Schnellere Algorithmen zum Finden von kleinen Kantenschnitten in planaren Graphen

Nach „Faster Algorithms for Finding Small Edge Cuts in Planar Graphs“

von Satish B. Rao

Timo Schreiter, TU Berlin

Juli 2000

Zusammenfassung

Es wird ein Algorithmus vorgestellt, welcher einen optimalen b-limitierten gewichteten Kantenschnitt für eine festgelegte Balance $b \leq 1/3$ bzw. einen 1,5-fachen optimalen Kantenschnitt in $\mathcal{O}(n^2 \min(W, C))$ Zeit liefert, wobei W und C Gesamtgewicht bzw. -kosten des Graphen sind.

Inhaltsverzeichnis

1 Einführung **3**
1.1 Wichtige Definitionen 3

2 Überlegungen zum Algorithmus **4**
2.1 Kantenschnitte und Kreise 4
2.2 Connected Circuits und gewichtete Kosten 4

3 Vorbereitende Sätze **6**
3.1 Ein optimaler b-limitierter gewichteter Schnitt mit 2 Komponenten 6
3.2 „Nur einmal kreuzen“ 7

4 Der Algorithmus **8**
4.1 Konstruktion des Hilfsgraphen 8
4.2 Ein gerichtetes-Pfad-Problem 14
4.3 Lösung des gerichteten-Pfad-Problems 15
4.4 Eine Abwandlung des Pfad-Problems 16

5 Schnelle Approximationen gewichteter Schnitte **17**

Abbildungsverzeichnis

2.1 Konstruktion des Dualgraphen 5
2.2 Ein Kantenschnitt entspricht einem Kreis im Dualgraphen 5
2.3 Ein Connected Circuit 6
2.4 Inneres und Äußeres eines Connected Circuit 6
3.1 Ein Connected Circuit wird entdeckt 8
4.1 Die Reihenfolge der Knoten im Treewalk eines Kürzesten-Wege-Baumes . . . 9
4.2 Der Beispielgraph für die Konstruktion von D_T 10
4.3 Der Hilfsgraph D_T 10
4.4 Zum Beweis von Fall 1 des Lemmas 4.2 12
4.5 Zum Beweis von Fall 2 des Lemmas 4.2 12
4.6 Die zu a korrespondierende Kante e in Fall 3 13
4.7 Knoten x wird vom Äußeren endeckt 13
4.8 Der Face Vertex Graph als Repräsentation 15

Verzeichnis der Algorithmen

1 Berechnung der $C_j(w)$ 16
2 Zusammenfassende Darstellung 17

1 Einführung

Viele kombinatorische Optimierungsverfahren beruhen auf dem Prinzip des „Divide and Conquer“. Ein Problem wird in zwei oder mehrere kleinere Probleme geteilt. Das Verfahren wird dann rekursiv auf die Subprobleme angewandt und die Verbindung derer Lösungen löst auch das ursprüngliche Problem.
In Graphen möchte man vernünftig große Knotenteilmengen finden, die durch eine möglichst kleine Anzahl von Kanten miteinander verbunden sind. Man kann auch nach zwei gleich großen Knotenteilmengen suchen, zwischen denen nur wenige Kanten existieren. Beispielsweise gehören *Placing* und *Routing* aus VLSI zu diesen Problemen([MWW92]). Ein integrierter Schaltkreis wird zunächst in Teilschaltkreise aufgeteilt und dann werden die verbindenden Leitungen wieder eingefügt. Sowohl die Qualität und Größe des Layouts als auch die Effizienz der Programme hängt sehr stark von der Anzahl der wiedereingefügten Leitungen ab.
Kleine Schnitte, welche einen Graphen möglichst gleichmäßig teilen, sind also für Divide-and-Conquer-Anwendungen sehr wünschenswert. Kommen wir nun zur konkreten Problemstellung.

Zunächst sind noch einige grundlegende Definitionen notwendig. Als *Schnitt* in einem Graphen bezeichnet man Menge von Kanten bzw. Knoten, deren Entfernen den Graphen in 2 Komponenten teilt. Die *Balance eines Schnittes* ist das Verhältnis von Gewicht der kleineren Seite des Schnitts zum Gesamtgewicht und die *gewichteten Kosten (Quotient Costs)* sind das Verhältnis von den Kosten des Schnitts zum Gewicht der kleineren Seite.

Das Ziel ist das Finden von kleinen Schnitten mit hoher Balance. Dabei betrachten wir zwei Problemstellungen:

1. Finden des Schnitts, der die *gewichteten Kosten* minimiert
2. Finden des kleinsten Schnitts einer vorher festgelegten Balance b

Das Ergebnis dieser auf [Rao92] basierenden Ausarbeitung wird ein Algorithmus sein, welcher einen optimalen b-limitierten gewichteten Kantenschnitt in einem planaren Graphen für eine Balance $b \leq 1/3$ in $\mathcal{O}(n^2 \min(W, C))$ Zeit berechnet, wobei W und C Gesamtgewicht bzw. -kosten und n die Knotenanzahl des Graphen sind. Da ein optimaler b-limitierter gewichteter Schnitt maximal 3/2 der Kosten eines optimalen Schnittes besitzt, liefert der Algorithmus somit auch einen 1,5-fachen optimalen Kantenschnitt.

1.1 Wichtige Definitionen

Ein *Graph* $G = (V, E)$ besteht aus einer Knotenmenge V und einer Kantenmenge $E \subseteq V \times V$. Ein *planarer Graph* ist ein Grph, welcher sich ohne Kantenüberschneidungen in die Ebene einzeichnen läßt.

Auf dem Graphen $G = (V, E)$ definieren wir eine *Gewichtsfunktion* $w : V \to \mathbb{R}_+$ und eine *Kostenfunktion* $c : E \to \mathbb{R}_+$.

Ein *Kantenschnitt* $(S, \overline{S})$ ist eine Menge von Kanten mit folgenden Eigenschaften: Seien S und $\overline{S}$ zwei disjunkte Teilmengen von V mit $V = S \cup \overline{S}$. Dann liegt eine Kante $e = \{u, v\}$ im Schnitt $(S, \overline{S})$, wenn $u \in S, v \in \overline{S}$ oder $u \in \overline{S}, v \in S$.

Die *Kosten eines Kantenschnittes* ist die Summe der Kosten der Kanten in dem Schnitt:

$$c(S, \overline{S}) = \sum_{e \in (S, \overline{S})} c(e).$$

Die *Balance des Schnittes* $(S, \overline{S})$ ist das Verhältnis vom Gewicht der kleineren Seite des Schnitts zum Gesamtgewicht W:

$$b(S, \overline{S}) = \frac{\min(w(S), w(\overline{S}))}{W}.$$

Als *Gewichtete Kosten (Quotient Cost)* eines Schnittes $(S, \overline{S})$ bezeichnet man das Verhältnis von Kosten des Schnittes und Gewicht der kleineren Seite:

$$\frac{c(S, \overline{S})}{\min(w(S), w(\overline{S}))} \quad \text{bzw.} \quad \frac{c(S, \overline{S})}{W \cdot b(S, \overline{S})}.$$

Der *minimale gewichtete Schnitt (minimum Quotient Cut, Flux, Edge Expension)* ist ein Schnitt mit minimalen gewichteten Kosten. Die Berechnung eines solchen Schnittes in allgemeinen Graphen ist ein $\mathcal{NP}$-vollständiges Problem ([GJS76]).

Die *b-limitierten gewichteten Kosten* eines Schnittes C sind die gewichteten Kosten des Schnittes, falls C weniger als b-balanciert ist, andernfalls nehmen wir an, er wäre nur b-balanciert:

$$\frac{c(C)}{W \cdot \min(b, b(C))}.$$

2 Überlegungen zum Algorithmus

2.1 Kantenschnitte und Kreise

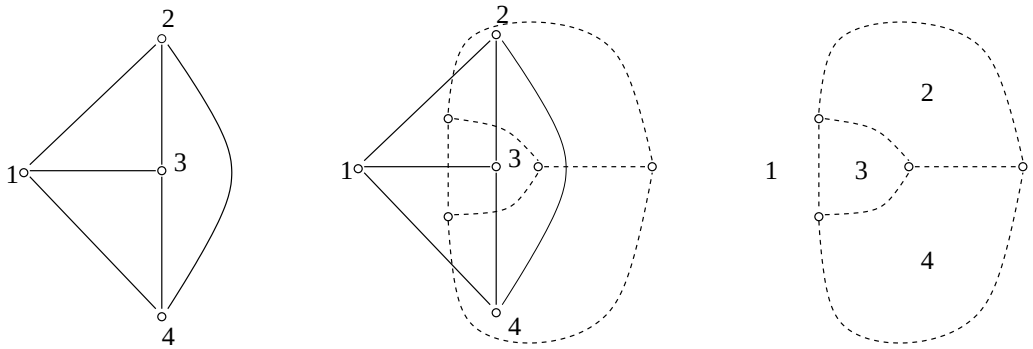
Der Algorithmus basiert auf dem *Dualgraphen* G^* des Originalgraphen G. Die Knoten in G^* korrespondieren zu den Facetten im Primalgraphen. Jede Kante e in G (und ihre Kosten) entsprechen der Kante zwischen zwei Knoten in G^*, welche zu den adjazenten Facetten der Kante e des Originalgraphen korrespondieren (siehe Abb. 2.1).
Also besitzt ein planarer Graph mit gewichteten Knoten einen Dualgraphen mit gewichteten Facetten. Ein Kreis im Dualgraphen teilt die Facetten und korrespondiert zu einem Kantenschnitt im Primalgraphen, welcher die entsprechenden Knoten in G trennt (Abb. 2.2).
Somit benötigen wir nur einen Algorithmus, der Kreisschnitte im Dualgraphen von G findet.

2.2 Connected Circuits und gewichtete Kosten

Zunächst wieder einige wichtige Definitionen:

Ein *Connected Circuit* ist eine Menge von Kreisen, verbunden durch eine Menge nichtzyklischer Pfade in einem planaren Graphen. Man kann den Connected Circuit als einfachen Kreis mit „zusammengedrückten Teilen“ auffassen, wie Abbildung 2.3 zeigt.

Die *äußere Facette* eines Connected Circuit C ist diejenige Facette, zu der alle Knoten und Kanten von C adjazent sind, wenn man C als planaren Graphen auffaßt.

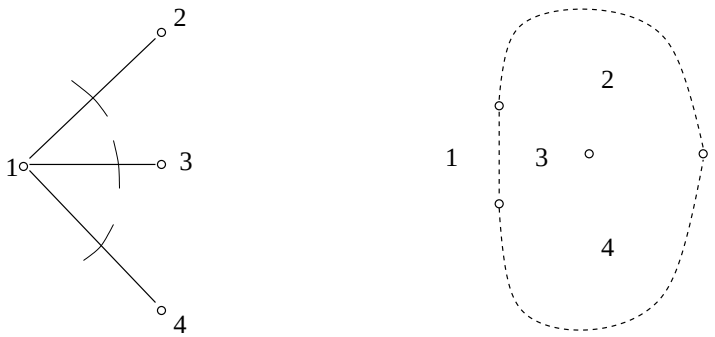


(a) Der Originalgraph G

(b) Konstruktion von G^*

(c) Der Dualgraph G^*

Abbildung 2.1: Konstruktion des Dualgraphen



(a) Ein Kantenschnitt

(b) Der korrespondierende Kreis

Abbildung 2.2: Ein Kantenschnitt entspricht einem Kreis im Dualgraphen

Das *Äußere (Exterior)* eines Connected Circuit sind alle Facetten und Kanten in der äußeren Facette.

Das *Innere (Interior)* eines Connected Circuit sind alle Facetten und Kanten, welche nicht im Äußeren oder auf dem Connected Circuit liegen. Abbildung 2.4 zeigt Inneres und Äußeres eines Circuit.

Die *Connected Circuit Kosten* eines Connected Circuit ist die Länge eines einfachen Kreises um die äußere Facette.

Der *Connected Circuit Cut* $(S, \overline{S})$ ist ein Schnitt, welcher zum Connected Circuit korrespondiert. Er hat S als Menge der Knoten im Originalgraphen, die den Facetten im Äußeren des Connected Circuit entsprechen.

Dabei ist zu beachten, dass die Connected Circuit Kosten sich im Allgemeinen von den realen Kosten des Schnittes unterscheiden, da die Kantenkosten der „zusammengedrückten Teile“ doppelt anstatt gar nicht gezählt werden.

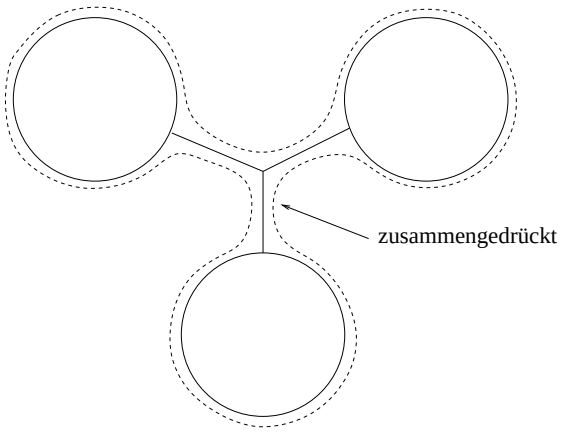


Abbildung 2.3: Ein Connected Circuit

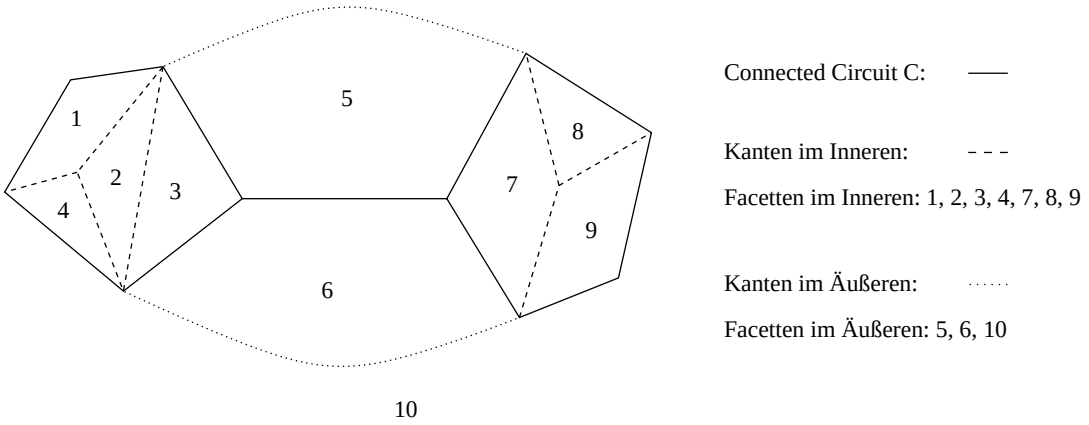


Abbildung 2.4: Inneres und Äußeres eines Connected Circuit

Ein Algorithmus, der optimale Connected Circuits in einem Graphen findet, löst somit das Kantenschnitt-Problem im Dualgraphen.

3 Vorbereitende Sätze

3.1 Ein optimaler *b*-limitierter gewichteter Schnitt mit 2 Komponenten

Lemma 3.1: *In jedem zusammenhängenden Graphen G existiert ein optimaler* b*-limitierter gewichteter Schnitt $(S, \overline{S})$, wobei die durch die Knotenmengen S und $\overline{S}$ induzierten Teilgraphen zusammenhängende Komponenten sind.*

Beweis:
Sei $C = (S, \overline{S})$ der optimale gewichtete Schnitt in einem Graphen G. Sei S o.B.d.A. eine Seite mit $k > 1$ Komponenten. Das Gesamtgewicht von S bzw. $\overline{S}$ sei $w(S)$ und $w(\overline{S})$. Es sei C_i die Menge der Kanten von C, die inzident zur i-ten Komponente von S sind. Das Gewicht der i-ten Komponente von S sei W_i.
Wir nehmen an, die m-te Komponente C_m von S ist diejenige mit dem minimalen Wert von

$\frac{c(C_i)}{W_i}$. Man betrachtet nun den Schnitt, welcher nur aus C_m besteht. Seine gewichteten Kosten sind nicht größer als die von C, da

$$\begin{aligned}\frac{c(C)}{\min(bW, w(S), w(\overline{S}))} &\geq \frac{\sum_{i=1}^{k} c(C_i)}{\sum_{i=1}^{k} W_i} \\ &\geq \frac{\sum_{i=1}^{k} W_i \left(\frac{c(C_i)}{W_i}\right)}{\sum_{i=1}^{k} W_i} \\ &\geq \left(\min_i \frac{c(C_i)}{W_i}\right) \frac{\sum_{i=1}^{k} W_i}{\sum_{i=1}^{k} W_i} = \frac{c(C_m)}{W_m}.\end{aligned}$$

Also ist C_m ein optimaler gewichteter Schnitt. $G' = (V, E \backslash C_m)$ enthält mindestens eine Komponente weniger, da alle anderen Komponenten von S mit einer Komponente von $\overline{S}$ verbunden sind, da der Graph zusammenhängend ist. Somit kann für jeden gewichteten Schnitt mit mehr als zwei Komponenten ein anderer Schnitt mit mindestens einer Komponente weniger erzeugt werden.
Es existiert also ein optimaler gewichteter Schnitt, welcher den Graphen in genau zwei Komponenten teilt.

□

Daraus folgt, dass ein optimaler gewichteter Schnitt existiert, welcher zu einem optimalen einfachen Kreis im dualen planaren Graphen korrespondiert.

Korollar 3.2: *Ein optimaler* b*-limitierter gewichteter Schnitt ist ein optimaler* b'*-balancierter Connected Circuit Cut mit* $b' \leq b$ *für ein* $b' \leq b$.

Mit diesem Korollar ist klar, das das Problem, einen optimalen b-limitierten gewichteten Schnitt zu finden, auf das Finden von b-balancierten Connected Circuit Cuts reduziert werden kann.

3.2 „Nur einmal kreuzen“

Wir betrachten nun einen planaren Graphen G, eine planare Einbettung von G und einen *gewurzelten aufspannenden Kürzesten-Wege-Baum* T von G, d.h. einen azyklischen Teilgraphen, welcher nur die kürzesten Wege von einer ausgezeichneten Wurzel zu jedem anderen Knoten beinhaltet. Es sei C ein Connected Circuit, der die Wurzel von T enthält.

Ein *Branch (Zweig)*, definiert durch einen Knoten v, in einem Kürzesten-Wege-Baum ist der eindeutige Pfad von v zur Wurzel des Baumes.

Ein Knoten v auf einem Connected Circuit C wird *von C's Innerem entdeckt*, wenn die zu v inzidente Kante im von v definierten Branch im Inneren von C liegt. Entsprechend können Knoten auf C *von C's Äußerem* oder *vom Circuit C entdeckt* werden, wenn die inzidente Kante des Branchs des betrachteten Knoten im Äußeren oder auf C liegt (Abb. 3.1).

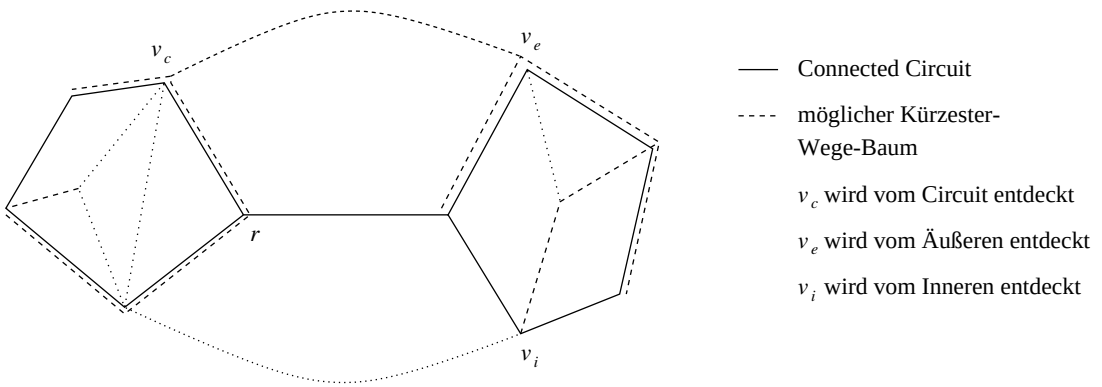


Abbildung 3.1: Ein Connected Circuit wird entdeckt

Satz 3.3: *Sei G ein planarer Graph, $b \leq 1/3$ und r ein Knoten, welcher auf einem optimalen Connected Circuit Cut C_{opt} liegt.*
Für jeden Kürzesten-Wege-Baum T von G mit Wurzel r existiert ein optimaler b*-balancierter Connected Circuit Cut C, der r enthält und jeder Knoten auf C wird von einem Branch von T entweder von C oder von C's Innerem entdeckt.*

Eine Beweisskizze findet man unter anderem in [Rao87].
Das heißt, es existiert ein optimaler b-balancierter Connected Circuit C, wo jeder Branch des Kürzesten-Wege-Baumes vom Inneren von C zum Äußeren von C höchstens einmal wechselt. Wenn ein Branch einmal außerhalb des Circuits ist, kann er nicht mehr zurückkehren.
Also benötigen wir nur einen Algorithmus, der den besten b-balancierten Connected Circuit Cut findet, der jeden Branch eines *gegebenen* Kürzesten-Wege-Baumes höchstens einmal kreuzt. Wir können den Algorithmus für jede mögliche Wurzel eines Kürzesten-Wege-Baumes laufen lassen und die beste Lösung ausgeben.

4 Der Algorithmus

4.1 Konstruktion des Hilfsgraphen

Wir nehmen an, dem Algorithmus wird eine planare Einbettung von G und ein Knoten r, der auf einem optimalen Connected Circuit Cut liegt, gegeben. Wir entfernen dieser Annahme durch die Anwendung des Algorithmus für jeden Knoten und Wählen der besten Lösung.
Eine der Facetten, die zu r adjazent sind, muß im Äußeren des optimalen Connected Circuit liegen, da r auf C liegt. Wir nehmen an, F sei diese Facette. Die Beseitigung der Annahme erfolgt durch Benutzung des Algorithmus für alle f_r adjazenten Facetten.

Wir definieren nun folgenden *Treewalk (Baumdurchlauf):*

1. Starte mit r und gehe die entgegen dem Uhrzeigersinn erste zu r und F inzidente Baumkante entlang.

2. Von einem Knoten u nimm die Baumkante $e = \{u, v\}$, welche entgegen dem Uhrzeigersinn am nächsten zu der Kante liegt, die benutzt wurde um zu u zu gelangen.

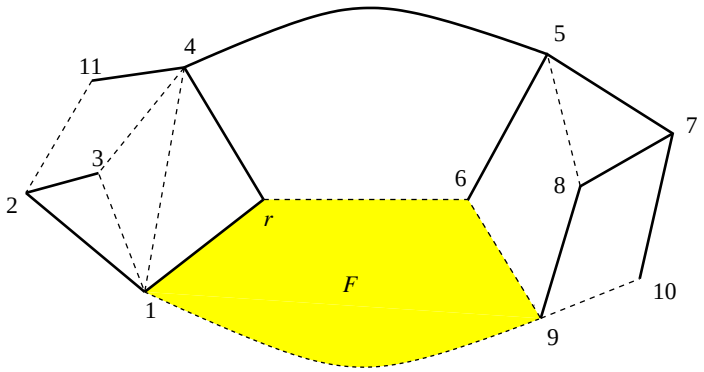


Abbildung 4.1: Die Reihenfolge der Knoten im Treewalk eines Kürzesten-Wege-Baumes

Abbildung 4.1 zeigt ein Beispiel eines Treewalks. Jede Baumkante wird genau zweimal benutzt, einmal in jede Richtung.
Wir sagen, *Knoten u ist an Position i des Treewalks*, wenn u der Knoten ist, der nach $i-1$ Kanten erreicht wird. Interne Knoten des Baumes kommen mehrmals vor, Blätter genau einmal. Der Treewalk *trifft auf eine Nichtbaumkante $e=\{u,v\}$ in u an Position i*, wenn u an Position i im Treewalk ist und e entgegen dem Uhrzeigersinn zwischen der Kante, die benutzt wurde, um zu u zu gelangen und der Kante, die u im Treewalk verläßt, liegt. Das heißt, jedes Ende einer Nichtbaumkante wird einmal vom Treewalk getroffen, da jede Nichtbaumkante entweder entgegen dem Uhrzeigersinn zwischen genau einem Paar von Baumkanten, welche inzident zu einem Knoten sind, liegt oder inzident zu einem Blatt des Baumes ist.

Basierend auf dem Treewalk wird ein gerichteter azyklischer Graph $D_T=(N_T, A_T^1 \cup A_T^2 \cup A_T^3)$ definiert:

$$
\begin{aligned}
N_T &= \{1,\ldots,2n-1\} \\
A_T^1 &= \Big\{(i,i+1) \mid i \in \{1,\ldots,2n-2\}\Big\} \\
A_T^2 &= \left\{ (i,j) \;\middle|\; i \le j, \begin{array}{l}\text{Nichtbaumkante } e \text{ wird} \\ \text{an Position } i \text{ und } j \\ \text{im Treewalk getroffen}\end{array} \right\} \\
A_T^3 &= \left\{ (i,j) \;\middle|\; i < j, \begin{array}{l}\text{Knoten } u \text{ ist nachein-} \\ \text{ander an Position } i \\ \text{und } j \text{ im Treewalk}\end{array} \right\}
\end{aligned}
$$

A_T^1 entspricht dem Entlanggehen auf den Baumkanten, A_T^2 den Nichtbaumkanten in einer Richtung und A_T^3 verbindet hintereinander liegende Positionen im Treewalk, die zu einem einzelnen Knoten in G korrespondieren.
Wir müssen nun noch die Gewichts- und die Kostenfunktion geeignet abändern:

Erweiterung der Kostenfunktion $c: E \to \mathbb{R}_+$:

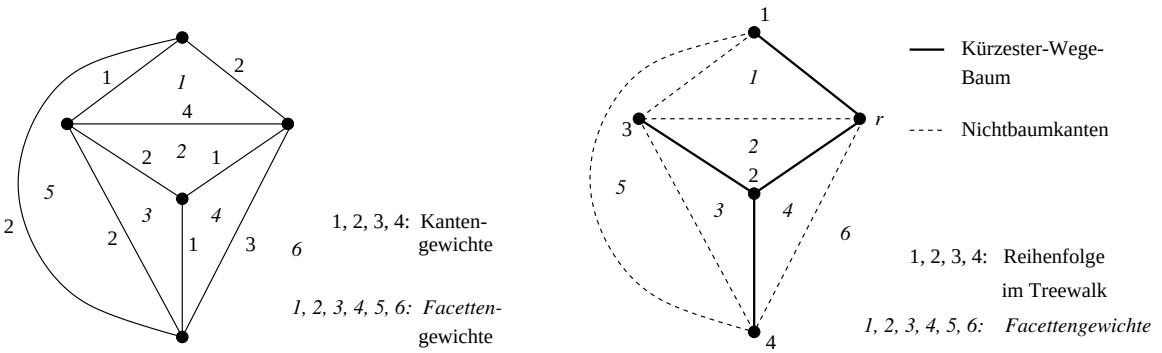
$$
\begin{aligned}
a=(i,j) \in A_T^1 &: c(a)=c(e) \text{ , } e \text{ ist } i\text{-te Kante im Treewalk} \\
a \in A_T^2 &: c(a)=c(e) \text{ , } e \text{ ist korrespondierende Kante laut Def. von } D_T \\
a \in A_T^3 &: c(a)=0
\end{aligned}
$$

Erweiterung der Gewichtsfunktion $w : V \to \mathbb{R}_+$:
Das Gewicht $w(a)$ eines Bogen a ist das Gewicht im Kreis, der durch T und die Kante e in G, die zu a korrespondiert, entsteht. Also ist $w(a) = 0$ für alle $a \in A_T^1 \cup A_T^3$. Das Gewicht im von einer Nichtbaumkante induzierten Kreis ist das auf der Seite, die nicht die äußere Facette F enthält.

Wie die Konstruktion des Digraphen D_T beispielhaft aussieht, zeigen Abbildung 4.2 und 4.3.

Zusammenfassend kann gesagt werden, dass

- ein Pfad σ in D_T von 1 bis $2n-1$ zu einem Entlanggehen eines Connected Circuit C in G korrespondiert, welcher in r startet und auch endet,
- jeder Bogen in σ einer entlanggegangenen Kante oder einem einzelnen Knoten in G entspricht,
- das Gewicht des Pfades das Gewicht des Inneren des Connected Circuit C ist.



(a) Der Dualgraph G des Originalgraphen

(b) der Kürzeste-Wege-Baum mit einer möglichen Wurzel r

Abbildung 4.2: Der Beispielgraph für die Konstruktion von D_T

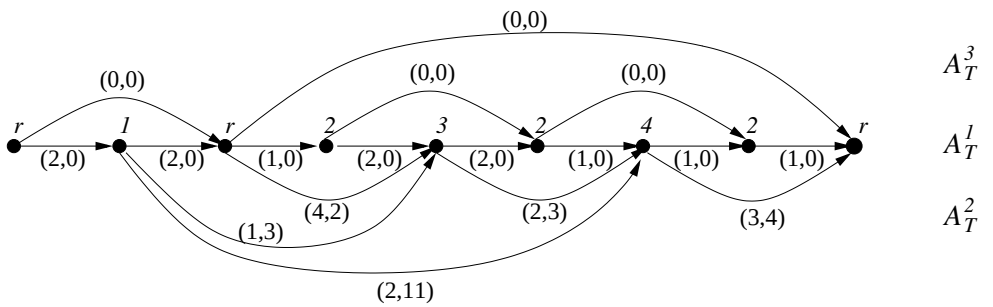


Abbildung 4.3: Der Hilfsgraph D_T

Nicht alle Connected Circuits in G können durch einfache Pfade in D_T repräsentiert werden, eine große Teilmenge jedoch schon. Wir müssen uns also im Folgenden auf diese Connected Circuits beschränken.

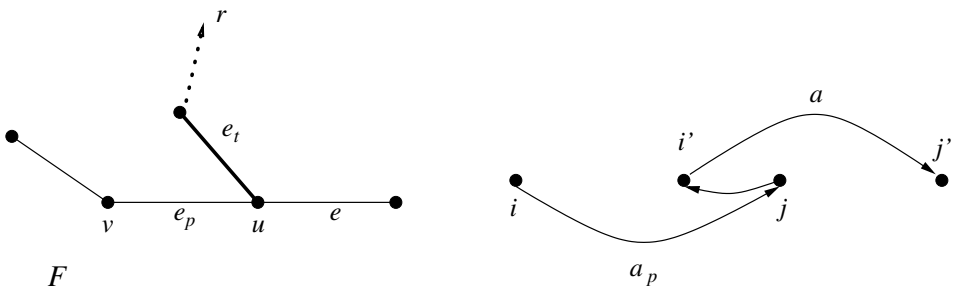
Definition: *Ein Connected Circuit hat die Eigenschaft (*), wenn er r enthält, F eine äußere Facette ist und vom Kürzesten-Wege-Baum immer vom Inneren oder vom Circuit entdeckt wird.*

Lemma 4.1: *Sei C ein Connected Circuit mit der Eigenschaft (*). Das Entlanggehen von C entgegen dem Uhrzeigersinn, welches an r startet und endet, korrespondiert zu einem einfachen Pfad σ in D_T von Knoten 1 bis $2n-1$.*
Das Gewicht des Pfades $\sum_{a\in\sigma} w(a)$ ist das vom Connected Circuit C eingeschlossene Gewicht.

Beweis:
Wir nehmen an, der erste Teil des Lemmas gilt und betrachten eine „kreuzende" Nichtbaumkante e, welche vom Connected Circuit C benutzt wird. Der erste Teil des Lemmas stellt sicher, dass keine Nichtbaumkante im Inneren des induzierten Kreises von e und dem kürzesten-Wege-Baum benutzt wird, da D_T entsprechend konstruiert wurde. Also ist das gesamte Innere des Kreises im Inneren von C.
Weiterhin können nur Nichtbaumkanten Gewicht von der Facette F „abschneiden". Also besteht das Gewicht des Inneren von C aus der Summe der Gewichte der Nichtbaumkanten, die von C benutzt werden. Somit ist das Gewicht des Pfades das eingeschlossene Gewicht des Connected Circuit.
Für den Beweis des ersten Teils des Lemmas betrachten wir einen Connected Circuit C mit Eigenschaft (*) und ein Entlanggehen gegen den Uhrzeigersinn von C.
Da das Entlanggehen an r startet und endet, 1 und $2n-1$ mit r korrespondieren und Bögen mit $c(a)=0$ zwischen allen anderen Knoten, die r entsprechen, existieren, entspricht irgendein Pfad von 1 nach $2n-1$ dem Entlanggehen von C.
Also gilt das Lemma genau dann nicht, wenn dieser Pfad einen Bogen in der falschen Richtung benutzt. Dass dies nicht passiert, zeigt das folgende Lemma.

Lemma 4.2: *Beim Entlanggehen entgegen dem Uhrzeigersinn eines Connected Circuit ist die äußere Facette immer zur Rechten des Weges, d.h. entgegen dem Uhrzeigersinn zwischen zwei aufeinanderfolgenden Kanten.*

Beweis:
Fall 1: Ein Bogen $a \in A_T^3$ wird in falscher Richtung benutzt.
Wir nehmen an, dies passiert an Knoten u. Sei $e_p = \{v, u\}$ die Kante, die u erreicht und e die Kante, die u beim Entlanggehen von C verläßt.
Die Kanten e_p und e korrespondieren somit zu den Bögen $a_p = (i, j)$ und $a = (i', j')$ in D_T, wobei $j > i'$. Die zu u inzidente Kante e_t seines Branches bzgl. T ist zur Linken des Entlanggehens von C, d.h. e_p liegt entgegen dem Uhrzeigersinn von e_t und im Uhrzeigersinn von e, oder e_p ist die u entdeckende Baumkante e_t (Abb. 4.4).
Die Kante e_t kann nicht zur Rechten liegen, da sonst u vom Äußeren entdeckt würde, was der Annahme widerspäche, C hätte die Eigenschaft (*).
Nach Definition des Treewalk muß e_p vor e im Baumdurchlauf stehen. Somit kann a_p nicht mit einer höheren Treewalkposition verbunden sein als a (Definition von D_T).

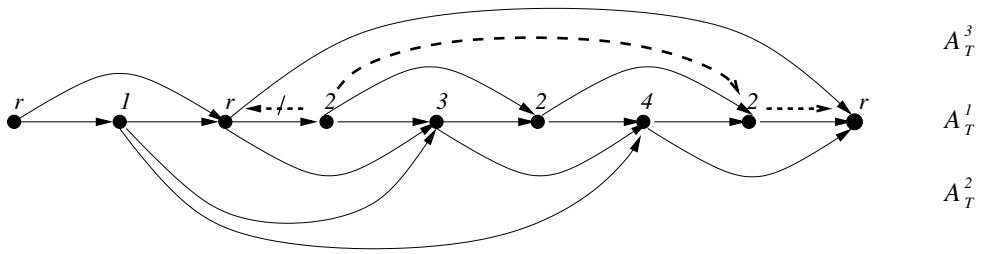


(a) Der Connected Circuit C (b) Der korrepondierende Graph D_T

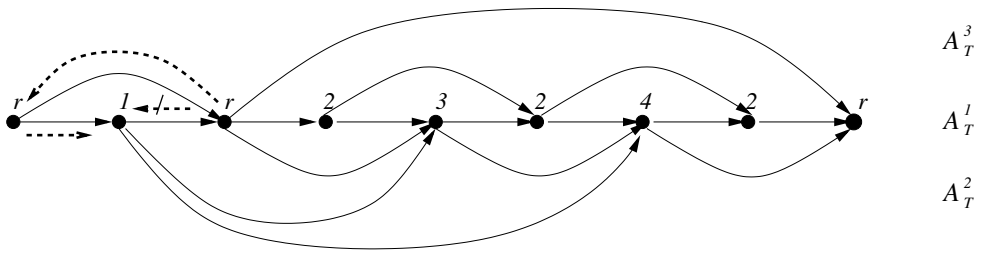
Abbildung 4.4: Zum Beweis von Fall 1 des Lemmas 4.2

Daraus folgt somit $j \leq i'$, was ein Widerspruch zur Voraussetzung darstellt.

Fall 2: Ein Bogen $a \in A_T^1$ wird in falscher Richtung benutzt.
Ein Bogen aus A_T^1 existiert für jede Richtung einer Baumkante. Anstatt den Bogen a in der falschen Richtung zu benutzen, wenn die a entsprechende Kante e im Entlanggehen des Connected Circuit benutzt wird, benutze Bögen aus A_T^3, die zum anderen zu e korrespondierenden Bogen führen und benutze diesen in der richtigen Richtung (Abb. 4.5(a)).
Dabei wird der Bogen aus A_T^3 möglicherweise in der falschen Richtung benutzt (Abbildung 4.5(b)). Dies ist jedoch nach Fall 1 ausgeschlossen (Widerspruch).



(a) Benutzung des äquivalenten Bogens in der richtigen Richtung



(b) Ein Bogen aus A_T^3 wird in der falschen Richtung benutzt

Abbildung 4.5: Zum Beweis von Fall 2 des Lemmas 4.2

Fall 3: Ein Bogen $a \in A_T^2$ wird in falscher Richtung benutzt.
Sei $e = \{u, v\}$ die Kante, welche dem durch das Entlanggehen von C in der falschen Richtung benutzten Bogen $a = (i, j)$ entspricht. Der Knoten u korrespondiert mit der Position j und v mit Position i. Sei p_b der Pfad im Entlanggehen von C bis zu dem Punkt, wo e benutzt wird (von r bis u). Der Rest von u zurück zu r sei Pfad p_r (siehe Abb. 4.6).
Die vor e im Entlanggehen von C benutzte Kante e_p muß entweder die Baumkante e_t sein, die u entdeckt oder sie ist eine Nichtbaumkante entgegen dem Uhrzeigersinn von e_t und im Uhrzeigersinn von e (analog zu Fall 1).

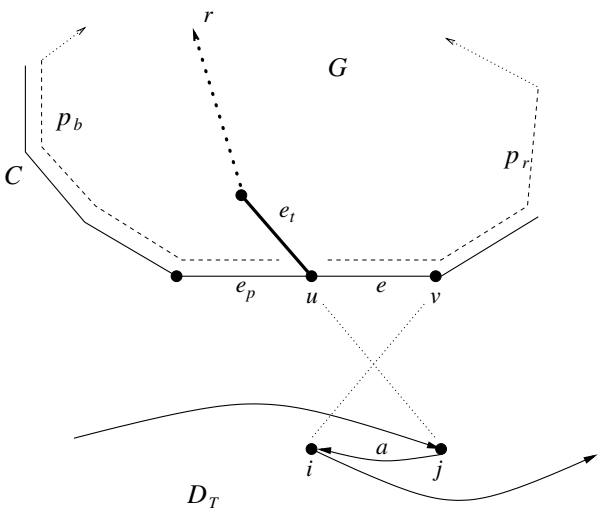


Abbildung 4.6: Die zu a korrespondierende Kante e in Fall 3

Da Knoten v im Treewalk vor Knoten u liegt ($i < j$), muß aufgrund der Planarität irgendein Vorgängerknoten von v im Baum auf p_b liegen. Außerdem liegt der Branch, der zu v führt, auf der linken Seite von p_b. Durch die Planarität des Graphen G wird jeder Pfad, der an v startet und entweder nur Vorgängerkanten im Baum oder Nichtbaumkanten, welche sich im Uhrzeigersinn von den Vorgängerkanten und entgegen dem Uhrzeigersinn von der eingehenden Kante befinden, benutzt, möglicherweise p_b von links schneiden.

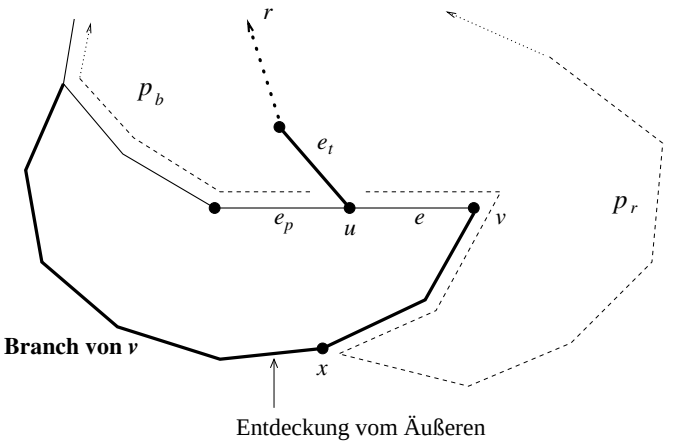


Abbildung 4.7: Knoten x wird vom Äußeren endeckt

Somit müßte ebenfalls p_r auf diese Weise p_b schneiden, falls er nur solche Kanten benutzt. Dies würde bedeuten, dass man p_r vor p_b entlangginge und wäre somit ein Widerspruch.
Also muß p_r an einem Knoten x eine Kante benutzen, welche sich entgegen dem Uhrzeigersinn von der Vorgängerkante von x und im Uhrzeigersinn von der zu x führenden Kante von p_r befindet (siehe Abb. 4.7). Dies bedeutet, dass sich die Vorgängerkante im Baum von x auf der rechten Seite von p_r befindet. Dies ist laut Voraussetzung das Äußere des Connected Circuit C. Also entdeckt der Branch des Baumes Knoten x vom Äußeren, was einen Widerspruch zu der Voraussetzung darstellt, C hätte die Eigenschaft (*).
Somit wird bei einem Entlanggehen eines Connected Circuit C mit der Eigenschaft (*) gegen den Uhrzeigersinn im korrespondierenden Pfad in D_T kein Bogen in der falschen Richtung benutzt.

□

4.2 Ein gerichtetes-Pfad-Problem

Nach Lemma 4.1 kann das Connected-Circuit-Cut-Problem auf $\mathcal{O}(n)$ der folgenden gerichteten-Pfad-Probleme reduziert werden:

Weighted minimum length directed path problem

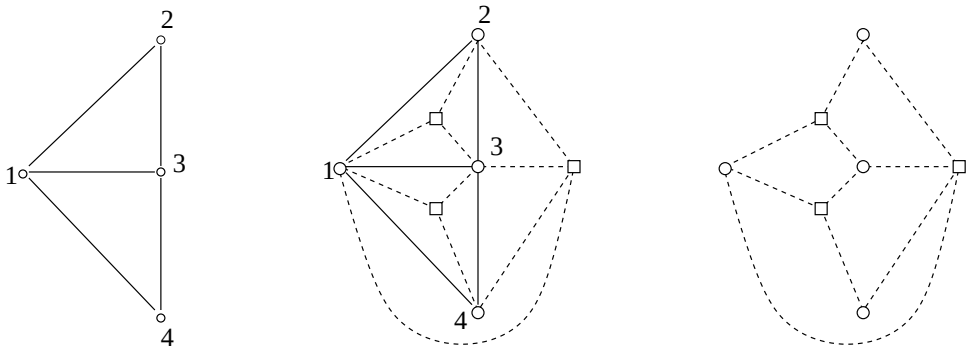
gegeben: DAG D mit Bogengewichten $w : A \to \mathbb{R}_+$ und Bogenkosten $c : A \to \mathbb{R}_+$, 2 Knoten $i, j \in D$

Lösung: für jedes Gewicht $w \in \{1, \ldots, \lceil W/3 \rceil\}$ ein Pfad p von i nach j mit minimalen Kosten und $w \leq w(p) \leq W - w$

Unser Schnitt-Problem kann auf dieses Pfad-Problem folgendermaßen reduziert werden:
Für jede Wurzel r eines Kürzesten-Wege-Baumes und adjazente Facette F wird D_T definiert. Nach Satz 3.3 ist für eine Wurzel r ein Connected Circuit mit Eigenschaft (*) ein optimaler Connected Circuit. Nach Lemma 4.1 korrespondieren alle Connected Circuits mit der Eigenschaft (*) für eine Wurzel r eines Kürzesten-Wege-Baumes T und der adjazenten Facette F zu Pfaden in dem gerichteten Graphen D_T bzgl. T und F.
Also wird für jede Balance w ein optimaler Connected Circuit Cut zu einem optimalen gewichteten kürzesten Pfad in einem bestimmten D_T für die Knoten 1 und $2n - 1$ korrespondieren.

Der dann vorgestellte Algorithmus löst das gewichtete Pfad Problem in $\mathcal{O}(nW)$ Zeit. Durch Modifikationen kann der Aufwand auf $\mathcal{O}(nC)$ bzw. $\mathcal{O}(nC')$ gesenkt werden, wobei C' die Kosten der optimalen Lösung sind.
Der Algorithmus muß für jede Kombination von r und adjazenter Facette F laufen. Diese Paare können durch den Face Vertex Graph des Graphen G repräsentiert werden. Dieser planare Graph besitzt Knoten für jeden Originalknoten und jede Facette sowie Kanten zwischen jedem Facettenknoten und seinen adjazenten Originalknoten, wie auch Abbildung 4.8 zeigt.
Der Graph hat also $|V| + f$ Knoten, wobei f die Anzahl der Facetten in der Einbettung des Originalgraphen G ist. Die Anzahl der Kanten kann nach der *Eulerschen Formel* durch $\mathcal{O}(|V| + f) = \mathcal{O}(n)$ abgeschätzt werden.

Insgesamt ergibt sich für das Berechnen optimaler Connected Circuit Cuts also ein Aufwand von $\mathcal{O}(n \cdot nW)$ oder $\mathcal{O}(n \cdot nC)$, also $\mathcal{O}(n^2 \min(W, C))$.
Nach Korollar 3.2 können wir auch optimale b-limitierte gewichtete Schnitte für jedes $b \leq 1/3$



(a) Der Originalgraph G

(b) Konstruktion Face Vertex Graph

(c) Der Face Vertex Graph

Abbildung 4.8: Der Face Vertex Graph als Repräsentation

durch einfaches Berechnen der gewichteten Kosten der optimalen b'-balancierten Connected Circuit Cuts für jedes $b' \leq b$ erhalten. Die Berechnung aller möglichen Werte von b braucht $\mathcal{O}(\min(W, C))$ Zeit. Also beträgt der Aufwand zum Finden von b-limitierten gewichteten Schnitten für jedes $b \leq 1/3$ ebenfalls $\mathcal{O}(n^2 \min(W, C))$.

4.3 Lösung des gerichteten-Pfad-Problems

Wir definieren $C_j(w)$ für jedes $j \in \{1, \ldots, 2n-1\}$ als die Kosten des besten gerichteten Pfades von 1 bis j mit dem Gewicht w. Wir nehmen dabei an, daß die Knoten in dem Digraphen topologisch sortiert sind, d.h. es existiert kein Pfad von i nach j mit $i > j$. Die $C_j(w)$ können durch folgende rekursive Gleichung berechnet werden:

$$C_j(w) = \min\Big(\min_{a=(i,j)\in D_T} C_i(w - w(a)) + c(a), C_j(w)\Big).$$

Diese Gleichung arbeitet korrekt, da jeder Pfad zu j einen seiner Vorgängerknoten benutzen muß, und dieser muß bis zu dem Punkt optimal sein. Somit können die $C_j(w)$'s nacheinander berechnet werden, da zur Berechnung von $C_j(w)$ nur die Werte der $C_i(w)$ benötigt werden mit $i < j$. Algorithmus 1 zeigt eine solche sequentielle Berechnung.

Jeder Bogen wird für jedes Gewicht nur einmal benutzt (beim Knoten am Kopf des Bogens). Somit liegt der Aufwand hierfür bei $\mathcal{O}(|A| \cdot W)$.
Jeder Knoten wird für jedes Gewicht einmal benutzt, also ein Aufwand von $\mathcal{O}(|V| \cdot W)$.
Ingesamt ergibt sich hiermit $\mathcal{O}((|A| + |V|) \cdot W)$, und da $|A| = \mathcal{O}(|E|) = \mathcal{O}(|V|)$ gilt, benötigt die Berechnung des gewichteten kürzesten gerichteten Pfades $\mathcal{O}(nW)$ Zeit.

Der Algorithmus kann so modifiziert werden, so dass er $\mathcal{O}(nC)$ Zeit benötigt. Wir berechnen für alle möglichen Kosten Kürzeste Pfade von 1 bis i unter bestimmten Gewichtseinschränkungen. Ich kann jedoch nicht näher auf diese Veränderungen eingehen, da [Rao92] diesen Punkt nicht weiter ausführt.

Algorithmus 1: Berechnung der $C_j(w)$

Data : DAG $D_T, V = \{1, \ldots, 2n-1\}$, Knoten topologisch geordnet
Result: $\forall\ w \in \{1, \ldots, \lceil W/3 \rceil\}$ Kosten eines minimaler Pfades p von 1 nach $2n-1$ mit $w \leq w(p) \leq W - w$

```
begin
    Initialisierung
    C_i(w) = +∞   ∀ i ∈ {1,...,2n-1},   ∀ w ∈ {0,...,W}
    C_1(0) = 0
    Berechnung der C_j
    for j = 2 to 2n-1 do
        for w = 0 to W do
            foreach a = (i,j) ∈ D_T do
                if w - w(a) < 0 then
                    continue
                if C_i(w - w(a)) + c(a) < C_j(w) then
                    C_j(w) := C_i(w - w(a)) + c(a)
    Updaten der C_{2n-1}, falls bessere Balance günstiger
    for w = ⌊W/2⌋ to 1 do
        if C_{2n-1}(w) > C_{2n-1}(w+1) then
            C_{2n-1}(w) := C_{2n-1}(w+1)
        if C_{2n-1}(w) > C_{2n-1}(W-w) then
            C_{2n-1}(w) := C_{2n-1}(W-w)
    Ausgabe der C_{2n-1}(w)   ∀ w ∈ {1,...,⌈W/3⌉}
end
```

4.4 Eine Abwandlung des Pfad-Problems

Minimum ratio directed path problem

gegeben: DAG D mit Bogengewichten $w : A \to \mathbb{R}_+$ und Bogenkosten $c : A \to \mathbb{R}_+$, 2 Knoten $i, j \in D$

Lösung: ein Pfad p von i nach j, welcher $\frac{\sum_{a \in p} c(a)}{\sum_{a \in p} w(a)}$ minimiert

Hiermit kann man die optimalen gwichteten Schnitte bis zum Faktor 3,5 durch Lösen von $\mathcal{O}(n)$ der oben genannten Pfadprobleme abschätzen.

Seien G der planare Graph T ein Kürzester-Wege-Baum mit Wurzel r und F eine zu r adjazente Facette. Der damit assoziierte Digraph sei D_T. Ein Bogen $a \in D_T$ ist *schwer*, wenn $w(a) \geq W/7$. Nach Entfernen aller schweren Bögen aus D_T löst man das ratio path problem im entstandenen Graphen für die Knoten 1 und $2n-1$. Diese Prozedur vollzieht man für alle $\mathcal{O}(n)$ möglichen D_T. Dies kann in $\mathcal{O}(n \min(n, \log WC))$ Zeit berechnet werden ([Meg79]).
Nun sei Q_e das Minimum der gewichteten Kosten für jeden von einer schweren Kante e mit T induzierten Kreis. Das Minimum über die berechneten Gewichtete-Kosten-Pfade sei Q_r.

Das folgende Lemma schätzt die optimalen gewichteten Kosten Q_{opt} bzgl. dieser Grenzen ab:

Lemma 4.3:

$$\min\left(\frac{2}{7}Q_e, \frac{2}{3}Q_r\right) \leq Q_{opt}$$

Eine zusammenfassende Darstellung der Vorgehensweise zum Finden minimaler gewichteter Schnitte zeigt Algorithmus 2.

Algorithmus 2: Zusammenfassende Darstellung

```
Data  : Ein planarer Graph G
Result: Ein minimaler b-limitierter gewichteter Schnitt mit einem b ≤ 1/3
begin
    Konstruktion des Dualgraphen G* zu G
    foreach Knoten r ∈ G* do
        Suche eines Kürzesten-Wege-Baumes T mit Wurzel r
        foreach Facette F adjazent zu r do
            Konstruktion des Hilfsgraphen D_T bzgl. r und F
            Lösen des Pfadproblems für D_T
    Auswahl der besten Lösung
end
```

5 Schnelle Approximationen gewichteter Schnitte

Satz 5.1: *Gegeben sei ein planarer Graph mit n Knoten. Ein $\mathcal{O}(k)$-facher optimaler gewichteter Schnitt kann in $\mathcal{O}(n^{1+1/k}(\min(n, \log WC))\log n \log C)$ Zeit berechnet werden.*

Bei den früheren Algorithmen haben wir eine Algorithmus A entwickelt, welcher auf gewurzelten Kürzesten-Wege-Bäumen basierte. Der Algorithmus lief dann für $\mathcal{O}(n)$ solcher Bäume. Durch die Benutzung der Graphendekomposition-Technik von Awerbuch, Berger, Cowen und Peleg [ABCP] läuft der Algorithmus nur auf einer kleinen Teilmenge von Bäumen. Algorithmus A muß somit nur noch $\mathcal{O}(n^{1/k}\log n \log C)$-mal benutzt werden. Durch die Benutzung des auf dem letztgenannten Pfadproblem basierenden Algorithmus erhalten wir die oben genannten Abschätzungen.

Die Laufzeit liegt bei $\mathcal{O}(n(\min(n, \log WC))\log n \log C)$, wenn man in diesem Satz k auf $\log n$ setzt. Wenn W und C polynomial in n sind, liegt der Aufwand bei $\mathcal{O}(n \log^3 n)$. Dies ist eine deutliche Verbesserung gegenüber $\mathcal{O}(n^2 \min(W, C))$, welcher zum exakten Berechnen eines optimalen b-limitierten gewichteten Schnittes benötigt wird.

Literatur

[ABCP] B. Awerbuch, B. Berger, L. Cowen, D. Peleg: *Fast constructions of sparse neighbourhood covers*, Unpublished Manuscript

[GJS76] M.R. Garey, D.S. Johnson, L. Stockmeyer: *Some simplified $\mathcal{NP}$-complete graph problems*, Theoretical Computer Science, Seiten 237-267, 1976

[MWW92] R.H. Möhring, D. Wagner, F. Wagner: *VLSI Network Design*, Technical Report 323-1992, Technische Universität Berlin, 1992

[Meg79] N. Meggido: *Combinatorial optimization with rational objective functions*, Mathematics of Operations Research, 4(4):414-424, 1979

[Rao87] Satish B. Rao: *Finding Near Optimal Seperators in Planar Graphs*, Proceedings 28th Annual Symposium on Foundations of Computer Science, Seiten 225-237, 1987

[Rao92] Satish B. Rao: *Faster Algorithms for Finding Small Edge Cuts in Planar Graphs*, 24th Annual ACM Symposium on the Theory of Computing, Seiten 229-240, 1992