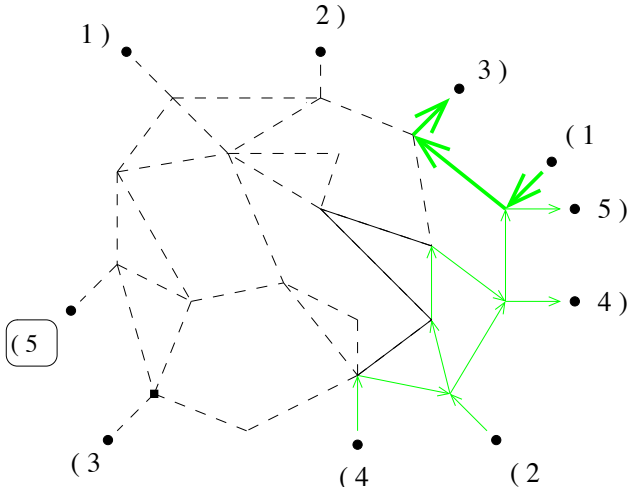

Abbildung 8.14: Der dritte Weg, den die RIGHT-FIRST-PROZEDUR für das Problembeispiel mit Klammerstruktur berechnet.

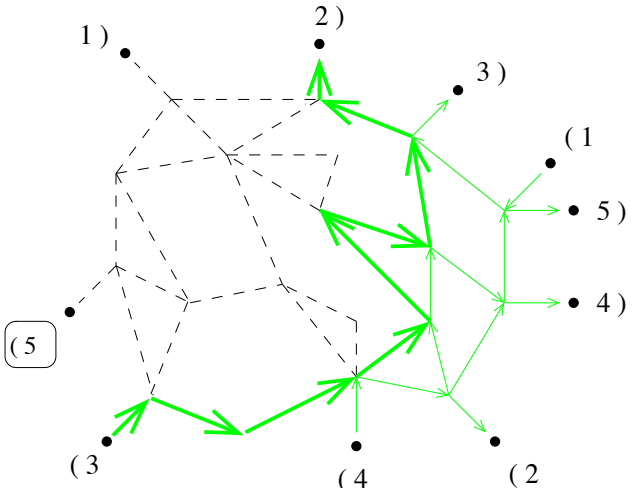

Abbildung 8.15: Der vierte Weg, den die RIGHT-FIRST-PROZEDUR für das Problembeispiel mit Klammerstruktur berechnet.

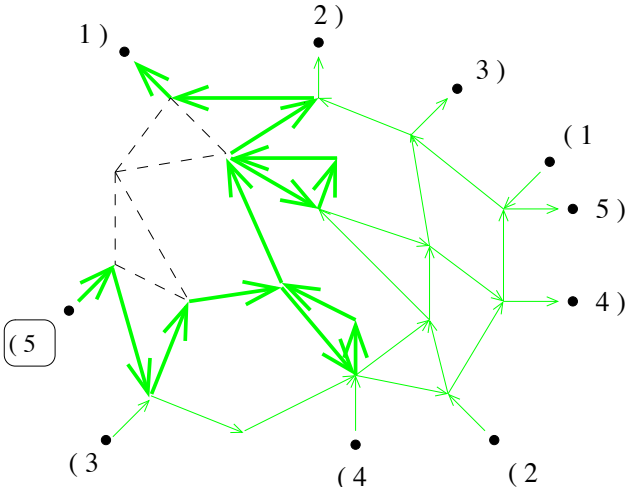

Abbildung 8.16: Der fünfte und letzte Weg, den die RIGHT-FIRST-PROZEDUR für das Problembeispiel mit Klammerstruktur berechnet.

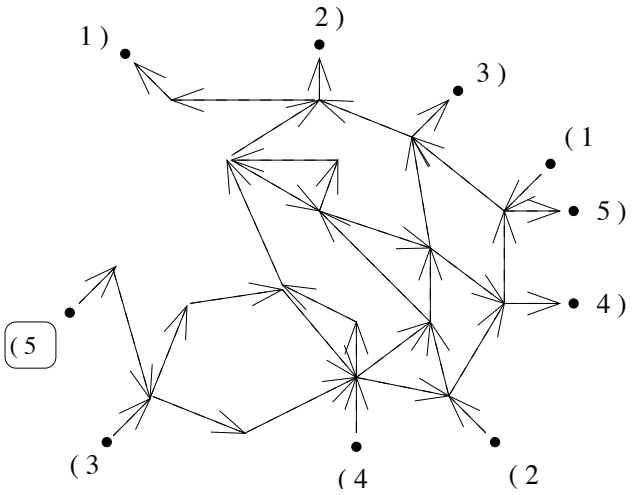

Abbildung 8.17: Der in Schritt 2 berechnete Hilfsgraph $\overrightarrow{G}$, in dem die RIGHT-FIRST-PROZEDUR für G und D arbeitet.

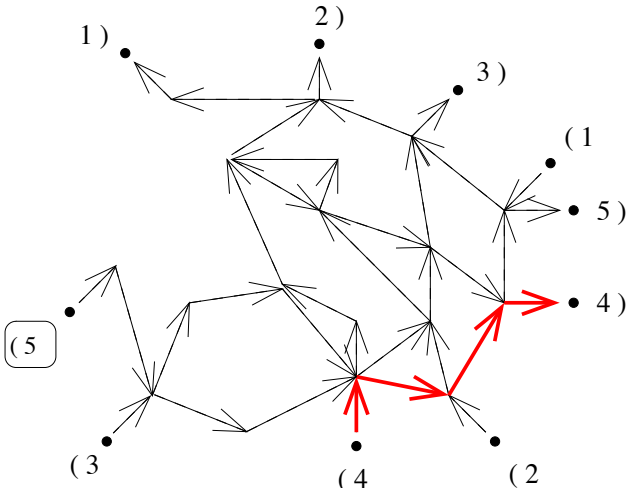

Abbildung 8.18: Der erste Weg, den die RIGHT-FIRST-PROZEDUR für G und D berechnet, verbindet Terminalpaar 4.

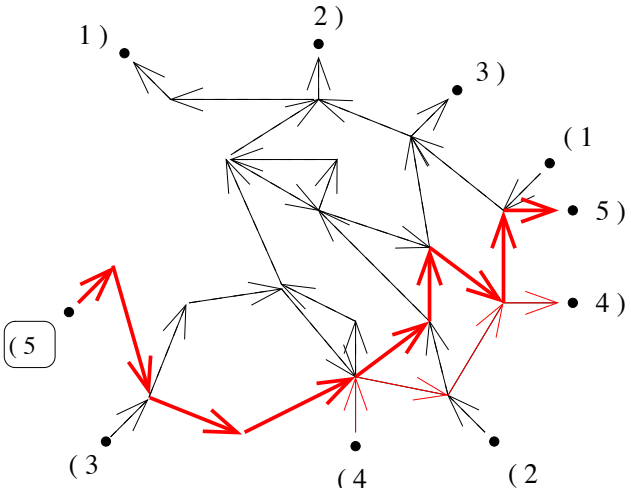

Abbildung 8.19: Der zweite Weg, den die RIGHT-FIRST-PROZEDUR für G und D berechnet, verbindet Terminalpaar 5.

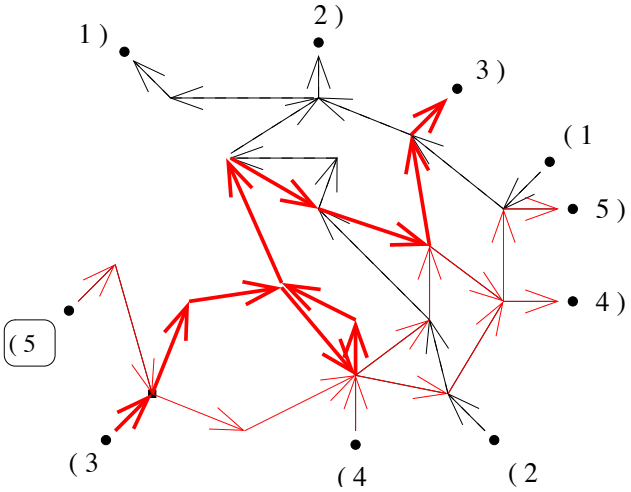

Abbildung 8.20: Der dritte Weg, den die RIGHT-FIRST-PROZEDUR für G und D berechnet, verbindet Terminalpaar 3.

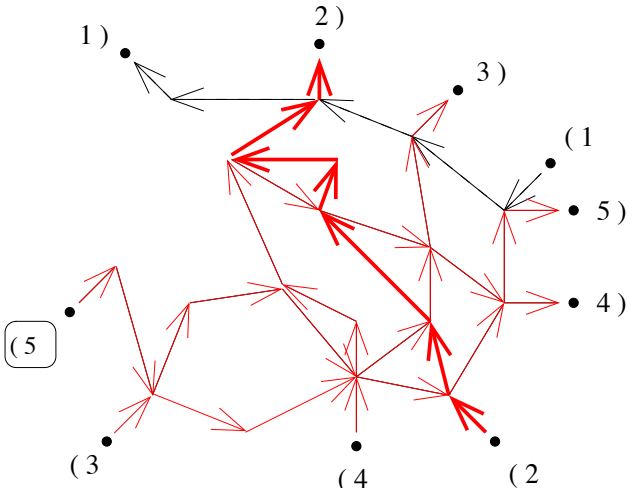

Abbildung 8.21: Der vierte Weg, den die RIGHT-FIRST-PROZEDUR für G und D berechnet, verbindet Terminalpaar 2.

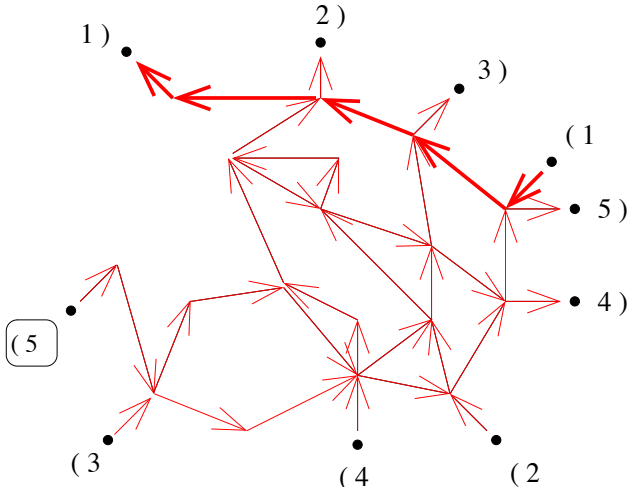

Abbildung 8.22: Der fünfte Weg, den die RIGHT-FIRST-PROZEDUR für G und D berechnet, verbindet Terminalpaar 1.