Humboldt - Universität zu Berlin

Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

Zwei approximative Algorithmen für das Matchingproblem in gewichteten Graphen

Diplomarbeit
im Studiengang Informatik

vorgelegt von
Sven Hanke

unter der Betreuung von
Herrn Prof. Dr. S. Hougardy

11. Oktober 2004

Zusammenfassung

In dieser Diplomarbeit werden zwei approximative Algorithmen zum Matchingproblem für gewichtete Graphen vorgestellt. Der `Greedy-`$\mathcal{A}_5$ erreicht eine Güte von $2/3 - \epsilon$ in einer Laufzeit von $\mathcal{O}(m + n)$, bzw. in Abhängigkeit der Güte: $\mathcal{O}((m+n+\frac{1}{\epsilon}\log\frac{n}{\epsilon})\cdot\log\frac{1}{\epsilon})$. Der `Greedy-`$\mathcal{A}_7$ ist eine Erweiterung von `Greedy-`$\mathcal{A}_5$ und erreicht eine Güte von $3/4 - \epsilon$ in einer Laufzeit von $\mathcal{O}((m+n)\log n)$, bzw. in Abhängigkeit der Güte: $\mathcal{O}((m+n)\cdot\frac{1}{\epsilon}\log\frac{n}{\epsilon}\log\frac{1}{\epsilon})$.

Selbständigkeits- und Einverständniserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Ich erkläre hiermit mein Einverständnis, dass die vorliegende Arbeit in der Bibliothek des Instituts für Informatik der Humboldt-Universität zu Berlin ausgestellt werden darf.

Berlin, den 11. Oktober 2004

.....
(Unterschrift)

Danksagung

An dieser Stelle möchte ich mich bei meinem Betreuer Prof. Dr. Stefan Hougardy für die Begleitung meiner Diplomphase bedanken, ohne dessen fachkundige Beratung diese Arbeit nicht zustande gekommen wäre. Weiterhin sei ihm und Dr. Amin Coja-Oghlan für das Korrekturlesen und die vielen Hinweise gedankt.

Besonderer Dank gilt meinen Eltern für ihre tatkräftige und finanzielle Unterstützung meiner Studienzeit. Ebenso möchte ich mich bei meiner Frau Conny für ihr Verständnis und ihren Beistand bedanken, wodurch mir die Arbeit um vieles erleichtert wurde.

Widmen möchte ich meine Diplomarbeit unserem noch nicht aber bald geborenen Sohn in Vorfreude auf einen neuen privaten und beruflichen Lebensabschnitt.

Inhaltsverzeichnis

1 Einleitung **1**

2 Grundlagen **4**
2.1 Terminologie 4
2.2 Überdeckung von $\mathcal{A}^*_\infty$ 6
2.3 Partitionen 11

3 Der Greedy-$\mathcal{A}_5$-Algorithmus **12**
3.1 Die Initialisierung 13
3.1.1 Die Partition der C_4-Kreise 13
3.1.2 Die Partition der Arme 13
3.1.3 Die Partition der $\mathcal{A}_5$-Pfade 13
3.2 Der Algorithmus 16
3.2.1 Die Laufzeit 16
3.2.2 Die Korrektheit 18
3.3 Die Approximation von $(2/3 - \epsilon)\,\omega(M^*)$ 21

4 Der Greedy-$\mathcal{A}_7$-Algorithmus **23**
4.1 Die Initialisierung 24
4.1.1 Die Sortierung der C_4-Kreise 24
4.1.2 Die Sortierung der Arme 24
4.1.3 Die Partition der Erweiterungen 25
4.1.4 Die Erweiterbarkeit der Arme 25
4.1.5 Die Partition der $\mathcal{A}_5$-Pfade 25
4.2 Der Algorithmus 31
4.2.1 Die Laufzeit 32
4.2.2 Die Korrektheit 34
4.3 Die Approximation von $(3/4 - \epsilon)\,\omega(M^*)$ 36

5 Empirische Evaluation **38**
5.1 Das Start-Matching 38
5.2 Der `Greedy`-$\mathcal{A}_5$ im Vergleich zu Drake & Hougardy 38
5.3 Sinnvolle Iterationswerte für den `Greedy`-$\mathcal{A}_5$ 42
5.4 Der `Greedy`-$\mathcal{A}_7$ im Vergleich 46
5.5 Zusammenfassung 49

Literatur **53**

Kapitel 1

Einleitung

Das *Matchingproblem* ist ein fundamentales Thema in der Graphentheorie. Eine Teilmenge der Kanten eines Graphen wird als *Matching* bezeichnet, wenn sich keine zwei dieser Kanten in einem Endpunkt berühren. Berührt jede Kante mindestens eine Matchingkante auf einem Endpunkt, so wird das Matching *maximal* genannt. Gehört jeder Knoten zu einer Matchingkante, so heißt das Matching *perfekt*.

In ungewichteten Graphen wird die Suche nach einem kardinalitätsgrößten Matching *Maximum Matching Problem* bzw. *Maximum Cardinality Matching Problem* genannt. Im gewichteten Fall, dem *Matching Problem für gewichtete Graphen* bzw. *Maximum Weighted Matching Problem*, wird ein Matching gesucht, welches bezüglich seinem Gesamtgewicht maximal ist. Neben dieser groben Einteilung gibt es mehrere Varianten und Abstraktionen dieses Problems, über deren Existenz und Eigenschaften viel geforscht wurde. Als Beispiel werden einige aufgeführt:

Minimum Maximal Matching [41]
Minimum Weighted Maximum Cardinality Matching
Minimum Weighted Cardinality-k Matching
Minimum/Maximum Weighted Perfect Matching [4, 40]/[26]
Maximum Vertex-Weighted Matching [3]
Maximum Induced Matching [42]
Online Weighted Matching [20]
Maximum Degree-Constrained Subgraph (b-Matching) [18, 30]
Matching auf bipartiten Graphen [18] oder andere Graphen-Beschränkungen
Distributed Weighted Matching (Parallele Algorithmen) [37, 39]

Für das Matching Problem in gewichteten Graphen gibt es eine Vielzahl von Anwendungen. Zunächst dient es als Lösung für andere Probleme, wie das Nice-Basis-Problem [32], das Graph-Partitioning-Problem [34], das Chinese-Postman-Problem [27] und das Minimum Weight T-Join Problem [11]. Weiterhin verwandte Beziehungen gibt es zum Fermat-Weber-Problem [19], zu den Lateinischen Quadraten [5] und zum Edge Dominating Set Problem [41]. Aber auch im praktischen Bereich gibt es die unterschiedlichsten Applikationen wie beim Übertragen von Bildern über Netzwerke [35], beim Partitionierungsproblem im VLSI-Design [29] und bei der NASA wird es für die Terminplanung von Simulator-Trainingseinheiten benutzt [2].

Ein grundlegender Meilenstein in der Entwicklung von Algorithmen zur Lösung des Matching Problems in gewichteten Graphen ist der erste polynomielle *blossom*-Algorithmus von Edmonds [10] aus dem Jahre 1965, welcher eine Laufzeit von $\mathcal{O}(n^2m)$ benötigt, wobei n die Anzahl der Knoten und m die Anzahl der Kanten bezeichnet. Edmonds Algorithmus ist eine Verallgemeinerung von Kuhns *Ungarischem* Algorithmus [22, 23] für Matchings in gewichteten *bipartiten* Graphen, welcher durch Fredman und Tarjan [12] in $\mathcal{O}(mn + n^2 \log n)$ implementiert wurde.

Durch effizientere Implementierung von Edmonds Algorithmus konnte die Laufzeit 1973 durch Lawler [24] und etwas später durch Gabow [13] auf $\mathcal{O}(n^3)$ verbessert werden. 1986 entwickelten Galil, Micali und Gabow [17] eine weitere Implementation mit einer Laufzeit von $\mathcal{O}(nm \log n)$, was wiederum duch Gabow, Galil und Spencer [15] auf $\mathcal{O}(mn \log\log\log_{\max(m/n,2)} n + n^2 \log n)$ verringert wurde. Abermals Gabow stellte 1990 in [14] den bis heute schnellsten bekannten Algorithmus für gewichtetes Matching in allgemeinen Graphen vor mit einem Zeitaufwand von $\mathcal{O}(mn + n^2 \log n)$, was bislang nur für bipartite Graphen möglich war. Eine genaue Auflistung von Implementationen des Algorithmus von Edmonds ist in [4] zu finden.

Unter speziellen Voraussetzungen gibt es schnellere exakte Algorithmen. Sind z.B. alle Gewichte ganzzahlig aus dem Bereich $[1 \ldots N]$, so benötigt der Algorithmus von Gabow und Tarjan [16] eine Laufzeit von $\mathcal{O}(\sqrt{n \cdot \alpha(m,n) \log n} \cdot m \log(Nn))$, wobei α die Inverse der Ackermann-Funktion [36] darstellt. Für ungewichtete Graphen ist der Algorithmus von Micali und Vazirani [28, 38] mit $\mathcal{O}(\sqrt{n}\, m)$ der schnellste. Für planare Graphen haben Lipton und Tarjan [25] die Lösbarkeit in $\mathcal{O}(n^{3/2} \log n)$ gezeigt.

Benutzt eine Anwendung sehr große Graphen-Instanzen oder aber berechnet Matchings sehr häufig, so kann die Benutzung exakter Algorithmen zu zeitaufwändig sein. In diesem Fall bewährt sich der Einsatz *approximativer* Algorithmen, die zwar mitunter nicht das Optimum berechnen, dafür aber eine gewisse Güte ihrer Lösung garantieren und vor allem schnell arbeiten. Ein Approximationsalgorithmus hat eine *Güte* von $c \leq 1$, wenn er für jede Probleminstanz eine Lösung findet, die mindestens c-mal so groß ist, wie die jeweils optimale Lösung. Ein weiterer Vorteil neben der kurzen Laufzeit ist der meist erheblich geringere Quellcode-Umfang, der im praktischen Einsatz ebenfalls eine Rolle spielt.

Eine der einfachsten approximativen Methoden ist das Berechnen eines *maximalen* Matchings in linearer Laufzeit $\mathcal{O}(n+m)$ durch sukzessive Auswahl von Kanten unter Einhaltung der Matching-Eigenschaft. Viele Varianten dieser Methode werden aufgrund ihrer Schnelligkeit praktisch genutzt [21], obwohl sie beliebig schlecht sein kann, also eine theoretische Güte von 0 hat. Auch in dieser Arbeit wurde eine Variante zum Erzeugen einer Startlösung genutzt, deren Größe zwischen 4 bis 15 Prozent vom Optimum auf den benutzten Instanzen abwich.

Eine theoretische Güte von 1/2 verspricht hingegen der *Greedy* Algorithmus [1], jedoch benötigt er eine Laufzeit von $\mathcal{O}(m \log n)$, da er die Kanten sortiert. Den ersten *linearen* Approximationsalgorithmus mit einer Güte besser als 0, und zwar 1/2, veröffentlichte Preis in [33]. Im Jahr 2003 präsentierten Drake und Hougardy [6] ihren PathGrowing-Algorithmus, welcher ebenfalls in $\mathcal{O}(m+n)$ ein Matching mit Güte 1/2 garantierte, jedoch wesentlich einfacher war als der Algorithmus von Preis.

In [7] und [8] führten sie zudem eine Heuristik (*local improvements*) ein,

welche in linearer Zeit die Ergebnisse vieler Tests deutlich verbesserte. Diese Idee weiterführend und mit Hilfe einer aufwändigen Analyse konnten sie in [9] einen relativ einfachen linearen Algorithmus mit einer Güte von $2/3 - \epsilon$ und einer Laufzeit von $\mathcal{O}((m+n) \cdot 1/\epsilon)$ präsentieren. Die Größe von ϵ wirkt sich umgekehrt proportional auf die Anzahl benötigter Iterationen des eigentlichen Algorithmenkerns aus, der eine Laufzeit von $\mathcal{O}(m+n)$ hat.

Pettie und Sanders [31] veröffentlichten dieses Jahr einen Algorithmus, welcher ebenfalls mit Hilfe von local improvements eine Güte von $2/3 - \epsilon$ in einer Laufzeit von $\mathcal{O}((m+n)\log 1/\epsilon)$ erreicht. Ihr Algorithmus benötigt nur $\log 1/\epsilon$ Iterationen und hat im Vergleich zu Drake und Hougardy somit eine exponentiell schnellere Konvergenz-Geschwindigkeit. Zudem kann er so verändert werden, dass er eine Güte von $\mathcal{O}(1-1/k-\epsilon)$, also beliebig nah an 1, erreicht, die Laufzeit $\mathcal{O}(n\,(\Delta-1)^{k-2}\,\log\frac{1}{\epsilon})$ hängt dann jedoch vom Maximalgrad Δ ab. Weiterhin entwickelten sie einen einfachen randomisierten Algorithmus mit einer erwarteten Güte von $2/3 - \epsilon$ mit einer erwarteten Laufzeit von $\mathcal{O}((m+n)\log 1/\epsilon)$.

In Rahmen dieser Arbeit wurden zwei weitere approximative Algorithmen entwickelt, der `Greedy-`$\mathcal{A}_5$ und `Greedy-`$\mathcal{A}_7$. Ein neuer Ansatz ist es, die local improvements mit Hilfe verschiedener *Partitionen* zu berechnen und zu erweitern. Ihr Gewicht wird durch eine konstruierte *Überdeckung* abgeschätzt, welche eine Erweiterung der in [7] im Beweis von Theorem 1 angewandten Methode ist. Pettie und Sanders benutzen in [31] im Theorem 2.1 ebenfalls eine Erweiterung dieser Methode, im Gegensatz dazu sind hier die in der Überdeckung enthaltenen Elemente jedoch nicht disjunkt.

Der `Greedy-`$\mathcal{A}_5$ erreicht ebenfalls eine Güte von $2/3 - \epsilon$ mit einer Laufzeit von $\mathcal{O}((m+n+\frac{1}{\epsilon}\log\frac{n}{\epsilon})\log 1/\epsilon)$. Ist $\epsilon \geq \frac{\log n}{m}$, so ergibt dies eine Laufzeit von $\mathcal{O}((m+n)\log 1/\epsilon)$, was dem Algorithmus von Pettie und Sanders entspricht. Man beachte, dass bei zu kleinem ϵ der exakte Algorithmus von Gabow [14] mit $\mathcal{O}(mn + n^2 \log n)$ schneller ist.

Der `Greedy-`$\mathcal{A}_7$ garantiert sogar eine Güte von $3/4 - \epsilon$, benötigt dazu jedoch superlineare Laufzeit $\mathcal{O}((m+n) \cdot \frac{1}{\epsilon}\,\log\frac{n}{\epsilon}\,\log\frac{1}{\epsilon})$.

Die Arbeit gliedert sich in fünf Kapitel. Im zweiten werden Grundlagen und gemeinsame Begriffe erläutert. Insbesondere werden die Eigenschaften von *Überdeckungen* und *Partitionen* erklärt. Das dritte Kapitel beschreibt den Algorithmus `Greedy-`$\mathcal{A}_5$ und analysiert seine Laufzeit und Korrektheit. Darauf aufbauend wird im nächsten Kapitel `Greedy-`$\mathcal{A}_7$ beschrieben und analysiert. Der praktische Einsatz beider Algorithmen im Vergleich zum Algorithmus von Drake und Hougardy wird im letzten Kapitel dargestellt.

Kapitel 2

Grundlagen

2.1 Terminologie

Eine Teilmenge der Kanten eines Graphen $G = (V, E)$ ist ein **Matching** $\boldsymbol{M}$, wenn kein Knoten in mehr als einer Kante aus M enthalten ist. Damit ist jeder Knoten $u \in V$ in höchstens einer Matchingkante $\boldsymbol{m_u} = uu'$ enthalten, und $\boldsymbol{m_u(u)}$ bezeichnet den Knoten u'. Falls $u \notin V(M)$, so sei $m_u = \emptyset = m_u(u)$.

Für die Nicht-Matching-Kante $uv \notin M$ ist $\{m_u, uv\}$ ein **Arm von** $\boldsymbol{v}$ über u. Ist m_u leer, so wird er **reduziert** genannt. Zum Arm $l = \{m_u, uv\}$ werden die Arme von $m_u(u)$ **Erweiterungen von** $\boldsymbol{l}$ **bzw.** $\boldsymbol{m_u(u)}$ genannt. Für reduzierte Arme sei die Menge der Erweiterungen leer.

Die Anzahl aller Kanten des Graphen $|E|$ wird mit $\boldsymbol{m}$ bezeichnet.

Lemma 1 *Es gibt insgesamt höchstens* $2m$ *Arme in* G*.*

Beweis. Ein Knoten hat maximal so viele Arme wie **Nachbarn**[1]. Alle Knoten haben insgesamt höchstens $2m$ Nachbarn. □

Ein Pfad oder Kreis heißt **alternierend** bezüglich einem Matching M, wenn seine Kanten abwechselnd aus M und $E \setminus M$ stammen. Ein alternierender Pfad oder Kreis A wird **Alternative** genannt, bzw. ist **alternativ**, wenn die disjunkte Vereinigung $M \setminus A \uplus A \setminus M$ wiederum ein Matching ist.[2]

Zu jeder Kante sei ein Gewicht durch die **Gewichtsfunktion** $\boldsymbol{\omega} : E \mapsto \mathbb{R}_{\geq 0}$ gegeben. Das Gewicht einer Kantenmenge A wird dann definiert als $\boldsymbol{\omega(A)} = \sum_{e \in A} \omega(e)$. Der **Gewinn** einer Kantenmenge A gegenüber dem Matching M ist definiert als

$$\mathbf{gain}_{\boldsymbol{M}}(\boldsymbol{A}) = \omega(A \setminus M) - \omega(A \cap M).$$

Ein Matching M, welches $\omega(M)$ maximiert, heißt **optimales Matching** $\boldsymbol{M^*}$ und sei nun beliebig aber fest gewählt. Das **Matchingproblem für gewichtete Graphen** entspricht der Suche nach einem optimalen Matching. Ein Matching heißt **maximal**, wenn jede Kante des Graphen mit einer Matchingkante inzident[3] ist. Wegen der nicht negativen Gewichte kann man davon ausgehen, dass jedes optimale Matching maximal ist.

[1] u ist ein Nachbar von bzw. benachbart mit v, wenn $uv \in E$.

[2] Bei einer Alternative muss $M \setminus A \uplus A \setminus M$ nicht größer sein als M bzgl. Gewicht oder Kantenzahl im Gegensatz zur allgemein bekannten „Augmentation".

[3] Zwei Kanten sind inzident, wenn sie einen gemeinsamen Endknoten haben.

M sei ein beliebiges aber fest gewähltes maximales Matching. Die Begriffe „Matching“ und „Matchingkante“ beziehen sich im Folgenden auf dieses Matching M. Die Zusammenhangskomponenten der symmetrischen Differenz zwischen M und dem Optimum $\boldsymbol{\mathcal{A}}^* = M \triangle M^*$ liefern eine Menge von „optimalen“ Alternativen, nämlich Pfade und gerade Kreise, deren Kanten abwechselnd aus M und M^* stammen. Entfernt man die Matchingkanten (bzgl. M) dieser Alternativen aus M und fügt die Nicht-Matching-Kanten zu M hinzu, so erhält man aus M gerade M^*. Ziel ist es daher, diese Alternativen zu approximieren.

$\mathcal{A}^*$ kann disjunkt zerlegt werden in $\mathcal{A}^* = \boldsymbol{\mathcal{A}^*_{C_4}} \uplus \boldsymbol{\mathcal{A}^*_{\infty}}$, wobei $\mathcal{A}^*_{C_4}$ genau die C_4-Kreise in $\mathcal{A}^*$ umfasst, $\mathcal{A}^*_{\infty}$ stellt das entsprechende Komplement dar. $\mathcal{A}^*$, $\mathcal{A}^*_{C_4}$ und $\mathcal{A}^*_{\infty}$ sind Mengen von Kanten. Jede Zusammenhangskomponente dieser Mengen bildet eine Alternative, jede der drei Mengen kann also auch als Menge disjunkter Alternativen betrachtet werden. Bezeichnet $\mathcal{A}$ eine dieser drei Mengen so wird sich mit der Alternative $A \in \mathcal{A}$ stets auf eine Zusammenhangskomponente bezogen und $\|\boldsymbol{\mathcal{A}}\|$ entspricht der Anzahl der Zusammenhangskomponenten, also der Anzahl dieser Alternativen. Nach Definition des Gewinns gilt $\forall A \in \mathcal{A}^* :\ \text{gain}(A) = \omega(M^* \cap A) - \omega(M \cap A)$.

Mit $\boldsymbol{\mathcal{A}_5}$ werden alle alternativen Pfade in G mit höchstens zwei Nicht-Matching-Kanten bezeichnet. Ein solcher nicht leerer Pfad hat somit höchstens fünf Kanten und mindestens eine Matchingkante, da das Matching M maximal ist. Eine Matchingkante einer solchen Alternative wird **Zentrum** genannt, wenn, beginnend beim Zentrum, jede andere Kante der Alternative über höchstens eine weitere Kante der Alternative erreicht werden kann.

Mit $\boldsymbol{\mathcal{A}_7}$ werden alle alternativen Pfade mit höchstens drei Nicht-Matching-Kanten, sowie alle alternativen C_6-Kreise in G bezeichnet. Ein solcher nicht leerer Pfad hat somit höchstens sieben Kanten und mindestens eine Matchingkante, da das Matching M maximal ist. Eine Matchingkante einer solchen Alternative wird **Zentrum** genannt, wenn, beginnend beim Zentrum, jede andere Kante der Alternative über höchstens drei weitere Kanten der Alternative erreicht werden kann. Bei einer Orientierung des Zentrums $\vec{c} = (vw)$ kann ein Arm von v dann als **linker Arm** von $\vec{c}$, bzw. ein Arm von w als **rechter Arm** von $\vec{c}$ bezeichnet werden.

$\mathcal{A}_5$ und $\mathcal{A}_7$ sind Mengen von Alternativen. Bei Mengenoperationen zwischen $\mathcal{A}_5$ bzw. $\mathcal{A}_7$ und $\mathcal{A}^*$, $\mathcal{A}^*_{C_4}$ bzw. $\mathcal{A}^*_{\infty}$ werden letztere wie oben ausgeführt ebenfalls als Mengen von Alternativen angesehen, bezogen auf Zusammenhangskomponenten.

Lemma 2 *Eine $\mathcal{A}_5$-Alternative A hat folgende Eigenschaft: Ist c ein Zentrum von A, so kann A wie in Abbildung 2.1 aufgeteilt werden in das Zentrum c und einen (mitunter leeren oder reduzierten) Arm zu jeder Seite. Eine $\mathcal{A}_7$-Alternative A hat die Eigenschaft: Ist c ein Zentrum von A, so kann A wie in Abbildung 2.2 bei geeigneter Orientierung des Zentrums aufgeteilt werden in das Zentrum $\vec{c}$ und einen (mitunter leeren oder reduzierten) Arm zu jeder Seite und einer (mitunter leeren) Erweiterung des linken Arms.* □

Eine solche Zerlegung einer $\mathcal{A}_5$-Alternative wird auch als Tupel (l, c, r) bzw. die Zerlegung einer $\mathcal{A}_7$-Alternative als $(x_l, l, \vec{c}, r)$ dargestellt, wobei l der linke Arm, r der rechte Arm und x_l die Erweiterung von l ist. Bei den C_6-Kreisen überschneiden sich der rechte Arm und die Erweiterung auf der Matchingkante.

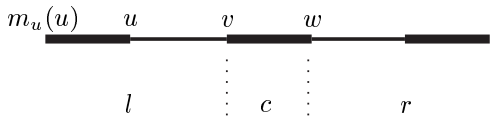

Abbildung 2.1: $\mathcal{A}_5$-Zerlegung in Zentrum c mit den Armen l und r

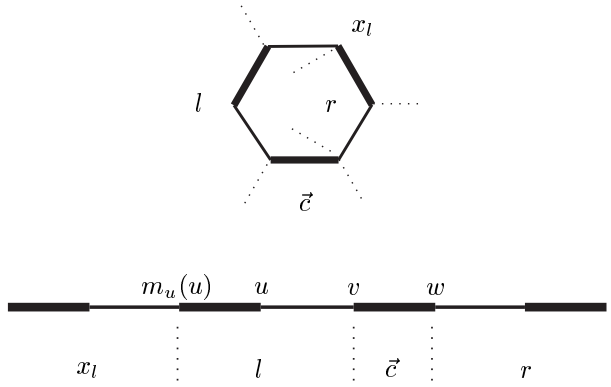

Abbildung 2.2: $\mathcal{A}_7$-Zerlegung analog in (l, c, r) und die Erweiterung x_l

Der Gewinn$_7$ bezüglich Zentrum $\vec{c}$ und der Zerlegung $(x_l, l, \vec{c}, r)$ wird definiert als

$$\mathbf{gain}_{\mathbf{7}}^{\vec{c}}(\boldsymbol{A}) = \text{gain}(x_l) + \text{gain}(l) - \omega(c) + \text{gain}(r).$$

Korollar 3 $\forall A \in \mathcal{A}_7 : \text{gain}_7^{\vec{c}}(A) \leq \text{gain}(A)$*, wobei Ungleichheit nur für Kreise eintritt. Die Werte unterscheiden sich dann genau um das Gewicht der überlappenden Matchingkante (im rechten Arm).* □

2.2 Überdeckung von $\mathcal{A}_\infty^*$

Die Idee, den Gewinn langer Alternativen durch sich überlappende kurze Alternativen abzuschätzen, entspricht den *local improvements* in [7, 8] und wurde hier adaptiert. Dabei ist eine Überdeckung Ω_5 bzw. Ω_7 von $\mathcal{A}_\infty^*$ eine Menge von mitunter mehrfach auftretenden oder sich überlappenden Alternativen aus $\mathcal{A}_5$ bzw. $\mathcal{A}_7$, wenn jede überdeckende Alternative nur Kanten aus $\mathcal{A}_\infty^*$ benutzt und umgekehrt jede Kante aus $\mathcal{A}_\infty^*$ in einer überdeckenden Alternative enthalten ist.

Lemma 4 *Es gibt eine Überdeckung Ω_5 von $\mathcal{A}_\infty^*$, die aus (mitunter mehrfach auftretenden oder sich überlappenden) Alternativen A aus $\mathcal{A}_5$ besteht, so dass ein Knoten in nicht mehr als drei dieser Alternativen enthalten ist und zudem gilt:*

$$\sum_{A \in \Omega_5} \text{gain}(A) \geq 2 \cdot \omega(M^* \cap \mathcal{A}_\infty^*) - 3 \cdot \omega(M \cap \mathcal{A}_\infty^*).$$

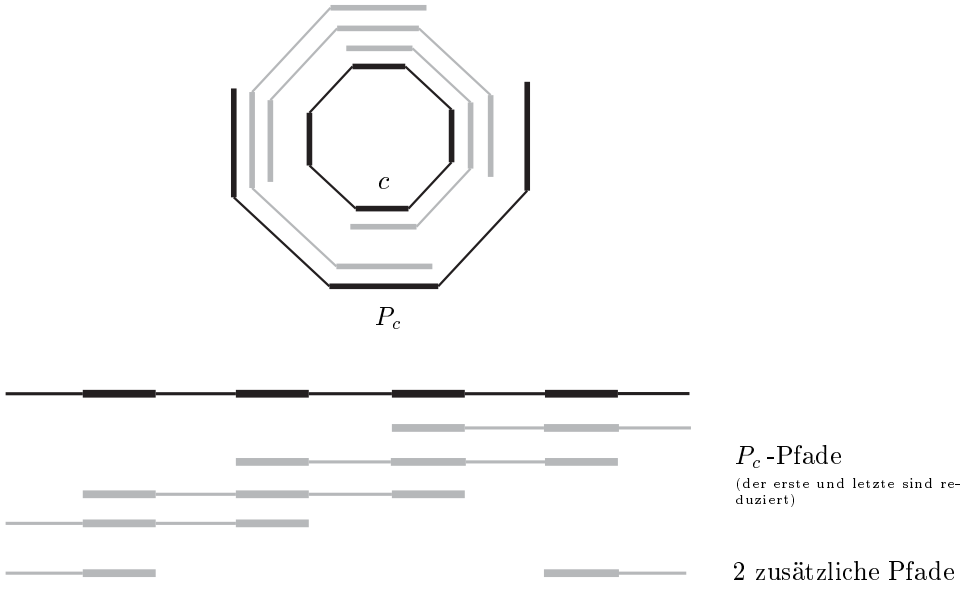

Abbildung 2.3: Überdeckung eines Kreises und Pfades durch $\mathcal{A}_5$-Alternativen

Beweis. Um eine Überdeckung Ω_5 zu konstruieren, bedarf es der Unterscheidung in mehrere knotendisjunkte Fälle. Wie oben angeführt, ist $\mathcal{A}_5$ eine Menge (nicht disjunkter) „kurzer“ Alternativen und $\mathcal{A}^*_\infty$ kann als Menge disjunkter Alternativen betrachtet werden. Folglich beziehen sich Mengenoperationen zwischen $\mathcal{A}_5$ und $\mathcal{A}^*_\infty$ auf die Zusammenhangskomponenten von $\mathcal{A}^*_\infty$.

1. Fall: Pfade in $\mathcal{A}^*_\infty \cap \mathcal{A}_5$

Ein $\mathcal{A}_5$-Pfad in $\mathcal{A}^*_\infty$ ist eine Zusammenhangskomponente von $\mathcal{A}^*_\infty$, die auch in $\mathcal{A}_5$ enthalten ist. Ein jeder dieser $\mathcal{A}_5$-Pfade wird mit sich selbst zweimal überdeckt, die so zur Überdeckung hinzugefügten Pfade werden (hilfsweise) mit Ω_P bezeichnet. Dann gilt:

$$\sum_{P \in \Omega_P} \text{gain}(P) = \sum_{P \in \mathcal{A}^*_\infty \cap \mathcal{A}_5} 2 \cdot \text{gain}(P) = \sum_{P \in \mathcal{A}^*_\infty \cap \mathcal{A}_5} 2 \cdot \omega(M^* \cap P) - 2 \cdot \omega(M \cap P).$$

2. Fall: Kreise in $\mathcal{A}^*_\infty$

Ein Kreis in $\mathcal{A}^*_\infty$ hat eine Länge von mindestens sechs. Drei im Kreis aufeinanderfolgende Matchingkanten bilden zusammen mit den zwei dazwischenliegenden Nicht-Matching-Kanten wie in Abbildung 2.3 einen Pfad $P_c \in \mathcal{A}_5$ mit Zentrum c. Zu jeder Matchingkante c des Kreises wird in Ω_5 solch ein Pfad P_c eingefügt. Auf diese Weise wird jede Matchingkante einmal als Zentrum und zweimal im Arm ihrer benachbarten Matchingkanten, also genau dreimal in Ω_5 eingefügt. Jede Nicht-Matching-Kante wird analog genau zweimal in Ω_5 eingefügt. Bezeichnet man die neu hinzugekommenen Pfade mit Ω_C, so folgt:

$$\sum_{P_c \in \Omega_C} \text{gain}(P_c) = \sum_{C \in \{Kreise\ in\ \mathcal{A}^*_\infty\}} 2 \cdot \omega(M^* \cap C) - 3 \cdot \omega(M \cap C).$$

3. Fall: Pfade aus $\mathcal{A}^*_\infty \setminus \mathcal{A}_5$

Wie im vorigen Fall wird jeder Pfad $P \in \mathcal{A}^*_\infty \setminus \mathcal{A}_5$ durch längstmögliche Pfade $P_c \in \mathcal{A}_5$ zu jeder seiner Matchingkanten c wie in Abbildung 2.3 überdeckt. Da jedoch insbesondere an den Pfad-Endstücken die Pfade P_c reduzierte oder gar keine Arme haben, müssen noch bis zu zwei weitere Alternativen hinzugefügt werden.

Angenommen, die Endkanten von P sind Matchingkanten. Dann ist jede Nicht-Matching-Kante von P mit zwei Matchingkanten c_1 und c_2 auf dem Pfad inzident und wird genau zweimal zu Ω_5 hinzugefügt, nämlich in einem Arm des Pfades P_{c_1} und in einem Arm von P_{c_2}. Die nicht äußeren Matchingkanten auf P sind analog zweimal im Arm anderer Zentren und einmal selbst als Zentrum in solch einem Pfad enthalten. Die beiden äußeren sind analog insgesamt genau zweimal in Ω_5 enthalten.

Endet der Pfad auf eine Nicht-Matching-Kante e mit benachbarter Matchingkante c, so wird zusätzlich noch die Alternative bestehend aus c und e zur Überdeckung hinzugefügt. Damit ist e einmal im Arm des Pfades P_c und einmal in der zusätzlichen Alternative enthalten. Die Matchingkante c wird nun einmal mehr als zuvor, also dreimal zur Überdeckung hinzugefügt.

Bezeichnet Ω_{P_+} die Menge der P_c-Alternativen und die bis zu zwei zusätzlichen Alternativen, so folgt:

$$\sum_{P_+ \in \Omega_{P_+}} \text{gain}(P_+) \geq \sum_{P \in \{Pfade\ in\ \mathcal{A}^*_\infty \setminus \mathcal{A}_5\}} 2 \cdot \omega(M^* \cap P) - 3 \cdot \omega(M \cap P).$$

Die drei Fälle bezogen sich auf knotendisjunkte Zusammenhangskomponenten von $\mathcal{A}^*_\infty$, damit kann man die Gleichungen zusammenfassen und erhält die gewünschte Überdeckung $\Omega_5 := \Omega_P \uplus \Omega_C \uplus \Omega_{P_+}$. □

Sei $\|\Omega_5\|$ die Anzahl der zur Überdeckung gehörenden Alternativen und $\boldsymbol{n}$ die Anzahl der Knoten des Graphen. Da jede Alternative aus mindestens zwei Knoten besteht, erhält man mit Lemma 4 die beiden Schlussfolgerungen:

Korollar 5 $\|\Omega_5\| \leq 3/2 \cdot n$. □

Korollar 6 *Eine Matchingkante ist höchstens dreimal in* Ω_5 *enthalten.* □

Lemma 7 *Es gibt eine Überdeckung* Ω_7 *von* $\mathcal{A}^*_\infty$*, die aus (mitunter mehrfach auftretenden oder sich überlappenden) Alternativen* A *aus* $\mathcal{A}_7$ *(mit jeweiligem Zentrum* $\vec{c}_A$*) besteht, so dass ein Knoten in nicht mehr als vier dieser Alternativen enthalten ist und zudem gilt:*

$$\sum_{A \in \Omega_7} \text{gain}_7^{\vec{c}_A}(A) \geq 3 \cdot \omega(M^* \cap \mathcal{A}^*_\infty) - 4 \cdot \omega(M \cap \mathcal{A}^*_\infty).$$

Beweis. Der Beweis verläuft analog zum Beweis von Lemma 4. Dabei sei eine Orientierung der Kanten vorausgesetzt, unter der die Pfade und Kreise jeweils zusammenhängend sind. Auch hier beziehen sich Mengenoperationen zwischen $\mathcal{A}_7$ und $\mathcal{A}^*_\infty$ auf die Zusammenhangskomponenten von $\mathcal{A}^*_\infty$.

1. Fall: Pfade in $\mathcal{A}^*_\infty \cap \mathcal{A}_7$

Alle Pfade $P \in \mathcal{A}^*_\infty \cap \mathcal{A}_7$ sind Zusammenhangskomponenten von $\mathcal{A}^*_\infty$, die auch in $\mathcal{A}_7$ enthalten sind. Sie werden mit sich selbst dreimal überdeckt, d.h. jeder Pfad wird dreimal zur Überdeckung Ω_7 hinzugefügt. Die so dazukommenden Pfade werden mit Ω_P bezeichnet. Es folgt mit Korollar 3 (S. 6) analog:

$$\begin{aligned}\sum_{P\in\Omega_P} \text{gain}_7^{\vec{c}_P}(P) &= \sum_{P\in\Omega_P} \text{gain}(P) = \sum_{P\in\{Pfade\ in\ \mathcal{A}^*_\infty\cap\mathcal{A}_7\}} 3\cdot\text{gain}(P)\\ &= \sum_{P\in\{Pfade\ in\ \mathcal{A}^*_\infty\cap\mathcal{A}_7\}} 3\cdot\omega(M^*\cap P) - 3\cdot\omega(M\cap P).\end{aligned}$$

2. Fall: Kreise in $\mathcal{A}^*_\infty \cap \mathcal{A}_7$

Alle Kreise $C \in \mathcal{A}^*_\infty \cap \mathcal{A}_7$ sind C_6-Kreise und werden mit sich selbst dreimal überdeckt, genauer wird jeder Kreis bezüglich jeder seiner Matchingkanten als Zentrum einmal zur Überdeckung Ω_7 hinzugefügt, der Gewinn_7 also zu jeder der drei Matchingkanten berechnet. Dabei überlappt jede Matchingkante eines Kreises aufgrund der gewählten Orientierung genau einmal (s. Abbildung 2.2, S. 6). Die so hinzugefügten Kreise werden mit Ω_C bezeichnet. Es folgt mit Korollar 3 (S. 6):

$$\begin{aligned}\sum_{C\in\Omega_C} \text{gain}_7^{\vec{c}}(C) &= \sum_{C\in\{Kreise\ in\ \mathcal{A}^*_\infty\cap\mathcal{A}_7\}} \text{gain}_7^{\vec{c}_1}(C) + \text{gain}_7^{\vec{c}_2}(C) + \text{gain}_7^{\vec{c}_3}(C)\\ &= \sum_{C\in\{Kreise\ in\ \mathcal{A}^*_\infty\cap\mathcal{A}_7\}} 3\cdot\text{gain}(C) - \omega(\vec{c}_1) - \omega(\vec{c}_2) - \omega(\vec{c}_3)\\ &= \sum_{C\in\{Kreise\ in\ \mathcal{A}^*_\infty\cap\mathcal{A}_7\}} 3\cdot\omega(M^*\cap C) - 4\cdot\omega(M\cap C).\end{aligned}$$

3. Fall: Kreise in $\mathcal{A}^*_\infty \setminus \mathcal{A}_7$

Jeder Kreis in $\mathcal{A}^*_\infty \setminus \mathcal{A}_7$ ist ein Kreis von Mindestlänge acht, bezeichnet als $C_{\geq 8}$. Es wird wie in Abbildung 2.4 zu jeder Matchingkante $\vec{c}$ in $C_{\geq 8}$ der Pfad $P_{\vec{c}}$ zu Ω_7 hinzugefügt, der sich aus dem Zentrum $\vec{c}$, einem linken und rechten Arm und der Erweiterung des linken Arms zusammensetzt. Durch die gewählte Orientierung wird damit jede Matchingkante von $C_{\geq 8}$ genau viermal (nämlich einmal als Zentrum eines Pfades, einmal im linken Arm, einmal im rechten Arm und einmal in der Erweiterung je eines Pfades), jede Nicht-Matching-Kante von $C_{\geq 8}$ analog genau dreimal zu Ω_7 hinzugefügt. So hinzugefügte Pfade werden als Ω_{C_+} bezeichnet. Es folgt mit Korollar 3 (S. 6):

$$\sum_{P_{\vec{c}}\in\Omega_{C_+}} \text{gain}_7^{\vec{c}}(P_{\vec{c}})) = \sum_{C_{\geq 8}\in\{Kreise\ in\ \mathcal{A}^*_\infty\setminus\mathcal{A}_7\}} 3\cdot\omega(M^*\cap C_{\geq 8}) - 4\cdot\omega(M\cap C_{\geq 8}).$$

4. Fall: Pfade in $\mathcal{A}^*_\infty \setminus \mathcal{A}_7$

Wie bei der Überdeckung mit $\mathcal{A}_5$-Alternativen werden nicht in $\mathcal{A}_7$ enthaltene Pfade durch maximale Pfade $P_{\vec{c}}$ zu jeder Matchingkante $\vec{c}$ des Pfades überdeckt, die jeweils aus dem Zentrum $\vec{c}$, und falls möglich, aus einem rechten

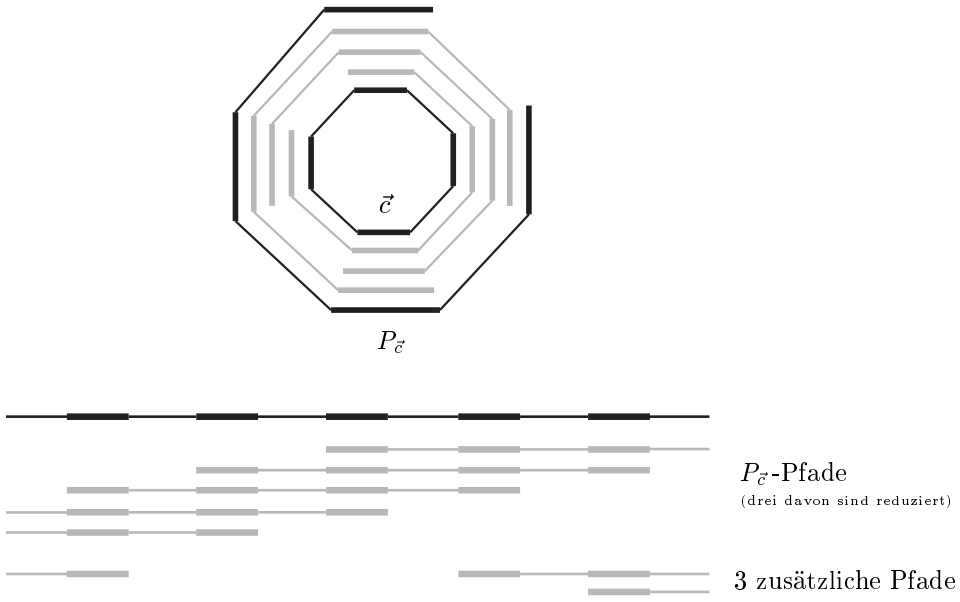

Abbildung 2.4: Überdeckung eines Kreises und Pfades durch $\mathcal{A}_7$-Alternativen

und linken Arm und einer Erweiterung des linken Arms bestehen. Werden die Nicht-Matching-Kanten des Pfades nicht alle dreimal überdeckt, so werden wie in Abbildung 2.4 bis zu drei weitere Pfade hinzugefügt, so dass analog zur $\mathcal{A}_5$-Überdeckung jede Nicht-Matching-Kante mindestens dreimal und jede Matchingkante höchstens viermal überdeckt ist. Bezeichnet Ω_{P_+} die Menge der $P_{\vec{c}}$-Alternativen und bis zu drei zusätzlichen Alternativen, so folgt mit Korollar 3 (S. 6):

$$\begin{aligned}\sum_{P_+ \in \Omega_{P_+}} \text{gain}_7^{\vec{c}}(P_+) &= \sum_{P_+ \in \Omega_{P_+}} \text{gain}(P_+) \\ &\geq \sum_{P \in \{Pfade\ in\ \mathcal{A}_\infty^* \setminus \mathcal{A}_7\}} 3 \cdot \omega(M^* \cap P) - 4 \cdot \omega(M \cap P).\end{aligned}$$

Die gewünschte Überdeckung erhält man als Vereinigung der knotendisjunkten Fälle: $\Omega_7 := \Omega_P \uplus \Omega_C \uplus \Omega_{C_+} \uplus \Omega_{P_+}$. □

Analog zu Korollar 5 und 6 (S. 8) folgen

Korollar 8 $\|\Omega_7\| \leq 2 \cdot n$. □

Korollar 9 *Eine Matchingkante ist höchstens viermal in Ω_7 enthalten.* □

2.3 Partitionen

Chen und Uehara haben in [37] Partitionsalgorithmen vorgestellt. Die dort vorgestellte Partition von Kanten wird hier verallgemeinert, da sie im Folgenden für unterschiedliche Mengen mehrfach gebraucht wird.

Sei S eine beliebige endliche Menge und $f : S \mapsto \mathbb{R}$ eine Bewertungsfunktion mit $f_{max} = \max(1 \cup f(S))$. Jedem $s \in S$ wird ein ganzzahliger **Rang $r(s)$** mit den **Parametern** $0 < \boldsymbol{\alpha, \beta} < 1 \leq \boldsymbol{n}$ zugewiesen und alle Elemente mit gleichem Rang i bilden die **Partitionsklasse $\mathcal{P}_i$**.

$$r(s) = \begin{cases} 0 & : \quad \frac{n}{\alpha} \cdot \frac{f(s)}{f_{max}} \leq 1 \\ i \in \mathbb{Z} & : \quad (1+\beta)^{i-1} < \frac{n}{\alpha} \cdot \frac{f(s)}{f_{max}} \leq (1+\beta)^i \quad \text{sonst.} \end{cases}$$

Lemma 10 *Für jedes Element a bzw. b von S gilt:*

i. Ist $r(a) = 0$, so ist $0 \geq f(a) - \frac{1}{n}\,\alpha\, f_{max}$.

ii. Ist $0 < r(a) \leq r(b)$, so ist $f(b) > f(a)/(1+\beta) > 0$.

Beweis.
Ad i.) Dies folgt direkt aus der Definition.
Ad ii.) Mit $0 < i = r(a) \leq r(b)$ folgt $(1+\beta)^{i-1} < \frac{n}{\alpha\, f_{max}} \cdot f(b)$ und $0 < \frac{n}{\alpha\, f_{max}} \cdot f(a) \leq (1+\beta)^i$ und damit $0 < f(a) < (1+\beta) \cdot f(b)$. □

Korollar 11 *Ist $r(a) \leq r(b)$ bzw. $r(a) = 0$, so ist*

i. $\max(0, f(b)) \geq \frac{f(a) - \frac{1}{n}\,\alpha\, f_{max}}{1+\beta}$ *bzw.* $0 \geq \frac{f(a) - \frac{1}{n}\,\alpha\, f_{max}}{1+\beta}$

ii. $\max(0, f(b)) \geq \frac{f(a) - \frac{1}{n}\,\alpha\, f_{max}}{(1+\beta)^2}$ *bzw.* $0 \geq \frac{f(a) - \frac{1}{n}\,\alpha\, f_{max}}{(1+\beta)^2}$.

Beweis. Falls $r(a) = 0$ folgt aus Lemma 10.i $0 \geq \frac{f(a) - \frac{1}{n}\,\alpha\, f_{max}}{1+\beta}$ und damit auch $0 \geq \frac{f(a) - \frac{1}{n}\,\alpha\, f_{max}}{(1+\beta)^2}$. Falls $r(a) > 0$ folgt aus 10.ii $f(b) > \frac{f(a)}{1+\beta} > 0$, d.h. es gilt auch $f(b) > \frac{f(a)}{(1+\beta)^2}$ und schließlich auch $f(b) > \frac{f(a) - \frac{1}{n}\,\alpha\, f_{max}}{(1+\beta)^2}$. □

Korollar 12 *Es gibt $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ verschiedene Partitionsklassen, die in $\mathcal{O}(|S| + \log_{1+\beta} \frac{n}{\alpha})$ erzeugt und durchlaufen werden können.*

Beweis. Nach obiger Konstruktion ist $r(s) \in \left[0, \ldots, \lceil \log_{1+\beta} \frac{n}{\alpha} \rceil\right]$. Man erzeugt ein Array $\mathcal{P}$ der Länge $1 + \lceil \log_{1+\beta} \frac{n}{\alpha} \rceil$ mit einem zunächst leeren Listen-Zeiger für jeden möglichen Rang. Indem man jedes Element s an den Anfang der Liste $\mathcal{P}[r(s)]$ einfügt, erhält man für jeden Rang i eine Liste $\mathcal{P}_i$. □

Kapitel 3

Der Greedy-$\mathcal{A}_5$-Algorithmus

$\mathcal{A}^*$ setzt sich aus alternativen Pfaden und Kreisen zusammen. Bildet man ein neues Matching aus M, indem man entlang dieser Pfade und Kreise die Kanten $\mathcal{A}^* \cap M$ aus M entfernt und dafür die Kanten $\mathcal{A}^* \setminus M$ hinzufügt, so erhält man das optimale Matching M^*. $\mathcal{A}^*$ zerfällt disjunkt in $\mathcal{A}^*_{C_4}$ und $\mathcal{A}^*_\infty$. Für $\mathcal{A}^*_\infty$ existiert eine Überdeckung Ω_5 von $\mathcal{A}_5$-Alternativen, so dass $\omega(\mathcal{A}^*_\infty)$ durch den Gewinn der Überdeckung abgeschätzt werden kann:

$$\sum_{A \in \Omega_5} \text{gain}(A) \geq 2 \cdot \omega(M^* \cap \mathcal{A}^*_\infty) - 3 \cdot \omega(M \cap \mathcal{A}^*_\infty) \quad \text{(Satz 4, S. 6)}.$$

Ziel war es, einen Algorithmus zu entwickeln, der Alternativen auswählt, mit deren Gewinn wiederum der Gewinn von $\mathcal{A}^*_{C_4}$ und Ω_5 abgeschätzt werden kann. Auf diese Weise kann der Gewinn von $\mathcal{A}^*_\infty$ approximiert werden, und damit auch die Gewichtsdifferenz zwischen M^* und M.

Der `Greedy-`$\mathcal{A}_5$ partitioniert alle alternativen C_4-Kreise nach Gewinn, wodurch stets mit geringem Fehler der bezüglich Gewinn größte approximiert werden kann, der knotendisjunkt zu allen bisher ausgewählten Alternativen ist. Weiterhin werden alle Arme nach Gewinn partitioniert. Dadurch kann mit geringem Fehler zu jeder Matchingkante c die größte $\mathcal{A}_5$-Alternative mit Zentrum c approximiert werden. Diese Approximationen werden ebenfalls nach ihrem Gewinn partitioniert und, bei regelmäßiger Aktualisierung, kann damit die jeweils größte $\mathcal{A}_5$-Alternative approximiert werden, die knotendisjunkt zu allen bisher ausgewählten Alternativen ist. Durch Auswahl eines C_4-Kreises oder einer $\mathcal{A}_5$-Alternative werden höchstens neun Alternativen aus $\mathcal{A}^*_{C_4}$ und Ω_5 geschnitten, deren Gewinn jedoch jeweils von der ausgewählten Alternative dominiert wird. Unter Berücksichtigung der Approximationsfehler wird gezeigt (Satz 21), dass

$$9 \cdot \sum_{A \in Resultat} \text{gain}(A) \geq \frac{2\,\omega(M^* \cap A^*) - 3\,\omega(M \cap A^*) - (7 + 3\,\beta)\,\alpha\,\omega(M^*) - 3\,\beta\,\omega(M)}{(1+\beta)^2}$$

und letztlich erhält man daraus einen Mindestgewinn des Resultats von ungefähr $\frac{1}{3}\left(\frac{2}{3}\,\omega(M^*) - \omega(M)\right)$. Mit jeder Ausführung des Algorithmus wird die Differenz zwischen $\omega(M)$ und $\frac{2}{3}\,\omega(M^*)$ also ungefähr gedrittelt, durch wiederholte Anwendung erhält man ein Matching mit einem Gewicht von $(\frac{2}{3} - \epsilon)\,\omega(M^*)$ mit einer Laufzeit von $\mathcal{O}((m + n + \frac{1}{\epsilon}\log\frac{n}{\epsilon}) \cdot \log\frac{1}{\epsilon})$.

3.1 Die Initialisierung

Wie oben bereits erwähnt, werden die Arme eines jeden Knotens und alle alternativen C_4-Kreise entsprechend ihrem Gewinn partitioniert. Desweiteren wird zu jeder Matchingkante c ein $\mathcal{A}_5$-Pfad mit Zentrum c berechnet, welcher den maximalen $\mathcal{A}_5$-Pfad mit Zentrum c approximiert. Anschließend werden diese Approximationen nach ihrem Gewinn partitioniert. Der Algorithmus `GetGood` mit konstanter Laufzeit zum Berechnen einer solchen Approximation wird innerhalb dieses Abschitts vorgestellt.

3.1.1 Die Partition der C_4-Kreise

Es werden alle alternativen C_4-Kreise enumeriert und nach Rang sortiert in die Liste $\boldsymbol{\mathcal{C}}$ eingefügt, abgesehen von Kreisen mit Rang Null.

Lemma 13 *Es gibt höchstens $m/2$ alternative C_4-Kreise.*

Beweis. Nach Lemma 1 (S. 4) gibt es maximal $2m$ Arme. Jeder Arm ist höchstens in einem alternativen C_4 enthalten und in jedem C_4 sind genau vier Arme enthalten. □

Man enumeriert alle alternativen C_4-Kreise mit der von Drake und Hougardy in [7] angegebenen Methode mit einer Laufzeit von $\mathcal{O}(m)$, indem man zu jeder Matchingkante vw zunächst alle Arme $\{m_u, uv\}$ von v durchläuft und, falls vorhanden, den Knoten $m_u(u)$ markiert. Anschließend testet man jeden Nachbarn von w, ob er markiert ist und damit ein Kreis entdeckt wurde. Dies gelingt in linearer Zeit, da es genau $2m$ Nachbarn gibt. Um nicht jeden Kreis zweimal zu enumerieren, wird er nur gezählt, wenn die zweite zum Kreis gehörende Matchingkante noch nicht abgearbeitet wurde. Alle so gefundenen Kreise werden partitioniert, wobei die Parameter α und β später spezifiziert werden. Parameter n entspricht der Anzahl der Knoten und als Partitionsfunktion wird $f(a) := \text{gain}(a)$ benutzt und damit $f_{max} \leq 2\,\omega_{max}$, wobei $\boldsymbol{\omega_{max}} := \max_{e \in E} \omega(e)$ ist. Anschließend durchläuft man alle Partitionsklassen außer der Partitionsklasse vom Rang Null der Reihe nach und fügt die Kreise nach Rang sortiert in $\mathcal{C}$ ein. Mit Korollar 12 folgt ein Gesamtaufwand von $\mathcal{O}(m + \log_{1+\beta} \frac{n}{\alpha})$.

3.1.2 Die Partition der Arme

Für jeden Knoten v wird eine Liste $\boldsymbol{Arms_v}$ erzeugt, welche die nach Rang sortierten Arme von v enthält. Dazu werden zunächst vorübergehend alle Arme partitioniert mit gleichem α, β, n und f wie die Kreise zuvor, und damit $f_{max} \leq \omega_{max}$. Anschließend wird einmal durch die Partitionsklassen der Arme gelaufen. Jeder Arm kann nur der Arm genau eines Knotens sein und demnach genau einer (noch leeren) Liste $Arms_v$ zugeordnet werden. Dies wird nun der Reihe nach getan. Die Arme in $Arms_v$ sind somit nach ihrem Rang sortiert. Mit Lemma 1 (S. 4) und Korollar 12 (S. 11) folgt ein Gesamtaufwand von $\mathcal{O}(m + \log_{1+\beta} \frac{n}{\alpha})$.

3.1.3 Die Partition der $\mathcal{A}_5$-Pfade

Die Arme eines Knotens v liegen in $Arms_v$ nach Rang sortiert vor. Zu einer Kante $c = \{vw\}$ seien $l_1, l_2, l_3 \in Arms_v$ und $r_1, r_2, r_3 \in Arms_w$ die ersten

drei Arme der jeweiligen Menge, welche aufgrund der Sortierung auch gleich die ranghöchsten Arme sind mit $r(l_1) \geq r(l_2) \geq r(l_3)$ und $r(r_1) \geq r(r_2) \geq r(r_3)$.

GetGood ($c = \{vw\} \in M$)
(1) L := $\{l_1,\ l_2,\ l_3,\ \emptyset\}$
(2) R := $\{r_1,\ r_2,\ r_3,\ \emptyset\}$
(3) *Result* := Maximum von $(L \times \{c\} \times R) \cap \mathcal{A}_5$ bzgl. Gewinn
(4) Wenn gain(*Result*) ≤ 0 so *Result* := $\emptyset$

GetGood kombiniert also die drei ranghöchsten Arme von c und den „leeren" Arm mit den drei ranghöchsten Armen von w und dem leeren Arm. Er liefert die Kombination zurück, die einen $\mathcal{A}_5$-Pfad bildet und den größten Gewinn hat, bzw. die leere Menge, falls das Maximum nicht positiven Gewinn hat.

Lemma 14 GetGood *berechnet mit konstantem Zeitaufwand eine $\mathcal{A}_5$-Alternative oder leere Menge $\boldsymbol{G(c)}$, so dass mit $c = vw$ für alle $(l, c, r) \in \mathcal{A}_5 \cap (Arms_v \cup \{\emptyset\}) \times c \times (Arms_w \cup \{\emptyset\})$ gilt:*

$$\text{gain}(G(c)) \geq \frac{\text{gain}(l, c, r) - \frac{2}{n}\,\alpha\,\omega(M^*) - \beta\,\omega(c)}{1 + \beta}.$$

Beweis. Nicht jedes $(l, c, r) \in L \times \{c\} \times R$ bildet eine $\mathcal{A}_5$-Alternative, wie in Abbildung 3.1 dargestellt. Jeder Arm aus $Arms_v$ kann sich jedoch höchstens mit zwei Armen aus $Arms_w$ auf einem Knoten überlappen, und umgekehrt. Folglich gibt es in $L \times \{c\} \times R$ für jeden Arm aus L bzw. R mindestens eine $\mathcal{A}_5$-Alternative, die diesen Arm und auf der anderen Seite keinen leeren Arm benutzt, sofern R bzw. L drei nicht leere Arme enthält.

Sind $\hat{l}_1, \ldots, \hat{l}_{|Arms_v|}$ die *nach Gewinn sortierten* Arme von v und entsprechend $\hat{r}_1, \ldots, \hat{r}_{|Arms_w|}$ die nach Gewinn sortierten Arme von w, so muss analog die größte $\mathcal{A}_5$-Alternative $(\hat{l}, c, \hat{r})$ unter $\{\hat{l}_1, \hat{l}_2, \hat{l}_3, \emptyset\} \times \{c\} \times \{\hat{r}_1, \hat{r}_2, \hat{r}_3, \emptyset\}$ zu finden sein. Der leere Arm wird benutzt, wenn sich alle anderen Arme einen Knoten teilen oder negativen Gewinn haben.

1. Fall: $\hat{l} \in L$ und $\hat{r} \in R$

Dann ist $\text{gain}(G(c)) \geq \text{gain}(\hat{l}, c, \hat{r})$, da $\text{gain}(G(c))$ über alle $\mathcal{A}_5$-Alternativen aus $L \times \{c\} \times R$ maximiert wird. Ist $\text{gain}(\hat{l}, c, \hat{r}) \geq 0$, so folgt $\text{gain}(G(c)) \geq$

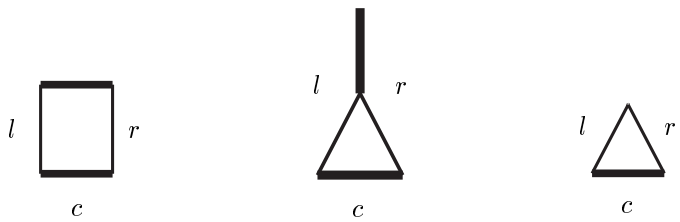

Abbildung 3.1: Sonderfälle durch Überschneidung der Arme

$\text{gain}(\hat{l},c,\hat{r}) \geq \frac{\text{gain}(\hat{l},c,\hat{r})}{1+\beta} \geq \frac{\text{gain}(\hat{l},c,\hat{r})-\frac{2}{n}\,\alpha\,\omega(M^*)-\beta\,\omega(c)}{1+\beta}$. Anderenfalls ist mit der letzten Zeile von `GetGood` ebenfalls $\text{gain}(G(c)) \geq 0 \geq \frac{\text{gain}(\hat{l},c,\hat{r})-\frac{2}{n}\,\alpha\,\omega(M^*)-\beta\,\omega(c)}{1+\beta}$

2. Fall: $\hat{l} \notin L$ und $\hat{r} \in R$ (bzw. analog $\hat{l} \in L$ und $\hat{r} \notin R$)

Enthält L nicht drei nicht-leere Arme, so ist $\hat{l}$ ebenfalls leer, was dann jedoch dem ersten Fall entspricht. Ansonsten bildet, wie oben festgestellt, $\hat{r} \in R$ mit mindestens einem $l_i \in L \setminus \emptyset$ als auch $\emptyset$ eine $\mathcal{A}_5$-Alternative. l sei das Maximum von $\emptyset$ und l_i bezüglich Gewinn. Da $Arms_v$ nach Rang sortiert ist, gilt $r(l_i) \geq r(\hat{l})$. Mit Korollar 11.i (S. 11) folgt:

$$\text{gain}(l) = \max(\text{gain}(\emptyset), \text{gain}(l_i) \geq \frac{\text{gain}(\hat{l}) - \frac{1}{n}\,\alpha\,\omega_{max}}{1+\beta}.$$

Nun gilt $\text{gain}(G(c)) \geq \text{gain}(l,c,\hat{r}) = \text{gain}(l)+\text{gain}(\hat{r})-\omega(c)$. Und da $\text{gain}(\hat{r}) \geq 0$ folgt:

$$\begin{aligned}\text{gain}(G(c)) &\geq \frac{\text{gain}(\hat{l}) - \frac{1}{n}\,\alpha\,\omega_{max}}{1+\beta} + \frac{\text{gain}(\hat{r})}{1+\beta} - \frac{(1+\beta)\,\omega(c)}{1+\beta} \\ &\geq \frac{\text{gain}(\hat{l},c,\hat{r}) - \frac{2}{n}\,\alpha\,\omega_{max} - \beta\,\omega(c)}{1+\beta}\end{aligned}$$

3. Fall: $\hat{l} \notin L$ und $\hat{r} \notin R$

Besteht L bzw. R nicht jeweils aus drei nicht-leeren Armen, so ist auch entsprechend $\hat{l}$ bzw. $\hat{r}$ leer und damit tritt der erste oder zweite Fall ein. Ansonsten gibt es, wie oben festgestellt, ein $l_i \in L \setminus \emptyset$ und ein $r_j \in R \setminus \emptyset$, so dass (l_i,c,r_j) eine $\mathcal{A}_5$-Alternative ist. Sei l das Maximum von $\emptyset$ und l_i und r das Maximum $\emptyset$ und r_j bezüglich Gewinn. Dann folgt analog:

$$\text{gain}(l) \geq \frac{\text{gain}(\hat{l}) - \frac{1}{n}\,\alpha\,\omega_{max}}{1+\beta} \quad \text{und} \quad \text{gain}(r) \geq \frac{\text{gain}(\hat{r}) - \frac{1}{n}\,\alpha\,\omega_{max}}{1+\beta}.$$

Analog gilt $\text{gain}(G(c)) \geq \text{gain}(l,c,r) = \text{gain}(l) + \text{gain}(r) - \omega(c)$ und dann:

$$\begin{aligned}\text{gain}(G(c)) &\geq \frac{\text{gain}(\hat{l}) - \frac{1}{n}\,\alpha\,\omega_{max}}{1+\beta} + \frac{\text{gain}(\hat{r}) - \frac{1}{n}\,\alpha\,\omega_{max}}{1+\beta} - \frac{(1+\beta)\,\omega(c)}{1+\beta} \\ &\geq \frac{\text{gain}(\hat{l},c,r_j) - \frac{2}{n}\,\alpha\,\omega_{max} - \beta\,\omega(c)}{1+\beta}.\end{aligned}$$

□

Es wird nun zu jedem Zentrum c mit `GetGood` eine $\mathcal{A}_5$-Alternative $G(c)$ berechnet. Dies sind maximal $n/2$ Werte, da es höchstens so viele Matchingkanten gibt. Diese Werte werden nun ebenfalls nach Gewinn partioniert mit den gleichen Parametern wie bei den Kreisen, also $f_{max} \leq 2\,\omega_{max}$. Dabei wird jede Partitionsklasse als Liste $\mathcal{P}_{r(G(c))}$ angelegt. Da die Elemente aus Partitionsklasse $\mathcal{P}_0$ im Folgenden nicht mehr vonnöten sind, werden sie gelöscht, d.h $\mathcal{P}_0 = \emptyset$. Mit Korollar 12 (S. 11) folgt ein Gesamtaufwand von $\mathcal{O}(n + \log_{1+\beta}\frac{n}{\alpha})$.

3.2 Der Algorithmus

Greedy-$\mathcal{A}_5$ ($G = (V, E)$, *maximales* M, $\omega : E \mapsto \mathbb{R}_+$)

```
(1)  Wiederhole
(2)      P_max := nicht leere Klasse P_i mit max. Rang
(3)      P := erster Pfad aus P_max
(4)      C := erster verfügbarer Kreis aus C
(5)      Resultat := Resultat ⊎ Maximum von C und P
(6)      ∀c ∈ M : Falls Pfad G(c) nicht mehr verfügbar ist, so:
(7)          Entferne G(c) aus der Partition
(8)          Berechne mit Getgood einen neuen verfügbaren Pfad
             G(c), falls vorhanden
(9)          Füge diesen in P_{r_min(c)} ein, falls r_min(c) > 0,
             wobei r_min(c) := min_{G(c)}(r(G(c)))[1]
(10) Bis C = P = ∅
(11) M := MaxMatch(M \ Resultat ⊎ Resultat \ M)
```

Der Algorithmus sammelt disjunkte Alternativen, die allesamt positiven Gewinn haben, da sowohl in $\mathcal{P}$ als auch in $\mathcal{C}$ keine Elemente der Partitionsklasse Null enthalten sind.

Nachdem in Zeile 5 eine Alternative ausgewählt wurde, sind ihre Knoten nicht mehr **verfügbar**. Entsprechend sind auch alle Alternativen, die solch einen Knoten beinhalten, nicht mehr verfügbar.

Lemma 15 *Der Algorithmus wählt nur knotendisjunkte Alternativen aus.*

Beweis. Zu Beginn sind alle Alternativen in $\mathcal{C}$ und $\mathcal{P}_i$ verfügbar. Aufgrund von Zeile 4 ist C immer verfügbar. Wegen Zeile 6-9 enthält $\mathcal{P}_i$ nur verfügbare Pfade, also ist auch P verfügbar. Damit wird in Zeile 5 eine verfügbare, d.h. zu allen zuvor ausgewählten Alternativen knotendisjunkte Alternative ausgewählt. □

3.2.1 Die Laufzeit

Lemma 16 *Es gibt höchstens $\mathcal{O}(n)$ Schleifendurchläufe.*

Beweis. Beim letzten Durchlauf gilt $C = P = \emptyset$. In diesem Fall wurde dem Resultat in Zeile 5 nichts hinzugefügt. Bei allen anderen Durchläufen war C oder P nicht leer, d.h. in Zeile 5 wurde mindestens eine Matchingkante ausgewählt. Da diese mit Lemma 15 alle disjunkt sind und es höchstens $n/2$ Matchingkanten gibt, folgt das Lemma. □

Lemma 17 *Das Berechnen von C benötigt insgesamt $\mathcal{O}(m)$ Laufzeit. C ist maximal bezüglich Rang.*

[1] Für das Zentrum c ist der Rang $r_{min}(c)$ der kleinste Rang, den eine Alternative $G(c)$ je hatte, berechnet von `GetGood` für das Zentrum c bei der Initialisierung oder in Zeile 8. Die neu berechnete Alternative wird also nicht in die Partitionsklasse $\mathcal{P}_{r(G(c))}$ eingefügt, sondern in $\mathcal{P}_{r_{min}(c)}$, denn der Rang des neuen $G(c)$ könnte größer sein als der Rang des vorigen $G(c)$ und die Suche nach $\mathcal{P}_{max}$ wäre dann nicht mehr amortisiert konstant. Siehe dazu auch Lemma 18.

Beweis. Die Verfügbarkeit eines C_4 kann mit konstantem Zeitaufwand entschieden werden, indem jeder im Resultat enthaltene Knoten als nicht mehr verfügbar markiert wird. Entsprechend wird in Zeile 4 solange das erste Element von $\mathcal{C}$ gelöscht, bis ein verfügbarer Kreis C gefunden wird oder $\mathcal{C}$ leer ist. Insgesamt werden dabei höchstens alle Elemente von $\mathcal{C}$ getestet, was mit Lemma 13 (S. 13) höchstens $m/2$ sind. C ist maximal bezüglich Rang, da $\mathcal{C}$ nach Rang sortiert ist. □

Lemma 18 *Das Berechnen von P benötigt insgesamt $\mathcal{O}(n+\log_{1+\beta}\frac{n}{\alpha})$ Laufzeit. Der Rang von P ist größer gleich dem Maximum von $\{i \mid \mathcal{P}_i \neq \emptyset\}$.*

Beweis. Nach der Initialsierung existiert zu jeder Matchingkante c genau ein Pfad $G(c)$, welcher in $\mathcal{P}_{r(G(c))}$ eingefügt wurde, falls $r(G(c)) > 0$. Damit ist $r_{min}(c)$ zu Beginn gerade $r(G(c))$. Im Laufe des Algorithmus wird der eingefügte Pfad $G(c)$ gegebenenfalls in Zeile 9 durch einen neuen ersetzt, welcher dann in $\mathcal{P}_{r_{min}(c)}$ eingefügt wird und $\mathcal{P}_{r_{min}(c)}$ ist damit nicht leer, solange $r_{min}(c) > 0$. Dadurch ist $r_{min}(c)$ immer kleiner gleich dem daraufhin ermittelten Index von $\mathcal{P}_{max}$, da dies der größte Index aller nicht leeren Partitionsklassen ist. Da $r_{min}(c)$ im Laufe des Algorithmus monoton fällt und neue $G(c)$ stets nur in $\mathcal{P}_{r_{min}(c)}$ eingefügt werden, bleiben folglich die Partitionsklassen oberhalb von $\mathcal{P}_{max}$ immer leer und die Partitionsklasse $\mathcal{P}_{max}$ in Zeile 2 entspricht der Partitionsklasse $\mathcal{P}_{max}$ des vorhergehenden Schleifendurchlaufes oder liegt unterhalb dieser Partitionsklasse. Insgesamt werden bei der Suche nach $\mathcal{P}_{max}$ also maximal alle $\mathcal{O}(\log_{1+\beta}\frac{n}{\alpha})$ Partitionsklassen einmal abgelaufen. P selbst wird dann pro Durchlauf mit konstantem Zeitaufwand gefunden.

Da beim Einfügen während der Initialisierung oder in Zeile 9 der Rang von $G(c)$ immer größer gleich $r_{min}(c)$ ist, hat auch P einen Rang größer gleich dem Index von $\mathcal{P}_{max}$, was jedoch gerade dem größten Index aller nicht leeren Partitionsklassen entspricht. □

Lemma 19 *Die Aktualisierung der $\mathcal{P}_i$ benötigt insgesamt $\mathcal{O}(m+n)$ Laufzeit.*

Beweis. Wird eine Alternative ausgewählt, so werden alle dazu nicht knotendisjunkten $\mathcal{A}_5$-Alternativen in Zeile 6-9 entfernt und neue mit gleichem Zentrum, soweit vorhanden, eingefügt.

Dazu werden zunächst alle Arme aus den *Arms*-Listen gelöscht, die durch die Auswahl in Zeile 5 nicht mehr verfügbar sind, d. h. ein Knoten in der ausgewählten Alternative enthalten ist. Ist der Knoten $m_u(u)$ des Arms $\{m_u, uv\}$ ausgewählt, so auch u, da stets Alternativen ausgewählt werden und somit auch beide Knoten einer Matchingkante. Folglich ist ein Arm $\{m_u, uv\}$ genau dann betroffen, wenn v oder u ausgewählt werden. In beiden Fällen ist der ausgewählte Knoten inzident zur Nicht-Matching-Kante uv. Somit können alle betroffenen Arme enumeriert werden, indem zu jedem ausgewählten Knoten v alle inzidenten Nicht-Matching-Kanten und die dazu gehörenden Arme untersucht und aus den Listen gelöscht werden, wenn diese noch in den *Arms*-Listen enthalten sind. Dies kann mit konstantem Zeitaufwand pro Arm geschehen, wenn die *Arms*-Listen doppelt verkettet sind und – da jede Nicht-Matching-Kante in höchstens zwei Armen vorkommt – bei jeder Nicht-Matching-Kante ein Verweis auf jeden der beiden *Arms*-Einträge vermerkt ist. Da jeder Knoten nur einmal ausgewählt

wird und es insgesamt nur $2m$ Nachbarn gibt, benötigt die Aktualisierung der $Arms$-Listen folglich einen Zeitaufwand von insgesamt $\mathcal{O}(m)$.

Eine Alternative ist von der Aktualisierung betroffen, wenn ihr Zentrum oder einer ihrer beiden Arme nicht mehr verfügbar ist. Da jede der $\mathcal{O}(n)$ Matchingkanten höchstens einmal ausgewählt wird, sind auch höchstens $\mathcal{O}(n)$ Zentren nicht mehr verfügbar. In diesem Fall (betrifft insbesondere die ausgewählte Alternative) wird $G(c)$ aus $\mathcal{P}_{r_{min}(c)}$ gelöscht. Das kann mit konstantem Zeitaufwand geschehen, wenn $\mathcal{P}_{r_{min}(c)}$ eine doppelt verkettete Liste ist und jedes Zentrum c einen Verweis auf seinen $G(c)$-Eintrag besitzt. Da c nicht mehr verfügbar ist, kann in Zeile 8 kein neuer verfügbarer Pfad mit Zentrum c gefunden werden. Da ebenso jeder Arm genau einmal aus $Arms_v$ gelöscht wird, also nicht mehr verfügbar ist, und es mit Lemma 1 (S. 4) höchstens $2m$ Arme gibt, sind auch nur $\mathcal{O}(m)$ Alternativen wegen nicht mehr verfügbaren Armen betroffen. Auch in diesem Fall wird $G(c)$ aus $\mathcal{P}_{r_{min}(c)}$ gelöscht und mit `GetGood` nach Lemma 14 (S. 14) mit konstantem Zeitaufwand eine neue verfügbare $\mathcal{A}_5$-Alternative oder leere Menge $G(c)$ berechnet. Diese wird in $\mathcal{P}_{r_{min}(c)}$ eingefügt, wenn $r_{min}(c)$ größer Null ist. □

Satz 20 *Der* `Greedy-`$\mathcal{A}_5$ *hat eine Laufzeit von* $\mathcal{O}(m + n + \log_{1+\beta} \frac{n}{\alpha})$.

Beweis. Die Initalisierung benötigt einen Aufwand von $\mathcal{O}(m + n + \log_{1+\beta} \frac{n}{\alpha})$. Der Algorithmus selbst benötigt mit Lemma 16, 17, 18 und 19 insgesamt $\mathcal{O}(m + n + \log_{1+\beta} \frac{n}{\alpha})$. □

3.2.2 Die Korrektheit

Satz 21 *Der vom Algorithmus erzielte Gewinn ist*

$$\sum_{A_{alg} \in Resultat} \text{gain}(A_{alg}) \geq \frac{1}{3\,(1+\beta)^2} \left(\left(\frac{2}{3} - \frac{7+3\,\beta}{3} \alpha \right) \omega(M^*) - (1+\beta)\,\omega(M) \right)$$

Beweis. Die Gewichtsdifferenz zwischen M^* und M kann durch die Alternativen in $\mathcal{A}^*$ abgeschätzt werden, welche wiederum durch die vom Algorithmus gewählten Alternativen approximiert werden. Im Folgenden wird jede Alternative aus A^* abgeschätzt, wobei sich $\mathcal{A}^*$ disjunkt zerlegt in $\mathcal{A}^*_{C_4}$ und $\mathcal{A}^*_{\infty}$, und anschließend die Gleichungen zusammengeführt.

Alle Alternativen A sind solange verfügbar bis einer ihrer Knoten vom Algorithmus ausgewählt wird, sich also eine Alternative des Resultats mit A überlappt. Diese wird als $\text{Cut}(A)$ bezeichnet. Ist A bis zum Schluß verfügbar, so ist $\text{Cut}(A)$ leer. Da $\text{Cut}(A)$ im Resultat enthalten oder leer ist, folgt $\text{gain}(\text{Cut}(A)) \geq 0$.

1. Fall: $\mathcal{A}^*_{C_4}$

Sei also C ein alternativer C_4-Kreis aus $\mathcal{A}^*_{C_4}$. Ist sein Rang Null, so folgt mit Korollar 11.ii (S. 11):

$$\text{gain}(\text{Cut}(C)) \geq 0 \geq (\text{gain}(C) - \frac{1}{n}\,\alpha\,2\,w_{max})/(1+\beta)^2. \tag{3.1}$$

Ist hingegen sein Rang größer Null, so ist er im Rahmen der Initialisierung in der nach Rang sortierten Liste $\mathcal{C}$ eingefügt worden. Solange C verfügbar und damit in $\mathcal{C}$ enthalten ist, ist der Rang des jeweils ersten verfügbaren Kreises $C_1 \in \mathcal{C}$ damit größer gleich dem von C. Mit Korollar 11.ii (S. 11) folgt:

$$\text{gain}(C_1) \geq (\text{gain}(C) - \frac{1}{n}\,\alpha\,2\,w_{max})/(1+\beta)^2.$$

Wird die Alternative $\text{Cut}(C)$ in Zeile 5 ausgewählt, so ist mit Zeile 4 und 5 ihr Gewinn größer gleich dem des im Moment ersten Kreises aus $\mathcal{C}$, also

$$\text{gain}(\text{Cut}(C)) \geq \text{gain}(C_1).$$

Für alle $C \in \mathcal{A}^*_{C_4}$ zusammen gilt (mit Gleichung 3.1) folglich:

$$\sum_{C \in \mathcal{A}^*_{C_4}} \text{gain}(\text{Cut}(C)) \geq \sum_{C \in \mathcal{A}^*_{C_4}} \frac{\text{gain}(C) - \frac{1}{n}\,\alpha\,2\,w_{max}}{(1+\beta)^2}.$$

Weil jeder Kreis C zwei Matchingkanten enthält und es nur $n/2$ Matchingkanten gibt, ist die Anzahl alternativer C_4-Kreise in $\mathcal{A}^*_{C_4}$, also $\|\mathcal{A}^*_{C_4}\|$, kleiner gleich $n/4$. Und mit $w_{max} \leq \omega(M^*)$ folgt:

$$\sum_{C \in \mathcal{A}^*_{C_4}} \text{gain}(\text{Cut}(C)) \geq \frac{\left(\sum_{C \in \mathcal{A}^*_{C_4}} \text{gain}(C)\right) - \frac{\alpha}{2}\,\omega(M^*)}{(1+\beta)^2}.$$

Nun ist $\sum_{C \in \mathcal{A}^*_{C_4}} \text{gain}(C) = \omega(M^* \cap \mathcal{A}^*_{C_4}) - \omega(M \cap \mathcal{A}^*_{C_4})$ und dann (zur späteren Verwendung etwas abgeschwächt):

$$2 \cdot \sum_{C \in \mathcal{A}^*_{C_4}} \text{gain}(\text{Cut}(C)) \geq \frac{2 \cdot \omega(M^* \cap \mathcal{A}^*_{C_4}) - 3 \cdot \omega(M \cap \mathcal{A}^*_{C_4}) - \alpha\,\omega(M^*)}{(1+\beta)^2}. \quad (3.2)$$

2. Fall: $\mathcal{A}^*_\infty$

Betrachtet werden nun die Alternativen in $\mathcal{A}^*_\infty$. Mit Lemma 4 (S. 6) gibt es eine Überdeckung Ω_5 von $\mathcal{A}^*_\infty$ durch $\mathcal{A}_5$-Alternativen, so dass

$$\sum_{A \in \Omega_5} \text{gain}(A) \geq 2 \cdot \omega(M^* \cap \mathcal{A}^*_\infty) - 3 \cdot \omega(M \cap \mathcal{A}^*_\infty). \quad (3.3)$$

Sei A solch eine Alternative mit Zentrum c_A und zwei (mitunter leeren oder reduzierten) Armen gemäß Lemma 2 (S. 5). Zu jeder Matchingkante c wurde im Rahmen der Initialisierung mittels `GetGood` ein (mitunter leerer) Pfad $G(c)$ berechnet, der eventuell im Laufe des Algorithmus mehrmals mittels `GetGood` aktualisiert wurde. Solange die Alternative A verfügbar ist und ihre Arme damit in den *Arms*-Listen enthalten sind, gilt für jedes so berechnete $G(c_A)$ mit Lemma 14 (S. 14):

$$\text{gain}(G(c_A)) \geq \frac{\text{gain}(A) - \frac{2}{n}\,\alpha\,\omega(M^*) - \beta\,\omega(c_A)}{1+\beta}.$$

Diese Ungleichung gilt, solange A verfügbar ist, insbesondere für den bezüglich Gewinn kleinsten bisher von `GetGood` für c_A berechneten Pfad $G_{min}(c_A)$ mit Rang $r_{min}(c_A)$. Mit Korollar 5 (S. 8) gilt $\|\Omega_5\| \leq 3\,n/2$ und mit Korollar 6 (S. 8) gilt $\sum_{A\in\Omega_5} \omega(c_A) \leq 3\,\omega(M)$, da jede Matchingkante maximal dreimal in der Überdeckung vorkommt. Solange sich jedes G_{min} auf ein verfügbares A bezieht, folgt damit:

$$\sum_{A\in\Omega_5} \text{gain}(G_{min}(c_A)) \geq \frac{\left(\sum_{A\in\Omega_5} \text{gain}(A)\right) - 3\,\alpha\,\omega(M^*) - 3\,\beta\,\omega(M)}{1+\beta}. \tag{3.4}$$

Ist $r_{min}(c_A)$ gleich Null, so folgt mit Korollar 11.i (S. 11):

$$0 \geq (\text{gain}(G_{min}(c_A)) - \frac{\alpha}{n}\,2\,w_{max})/(1+\beta). \tag{3.5}$$

Ist $r_{min}(c_A)$ hingegen größer als Null, so ist die Partitionsklasse $\mathcal{P}_{r_{min}(c_A)}$ nicht leer. Der in Zeile 3 jeweils ermittelte erste Pfad P hat dann mit Lemma 18 (S. 17) einen Rang größer gleich $r_{min}(c_A)$, so dass mit Korollar 11.i (S. 11) gilt:

$$\text{gain}(P) \geq (\text{gain}(G_{min}(c_A)) - \frac{1}{n}\,\alpha\,2\,w_{max})/(1+\beta). \tag{3.6}$$

Wird die Alternative $\text{Cut}(A)$ in Zeile 5 ausgewählt, so ist mit Zeile 3 und 5 ihr Gewinn größer gleich dem des im Moment ersten Pfades $P \in \mathcal{P}_{max}$, also

$$\text{gain}(\text{Cut}(A)) \geq \text{gain}(P).$$

Zusammen mit den Gleichungen 3.5 und 3.6 folgt

$$\sum_{A\in\Omega_5} \text{gain}(\text{Cut}(A)) \geq \sum_{A\in\Omega_5} \frac{\text{gain}(G_{min}(c_A)) - \frac{1}{n}\,\alpha\,2\,w_{max}}{1+\beta}$$

und da $w_{max} \leq \omega(M^*)$ und mit Korollar 5 (S. 8) $\|\Omega_5\| \leq 3\,n/2$ folgt:

$$\sum_{A\in\Omega_5} \text{gain}(\text{Cut}(A)) \geq \frac{\left(\sum_{A\in\Omega_5} \text{gain}(G_{min}(c_A))\right) - 3\,\alpha\,\omega(M^*)}{(1+\beta)}.$$

Bis $\text{Cut}(A)$ ausgewählt wird, ist A noch verfügbar. $G_{min}(c_A)$ bezieht sich also auf ein verfügbares A und somit kann nun Gleichung 3.4 eingesetzt werden:

$$\sum_{A\in\Omega_5} \text{gain}(\text{Cut}(A)) \geq \frac{\left(\sum_{A\in\Omega_5} \text{gain}(A)\right) - (6+3\,\beta)\,\alpha\,\omega(M^*) - 3\,\beta\,\omega(M)}{(1+\beta)^2}$$

und mit Gleichung 3.3 folgt:

$$\sum_{A\in\Omega_5} \text{gain}(\text{Cut}(A)) \geq \frac{2\,\omega(M^*\cap\mathcal{A}^*_\infty) - 3\,\omega(M\cap\mathcal{A}^*_\infty) - (6+3\,\beta)\,\alpha\,\omega(M^*) - 3\,\beta\,\omega(M)}{(1+\beta)^2}.$$

Es werden nun beide Fälle zusammengeführt. Zusammen mit der im ersten Fall hergeleiteten Ungleichung 3.2 für Kreise folgt:

$$\sum_{A\in\Omega_5} \text{gain}(\text{Cut}(A)) + \sum_{C\in\mathcal{A}^*_{C_4}} 2\cdot\text{gain}(\text{Cut}(C)) \geq$$
$$\frac{2\,\omega(M^*\cap A^*) - 3\,\omega(M\cap A^*) - (7+3\,\beta)\,\alpha\,\omega(M^*) - 3\,\beta\,\omega(M)}{(1+\beta)^2}$$

Mit $\mathcal{A}^* = (M^* \cup M) \setminus (M^* \cap M)$ folgt $\omega(M^* \cap \mathcal{A}^*) = \omega(M^* \setminus M^* \cap M) = \omega(M^*) - \omega(M^* \cap M)$. Analog ist $\omega(M \cap \mathcal{A}^*) = \omega(M) - \omega(M^* \cap M)$ und damit

$$2 \cdot \omega(M^* \cap \mathcal{A}^*) - 3 \cdot \omega(M \cap \mathcal{A}^*) \geq 2 \cdot \omega(M^*) - 3 \cdot \omega(M) + \omega(M^* \cap M).$$

Enthält eine Alternative einen Knoten u, so auch die zugehörige Matchingkante m_u, falls diese existiert. Eine im Resultat enthaltene Alternative besitzt zusammengenommen höchstens drei nicht zum Matching gehörende Knoten oder aber Matchingkanten. Mit jeder dieser Knoten oder Matchingkanten kann sie entweder genau einen Kreis in $\mathcal{A}^*_{C_4}$ oder, mit Lemma 4 (S. 6) und Korollar 6 (S. 8), höchstens drei Alternativen aus Ω_5 überlappen. Damit kann jede im Resultat enthaltene Alternative höchstens neunmal auf der linken Seite der obigen Ungleichung als $\mathrm{Cut}(A)$ bzw. $\mathrm{Cut}(C)$ auftauchen. Da jedes $\mathrm{Cut}(A)$ entweder leer oder im Resultat enthalten ist und alle Alternativen des Resultats positiven Gewinn haben, folgt:

$$9 \cdot \sum_{A \in Resultat} \mathrm{gain}(A) \geq \frac{2\,\omega(M^*) - 3\,\omega(M) - (7+3\,\beta)\,\alpha\,\omega(M^*) - 3\,\beta\,\omega(M)}{(1+\beta)^2}$$

□

3.3 Die Approximation von $(2/3-\epsilon)\,\omega(M^*)$

Dazu wird der `Greedy-`$\mathcal{A}_5$ wiederholt ausgeführt. Pro Aufruf erzielt er mit Satz 21 (S. 18) einen Gewinn von

$$\sum_{A_{alg} \in Resultat} \mathrm{gain}(A_{alg}) \geq \frac{1}{3\,(1+\beta)^2} \left(\left(\frac{2}{3} - \frac{7+3\,\beta}{3} \alpha \right) \omega(M^*) - (1+\beta)\,\omega(M) \right)$$

Entsprechend gilt für das maximale Matching M_i nach dem i-ten Schritt mit $\rho = \frac{1}{3\,(1+\beta)}$ und $\mu = \frac{\rho}{1+\beta} \left(\frac{2}{3} - \frac{7+3\,\beta}{3}\,\alpha \right) \omega(M^*)$:

$$\omega(M_i) \geq \omega(M_{i-1}) + \sum_{A_{alg} \in Resultat} \mathrm{gain}(A_{alg}) \geq (1-\rho)\,\omega(M_{i-1}) + \mu$$

und damit:

$$\begin{aligned}
\omega(M_0) &\geq 0 \\
\omega(M_1) &\geq (1-\rho)\ \omega(M_0) + \mu \\
\omega(M_2) &\geq (1-\rho)^2 \omega(M_0) + (1-\rho)\ \mu + \mu \\
\omega(M_3) &\geq (1-\rho)^3 \omega(M_0) + (1-\rho)^2 \mu + (1-\rho)\mu + \mu \\
&\vdots \qquad\qquad\qquad \vdots \\
\omega(M_k) &\geq (1-\rho)^k \omega(M_0) + \sum_{i=0}^{k-1} (1-\rho)^i \mu, \quad \text{und mit } \rho < 1 \text{ und } \omega(M_0) \geq 0 \\
\omega(M_k) &\geq \frac{1-(1-\rho)^k}{1-(1-\rho)} \cdot \mu = \frac{1-(1-\rho)^k}{\rho} \cdot \frac{\rho}{1+\beta} \left(\frac{2}{3} - \frac{7+3\,\beta}{3}\,\alpha \right) \omega(M^*) \\
\omega(M_k) &\geq \frac{1-\left(1-\frac{1}{3\,(1+\beta)}\right)^k}{1+\beta} \left(\frac{2}{3} - \frac{7+3\,\beta}{3}\,\alpha \right) \omega(M^*)
\end{aligned}$$

Mit $\epsilon < 2/3$ werden nun $\alpha = \beta = \epsilon/9 < 2/27$ fixiert. Weiterhin sei $k > \log_{29/20} 1/\epsilon$, also $\epsilon \geq \left(\frac{20}{29}\right)^k = \left(1 - \frac{1}{3\,(1+2/27)}\right)^k \geq \left(1 - \frac{1}{3\,(1+\beta)}\right)^k$ und damit

$$\begin{aligned}
\omega(M_k) &\geq \frac{1-\epsilon}{1+\epsilon/9}\left(\frac{2}{3} - \frac{7+3\,\epsilon/9}{3}\,\epsilon/9\right)\omega(M^*) \\
&= \frac{\frac{2}{3} - \frac{25}{27}\epsilon + \frac{20}{81}\epsilon^2 + \frac{1}{81}\epsilon^3}{1+\epsilon/9}\,\omega(M^*) \\
&> \frac{(\frac{2}{3}-\epsilon)(1+\epsilon/9)}{1+\epsilon/9}\,\omega(M^*) = (2/3-\epsilon)\,\omega(M^*).
\end{aligned}$$

Mit Satz 20 (S. 18) wird die erforderliche Güte in einer Laufzeit von $\mathcal{O}((m+n+\log_{1+\beta}\frac{n}{\alpha})\cdot k)$ erreicht.

$$\begin{aligned}
\log_{1+\beta}\frac{n}{\alpha} &= \log_{1+\epsilon/9}\frac{n}{\epsilon/9} = \frac{\ln 9\,n/\epsilon}{\ln(1+\epsilon/9)} < \frac{18}{\epsilon}\cdot\ln\frac{9n}{\epsilon}, \quad \text{da} \\
\ln(1+\epsilon/9) &= \underbrace{\frac{\epsilon}{9} - \frac{1}{2}\left(\frac{\epsilon}{9}\right)^2}_{>0} + \underbrace{\frac{1}{3}\left(\frac{\epsilon}{9}\right)^3 - \frac{1}{4}\left(\frac{\epsilon}{9}\right)^4}_{>0} + \ldots > \frac{\epsilon}{9} - \frac{1}{2}\left(\frac{\epsilon}{9}\right)^2 > \frac{\epsilon}{18} \\
&\quad \text{mit} \quad \frac{1}{i}\left(\frac{\epsilon}{9}\right)^i > \frac{1}{i+1}\left(\frac{\epsilon}{9}\right)^{i+1}.
\end{aligned}$$

Dies ergibt eine Gesamtlaufzeit von $\mathcal{O}((m+n+\frac{1}{\epsilon}\log\frac{n}{\epsilon})\cdot\log\frac{1}{\epsilon})$. Ist $\epsilon \geq \frac{\log n}{m}$, so ergibt dies eine Laufzeit von $\mathcal{O}((m+n)\log 1/\epsilon)$, was der Laufzeit des Algorithmus von Pettie und Sanders entspricht bzw. bezogen auf die Genauigkeit ϵ eine exponentiell schnellere Geschwindigkeit verspricht im Vergleich zum Algorithmus von Drake und Hougardy mit $\mathcal{O}((n+m)\cdot 1/\epsilon)$ mit gleicher Güte. Man beachte, dass bei zu kleinem ϵ der exakte Algorithmus von Gabow [14] mit $\mathcal{O}(mn + n^2\log n)$ schneller ist.

Kapitel 4

Der Greedy-$\mathcal{A}_7$-Algorithmus

Die Gewichtsdifferenz zwischen M^* und M kann durch die Alternativen in $\mathcal{A}^*$ abgeschätzt werden. $\mathcal{A}^*$ zerfällt disjunkt in $\mathcal{A}^*_{C_4}$ und $\mathcal{A}^*_\infty$. Der `Greedy-`$\mathcal{A}_5$ wählte Alternativen aus, welche die Kreise aus $\mathcal{A}^*_{C_4}$ und die $\mathcal{A}_5$-Alternativen aus der Überdeckung von $\mathcal{A}^*_\infty$ dominierten. Analog wird $\mathcal{A}^*_\infty$ nun durch $\mathcal{A}_7$-Alternativen überdeckt, so dass:

$$\sum_{A \in \Omega_7} \text{gain}_7^{\vec{c}_A}(A) \geq 3 \cdot \omega(M^* \cap \mathcal{A}^*_\infty) - 4 \cdot \omega(M \cap \mathcal{A}^*_\infty) \quad \text{(Satz 7, S. 8)}.$$

Die Gewinn$_7$-Funktion wird benutzt, da in $\mathcal{A}_7$ auch C_6-Kreise enthalten sind, auf denen sie sich zum normalen Gewinn unterscheidet.

In diesem Fall schneidet eine Alternative aus dem Resultat höchstens sechzehn Alternativen aus $\mathcal{A}^*_{C_4}$ bzw. Ω_7 und unter Berücksichtigung der Approximationsfehler wird diesmal mit Satz 34 (S. 34) gefolgert:

$$16 \cdot \sum_{A \in Resultat} \text{gain}(A) \geq \frac{3 \cdot \omega(M^*) - 4 \cdot \omega(M) - (8 + 6\,\beta)\,\alpha\,\omega(M^*) - 4\,\beta\,\omega(M)}{(1+\beta)^2}$$

und am Ende erhält man einen Mindestgewinn des Resultats von ungefähr $\frac{1}{4}\left(\frac{3}{4}\,\omega(M^*) - \omega(M)\right)$.

Der `Greedy-`$\mathcal{A}_5$ stellte bei jedem Schleifendurchlauf eine Approximation der größten verfügbaren $\mathcal{A}_5$-Alternative sicher. Analog muss der `Greedy-`$\mathcal{A}_7$ eine Approximation der größten verfügbaren $\mathcal{A}_7$-Alternative in annehmbarer Laufzeit bereitstellen. Dazu *sortiert* der `Greedy-`$\mathcal{A}_7$ die Arme nach ihrem *Gewinn* und partitioniert die möglichen Erweiterungen. Auf diese Weise kann er mit geringem Fehler zu jeder orientierten Matchingkante $\vec{c}$ die größte $\mathcal{A}_5$-Alternative mit Zentrum $\vec{c}$ approximieren, die eine linksseitige Erweiterung vom Rang i besitzt. Diese Approximationen werden ebenfalls nach ihrem Gewinn$_7$ partitioniert und damit kann die jeweils größte noch verfügbare $\mathcal{A}_7$-Alternative approximiert werden.

Die wiederholte Anwendung des `Greedy-`$\mathcal{A}_7$ führt zu einem Matching mit einem Gewicht von $(\frac{3}{4} - \epsilon)\,\omega(M^*)$ und benötigt dazu eine Laufzeit von $\mathcal{O}((n+m) \cdot \frac{1}{\epsilon} \log \frac{n}{\epsilon} \log \frac{1}{\epsilon})$.

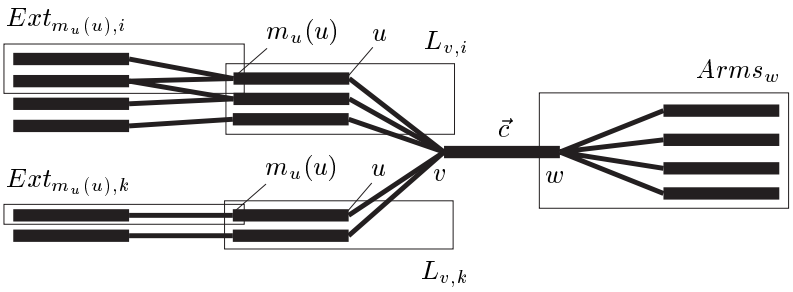

Abbildung 4.1: Die Datenstrukturen

4.1 Die Initialisierung

Im Gegensatz zum `Greedy-`$\mathcal{A}_5$ werden hier die C_4-Kreise und die Arme nach *Gewinn* sortiert. Desweiteren werden die Erweiterungen eines Knotens partitioniert und die Arme werden in verschiedene Listen eingefügt, abhängig davon, ob sie eine Erweiterung eines bestimmten Ranges besitzen. Zudem werden zu jeder Matchingkante $\vec{c}$ und jedem Rang i ein maximaler $\mathcal{A}_5$-Pfad mit Zentrum $\vec{c}$ berechnet, der eine linksseitige Erweiterung vom Rang i besitzt, mit der er zunächst auch zu einer $\mathcal{A}_7$-Alternative erweitert wird. Diese Alternativen werden wiederum nach ihrem Gewinn$_7$ partitioniert. Der Algorithmus `GetGood` mit amortisiert konstantem Zeitaufwand zum Berechnen einer solchen Alternative wird innerhalb dieses Abschitts vorgestellt.

4.1.1 Die Sortierung der C_4-Kreise

Wie beim `Greedy-`$\mathcal{A}_5$ werden alle alternativen C_4-Kreise enumeriert. Da es mit Lemma 13 (S. 13) nur $m/2$ dieser Kreise gibt, werden diese hier nach Gewinn absteigend sortiert und in die Liste $\boldsymbol{\mathcal{C}}$ eingefügt, wenn ihr Gewinn positiv ist. Dazu wird insgesamt eine Laufzeit von $\mathcal{O}(m \log m) \subseteq \mathcal{O}(m \log n)$ benötigt.

4.1.2 Die Sortierung der Arme

Für jeden Knoten v wird eine Liste $\boldsymbol{Arms_v}$ erzeugt, welche im Gegensatz zum `Greedy-`$\mathcal{A}_5$ die nach *Gewinn* sortierten Arme von v enthält. Dazu werden zunächst vorübergehend alle Arme sortiert. Anschließend wird einmal durch diese Sortierung gelaufen und jeder Arm in die entsprechende (noch leere) Liste $Arms_v$ eingefügt. Die Arme in $Arms_v$ sind somit ebenfalls sortiert. Mit Lemma 1 (S. 4) folgt ein Gesamtaufwand von $\mathcal{O}(m \log m) \subseteq \mathcal{O}(m \log n)$.

Korollar 22 *Es gibt pro Knoten v genau eine Liste $Arms_v$ und jeder Arm ist in genau einer dieser Listen enthalten.* □

4.1.3 Die Partition der Erweiterungen

Die Erweiterungen[1] eines Knotens v werden partitioniert. Dabei werden die Erweiterungen x_v von v mit Rang $i > 0$ in die Liste $\boldsymbol{Ext_{v,i}}$ eingefügt. Mit $l = \{m_u, uv\}$ wird $Ext_{m_u(u),i}$ auch als $\boldsymbol{Ext_{l,i}}$, also die Erweiterungen von l vom Rang i, bezeichnet. Die Partitionsparameter α und β werden später spezifiziert. Weiterhin entspricht n der Anzahl der Knoten des Graphen. Als Partitionsfunktion wird $f(a) := \text{gain}(a)$ benutzt und damit $f_{max} \leq \omega_{max}$. Ist d_v die Anzahl der Erweiterungen von v, so folgt mit Korollar 12 (S. 11) ein Aufwand pro Knoten von $\mathcal{O}(d_v + \log_{1+\beta} \frac{n}{\alpha})$, nach Lemma 1 (S. 4) demnach ein Gesamtaufwand von $\mathcal{O}(m + n \log_{1+\beta} \frac{n}{\alpha})$. Da die Arme von v bereits nach Gewinn sortiert in $Arms_v$ vorliegen, sind sie auch nach Rang sortiert und brauchen zur Erstellung der Partition nur einmal durchlaufen werden. Die Partitionsklassen sind somit zusätzlich nach Gewinn sortiert.

Korollar 23 *Es gibt pro Knoten v $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ $Ext_{v,i}$-Listen und jeder Arm ist in genau einer dieser Listen enthalten.* □

4.1.4 Die Erweiterbarkeit der Arme

Jeder Arm $l = \{m_u, uv\}$ von v wird in die Liste $\boldsymbol{L_{v,i}}$ eingefügt, wenn er eine Erweiterung vom Rang $i > 0$ hat, also wenn $Ext_{m_u(u),i} \neq \emptyset$. Dies kann für jeden Rang mit konstantem Zeitaufwand getestet werden. Mit Korollar 12 (S. 11) gibt es $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ verschiedene Ränge bzw. Tests pro Arm, also folgt ein Gesamtaufwand von $\mathcal{O}((n+m) \log_{1+\beta} \frac{n}{\alpha})$, um die Listen zu erstellen. Durchläuft man dabei die Arme in der Reihenfolge wie in den $Arms_v$-Listen vorgegeben, so ist auch $L_{v,i}$ nach Gewinn sortiert.

Korollar 24 *Es gibt pro Knoten v $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ $L_{v,i}$-Listen und jeder Arm ist in höchstens allen Listen genau eines Knotens enthalten.* □

4.1.5 Die Partition der $\mathcal{A}_5$-Pfade

Ähnlich zum `Greedy-`$\mathcal{A}_5$ werden auch hier $\mathcal{A}_5$-Pfade partitioniert, die zuvor jedoch provisorisch[2] zu $\mathcal{A}_7$-Alternativen erweitert werden. Außerdem werden zu jedem orientierten Zentrum $\vec{c}$ nicht nur ein, sondern $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ verschiedene Alternativen berechnet. Und zwar wird zu jedem Rang i der maximale $\mathcal{A}_5$-Pfad mit Zentrum $\vec{c}$ gesucht, der eine Erweiterung des linken Arms vom Rang i besitzt. Dies approximiert den maximalen $\mathcal{A}_7$-Pfad mit Zentrum $\vec{c}$ und linksseitiger Erweiterung vom Rang i.

Analog zum `Greedy-`$\mathcal{A}_5$ sind am Anfang alle Knoten **verfügbar**, jedoch kann ein Knoten ab einem bestimmten Zeitpunkt nicht mehr verfügbar sein (siehe dazu Abschnitt 4.2, S. 31). Eine Alternative ist entsprechend verfügbar, wenn alle ihre Knoten verfügbar sind. Um nur verfügbare Alternativen zu finden, wird im Folgenden davon ausgegangen, dass die Listen $Arms_v$, $L_{v,i}$ und $Ext_{v,i}$ nur verfügbare Arme enthalten, was mit der Aktualisierung der Listen sichergestellt wird (s. Lemma 32, S. 32).

[1] Nach Definition sind dies gerade die Arme von v.
[2] Die Erweiterung ist provisorisch, da diese später eventuell durch eine andere ersetzt wird.

GetGood($\vec{c} = (vw) \in M$, Rang i)

(1) Wenn $i > 0$:
(2) Finde eine bzgl. Gewinn maximale $\mathcal{A}_5$-Alternative $\boldsymbol{G_i(\vec{c})} = (l, \vec{c}, r) \in L_{v,i} \times \{\vec{c}\} \times (Arms_w \cup \{\emptyset\})$, falls vorhanden, für die eine Erweiterung x_l von l vom Rang i existiert, die mit $G_i(\vec{c})$ eine $\mathcal{A}_7$-Alternative bildet, wobei $\boldsymbol{G_i^+(\vec{c})}$ mit der kleinstmöglichen (provisorischen) Erweiterung vom Rang i gebildet wird.
(3) Wenn $i = 0$:
(4) Finde eine bzgl. Gewinn maximale $\mathcal{A}_5$-Alternative $\boldsymbol{G_i(\vec{c})} = (l, \vec{c}, r) \in (Arms_v \cup \{\emptyset\}) \times \{\vec{c}\} \times (Arms_w \cup \{\emptyset\})$ und $\boldsymbol{G_i^+(\vec{c})} = G_i(\vec{c})$

Lemma 25 *Zu gegebenem verfügbaren Zentrum $\vec{c} = (vw)$ kann ein maximaler verfügbarer Pfad $G_0(\vec{c}) = (l, \vec{c}, r) \in (Arms_v \cup \{\emptyset\}) \times \{\vec{c}\} \times (Arms_w \cup \{\emptyset\})$ mit konstantem Zeitaufwand gefunden werden.*

Beweis. Die jeweils größten Arme aus $Arms_v \cup \{\emptyset\}$ und $Arms_w \cup \{\emptyset\}$ bilden die gesuchte Alternative, es sei denn sie schneiden sich wie in Abbildung 3.1 (S. 14). Sei $\hat{l}$ bzw. $\hat{r}$ der größte Arm aus $Arms_v \cup \{\emptyset\}$ bzw. $Arms_w \cup \{\emptyset\}$ und $\hat{v}$ ein gemeinsamer Knoten. $\bar{l}$ bzw. $\bar{r}$ sei dann ein größter Arm aus $Arms_v \cup \{\emptyset\}$ bzw. $Arms_w \cup \{\emptyset\}$, der $\hat{v}$ nicht benutzt. $\hat{v}$ kann höchstens in zwei linken und zwei rechten Armen von $\vec{c}$ auftreten, $\bar{l}$ und $\bar{r}$ sind also unter den drei größten linken bzw. rechten Armen mit konstantem Zeitaufwand zu finden. Sei ein maximaler Pfad nun fixiert. Ist $\hat{v}$ nicht in seinem linken Arm enthalten, so ist $(\bar{l}, \vec{c}, \hat{r})$ auch maximal, ist $\hat{v}$ nicht im rechten Arm enthalten, so ist$(\hat{l}, \vec{c}, \bar{r})$ maximal. Da $\hat{v}$ nicht in beiden Armen des fixierten Maximums vorkommen kann, ist das Maximum von $(\hat{l}, \vec{c}, \bar{r})$ und $(\bar{l}, \vec{c}, \hat{r})$ die gesuchte maximale $\mathcal{A}_7$-Alternative (ohne Erweiterung). □

Lemma 26 *Zu gegebenem verfügbaren Zentrum $\vec{c} = (vw)$ und $i > 0$ kann, falls vorhanden, ein maximaler verfügbarer Pfad $G_i(\vec{c}) = (l, \vec{c}, r) \in L_{v,i} \times \{\vec{c}\} \times (Arms_w \cup \{\emptyset\})$ mit amortisiert[3] konstantem Zeitaufwand gefunden werden bei insgesamt einmaligem Zeitaufwand von $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$, so dass*

- *l eine verfügbare Erweiterung x_l vom Rang i besitzt und*
- *x_l zusammen mit $G_i(\vec{c})$ eine $\mathcal{A}_7$-Alternative bildet.*

Beweis. Zunächst ein paar Vorbetrachtungen. Da l eine Erweiterung vom Rang i besitzen soll, ist $l \in L_{v,i}$[4] und $r \in Arms_w \cup \{\emptyset\}$[5]. Sowohl $L_{v,i}$ als auch $Arms_w \cup \{\emptyset\}$ sind sortiert, damit ist $\text{gain}(l_1) \geq \ldots \geq \text{gain}(l_{|L_{v,i}|})$ und $\text{gain}(r_1) \geq \ldots \geq \text{gain}(r_{|Arms_w \cup \{\emptyset\}|})$, wobei l_j bzw. r_j der j-te Arm von $L_{v,i}$

[3] amortisiert über alle `GetGood`-Aufrufe

[4] Falls $L_{v,i}$ leer ist, so wird kein Pfad gefunden, da es dann keine linken Arme von $\vec{c}$ gibt, die eine Erweiterung vom Rang i besitzen.

[5] Der leere Arm kann als einsortiert betrachtet werden, da er im Folgenden nur benutzt wird, wenn er unter den vier größten rechten Armen ist, wo er mit konstantem Zeitaufwand einsortiert werden kann.

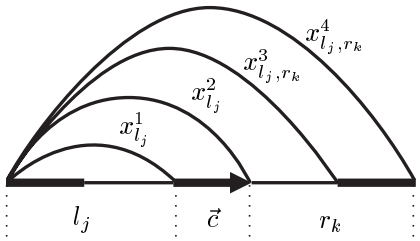

Abbildung 4.2: Sonderfälle durch Überschneidung mit der Erweiterung

bzw. $Arms_w \cup \{\emptyset\}$ ist. $P_{j,k} = (l_j, \vec{c}, r_k)$ bezeichnet dann die Kombination aus dem Zentrum mit dem entsprechenden linken und rechten Arm, wobei sich die Arme auch wie in Abbildung 3.1 (S. 14) schneiden können. Beim gesuchten $P_{j,k}$ überschneiden sich die Arme jedoch nicht. Sei also $P_{j,k}$ ein Pfad der Länge kleiner gleich fünf. Alle verfügbaren Erweiterungen des Arms l_j mit Rang i sind in $Ext_{l_j,i}$ enthalten. Da sich diese Erweiterungen wie in Abbildung 4.2 mit $P_{j,k}$ schneiden können, ergibt nicht jede Erweiterung von l_j mit $P_{j,k}$ eine $\mathcal{A}_7$-Alternative. Die Erweiterungen $x^1_{l_j}$ und $x^2_{l_j}$ enthalten das Zentrum c, hingegen $x^3_{l_j,r_k}$ und $x^4_{l_j,r_k}$ die Matchingkante des rechten Arms r_k, wobei die Erweiterung $x^4_{l_j,r_k}$ mit r_k einen C_6, also eine $\mathcal{A}_7$-Alternative, erzeugt. Ist r_k ein reduzierter Arm, so überlappen sich r_k und $x^3_{l_j,r_k}$ auf dem nicht zum Matching gehörenden Knoten, $x^4_{l_j,r_k}$ hingegen existiert in diesem Fall nicht.

Für den Pfad $P_{j,k}$ liegt eine der folgenden Situationen vor:

1. Ist $\emptyset = Ext_{l_j,i}$, so existieren für den Arm l_j keine verfügbaren Erweiterungen mehr mit Rang i. Sobald man dies feststellt, wird er aus $L_{v,i}$ entfernt. Mit Lemma 24 (S. 25) kann ein Arm in $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ solcher $L_{v,i}$-Listen auftreten, also gibt es mit Lemma 1 (S. 4) insgesamt maximal $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$ Löschungen.

2. Ist $\emptyset \neq Ext_{l_j,i} \subseteq \{x^1_{l_j}, x^2_{l_j}\}$, so führt ebenfalls keine Erweiterung mit Rang i zu einer $\mathcal{A}_7$-Alternative mit l_j, denn bei jeder der beiden Erweiterungen enthält der erweiterte Pfad $P_{j,k}$ einen C_3 oder C_4. Sobald man dies für einen Arm l_j festgestellt hat, wird er analog zum vorigen Fall aus $L_{v,i}$ entfernt mit insgesamt $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$ Löschungen.

3. Ist $x^3_{l_j,r_h} \in Ext_{l_j,i} \subseteq \{x^1_{l_j}, x^2_{l_j}, x^3_{l_j,r_h}\}$, so führt die Erweiterung $x^3_{l_j,r_h}$ von $P_{j,k}$ genau dann zu keiner $\mathcal{A}_7$-Alternative, wenn $h = k$ ist. Denn mit r_h bildet sie einen C_5, zu jedem anderen rechten Arm (inkl. leerem Arm) ist sie jedoch disjunkt oder bildet einen $C_6 \in \mathcal{A}_7$.

4. In jedem anderen Fall kann $P_{j,k}$ zu einer $\mathcal{A}_7$-Alternative erweitert werden.

Eine Löschung kann mit konstantem Zeitaufwand durchgeführt werden, wenn zum einen die Listen doppelt verkettet sind und zum anderen ein Verweis vom Arm auf den jeweiligen Listeneintrag existiert. Da jede Nicht-Matching-Kante zu höchstens zwei Armen gehört, kann zu jedem Listeneintrag ein Zeiger auf den Eintrag bei der Nicht-Matching-Kante des Arms[6] abgespeichert werden wie

[6] Ein Zeiger-Array ermöglicht den direkten Zugriff auf jeden Zeiger.

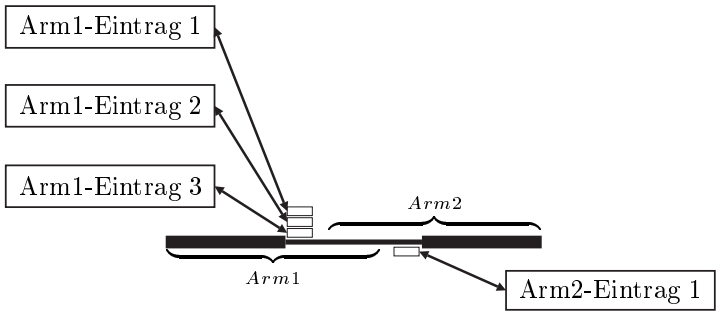

Abbildung 4.3: Verweise auf die Listeneinträge der Arme

in Abbildung 4.3 dargestellt.

Doch nun zur Suche nach dem maximalen erweiterbaren Pfad.

1. Fall: $r \notin \{r_1, r_2, r_3\}$

Um diesen zu finden, wird zunächst der größte verfügbare linke Arm betrachtet, der eine Erweiterung vom Rang i besitzt. Das ist $l_1 \in L_{v,i}$. Solange für l_1 die Situation 1 oder 2 vorliegt, wird l_1 entsprechend gelöscht und ein neues l_1 ermittelt. Wie bei diesen Situationen festgestellt, gibt es maximal $\mathcal{O}(m \log \frac{n}{\alpha})$ Löschungen. Zudem sei $\bar{r} \in Arms_w \cup \{\emptyset\}$ der rechte verfügbare Arm mit dem kleinsten Index (größten Gewinn), der mit l_1 und einer Erweiterung $x_{l_1} \in Ext_{l_1,i}$ eine $\mathcal{A}_7$-Alternative bildet. Da sich l_1 nur mit zwei (nicht leeren) rechten Armen schneiden kann (s. Abbildung 3.1, S. 14) und nur mit exakt einem (nicht leeren) rechten Arm die Situation 3 eintreten kann, ist $\bar{r}$ in $\{r_1, r_2, r_3, r_4\}$[7] zu finden.

Sei ein maximaler Pfad fixiert. Ist sein rechter Arm r nicht in $\{r_1, r_2, r_3\}$, so ist $(l_1, \vec{c}, \bar{r})$ in diesem Fall ebenfalls maximal. Als Erweiterung kann ein beliebiges $x_{l_1} \in Ext_{l_1,i} \setminus \{x^1_{l_j}, x^2_{l_j}, x^3_{l_j,\bar{r}}\}$ gewählt werden.

Da zu einem gegebenem $P_{j,k}$ mit konstantem Zeitaufwand festgestellt werden kann, ob sich l_j und r_k schneiden oder ob $Ext_{l_j,i} \subseteq \{x^1_{l_j}, x^2_{l_j}, x^3_{l_j,r_k}\}$ ist, kann $\bar{r}$ deshalb mit konstantem Zeitaufwand in der Menge $\{r_1, r_2, r_3, r_4\}$ (unter Beachtung des leeren Arms) gefunden werden.

2. Fall: $l \notin L$

Nun werden die restlichen linken Arme betrachtet. Dabei sei $\bar{l} \in L_{v,i}$ der linke verfügbare Arm mit dem kleinsten Index (größten Gewinn), der mit r_1 und einer Erweiterung $x_{\bar{l}}$ vom Rang i eine $\mathcal{A}_7$-Alternative bildet. Sei L die Menge der linken Arme aus $L_{v,i}$ mit kleinerem Index als $\bar{l}$. Falls $\bar{l}$ nicht existiert, sei $L = L_{v,i}$. Jeder Arm aus L schneidet sich entweder direkt mit r_1 (davon gibt es maximal zwei Arme) oder aber es tritt Situation 1, 2 oder 3 für ihn ein, d. h. alle seine Erweiterungen vom Rang i schneiden sich mit $\vec{c}$ oder r_1.

Im Falle, dass der linke Arm l des fixierten Maximums nicht in L enthalten ist, folgt $L \neq L_{v,i}$. Also existiert $\bar{l}$ und $(\bar{l}, \vec{c}, r_1)$ entspricht in diesem Fall der gesuchten maximalen Alternative zusammen mit einem beliebigen $x_{\bar{l}} \in Ext_{\bar{l},i} \setminus \{x^1_{\bar{l}}, x^2_{\bar{l}}, x^3_{\bar{l},r_1}\}$.

[7] Der leere Arm ist in dieser Menge enthalten, falls r_5 einen negativen Gewinn hat oder falls es gar keine vier rechten verfügbaren Arme gibt.

Um $\bar{l}$ zu finden, muss die Menge L durchlaufen werden. Merkt man sich jedoch das Ende von L, so kann man bei der nächsten Suche nach $\bar{l}$ am Ende von L beginnen und muss L kein zweites Mal durchlaufen, solange r_1 verfügbar und damit in $Arms_w$ enthalten ist. Ist $\vec{c}$ noch verfügbar, der Arm r_1 aus $Arms_w$ jedoch nicht mehr verfügbar, so wurde er bei der Aktualisierung aus $Arms_w$ entfernt. Damit sind auch die (maximal zwei) linken Arme, die sich mit r_1 direkt schneiden, nicht mehr verfügbar und wurden bei der Aktualisierung aus $Arms_v$ gelöscht. Für alle anderen Arme l_j aus L ist wie oben festgestellt $Ext_{l_j,i} \subseteq \{x^1_{l_j}, x^2_{l_j}, x^3_{l_j,r_1}\}$, aber die Erweiterung $x^3_{l_j,r_1}$ ist wie r_1 nun auch nicht mehr verfügbar (d.h. aus $Ext_{l_j,i}$ entfernt) und damit tritt für sie die Situation 1 oder 2 ein und sie werden aus $L_{v,i}$ bei der nächsten Anfrage entfernt (auf der Suche nach l_1, mit einem Gesamtaufwand von $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$). Auf der Suche nach $\bar{l}$ wird jeder Arm in $L_{v,i}$ also maximal einmal negativ getestet, mit Lemma 1 (S. 4) und Korollar 24 (S. 25) benötigt dies einen Gesamtaufwand von $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$. $\bar{l}$ wird also mit amortisiert konstantem Zeitaufwand gefunden.

3. Fall: $l \in L$ und $r \in \{r_2, r_3\}$

Laut Definition bildet kein Arm in L mit r_1 und $\vec{c}$ einen Pfad, der zu einer verfügbaren $\mathcal{A}_7$-Alternative mit einer Erweiterung vom Rang i linksseitig erweitert werden könnte. Es verbleibt somit für die Arme des fixierten Maximums lediglich der Fall $l \in L$ und $r \in \{r_2, r_3\}$.

Zu $r \in \{r_2, r_3\}$ bezeichne deshalb $\bar{l}_r$, falls vorhanden, den größten linken Arm aus L, der zusammen mit r und einer Erweiterung $x_{\bar{l}_r}$ eine $\mathcal{A}_7$-Alternative bildet. Das Maximum von $(\bar{l}_{r_2}, \vec{c}, r_2)$ und $(\bar{l}_{r_3}, \vec{c}, r_3)$ ist in diesem Fall also die gesuchte Alternative mit entsprechend wie oben zu wählender Erweiterung.

$\bar{l}_{r_2}$ (bzw. analog $\bar{l}_{r_3}$) ist einfach zu finden. Wie oben festgestellt, überlappt sich jeder Arm l_j in L entweder mit r_1 direkt oder aber seine Erweiterungen $Ext_{l_j,i}$ ist Teilmenge von $\{x^1_{l_j}, x^2_{l_j}, x^3_{l_j,r_1}\}$. Ignoriert man zunächst alle Arme aus L, für welche die Situation 1 oder 2 eintritt, so gibt es maximal zwei Arme in L, die sich mit r_1 schneiden, alle anderen besitzen die Erweiterung $x^3_{l_j,r_1}$. Weiterhin gibt es höchstens zwei linke Arme, die sich direkt mit r_2 überlappen. Damit gibt es unter den ersten fünf Armen von L mindestens einen Arm l_j, der sich weder mit r_1 noch mit r_2 schneidet und zudem die Erweiterung $x^3_{l_j,r_1}$ besitzt, die mit r_2 einen C_6 bildet (s. Situation 3) oder disjunkt zu r_2 ist (s. Situation 4). Um $\bar{l}_{r_2}$ zu finden läuft man also L ab. Dabei löscht man Arme aus L, für welche Situation 1 oder 2 eintritt mit einem Gesamtaufwand von $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$, und findet $\bar{l}_{r_2}$ unter den restlichen ersten fünf. Der Arm $\bar{l}_{r_2}$ (bzw. analog $\bar{l}_{r_3}$) kann also mit amortisiert konstantem Zeitaufwand pro Anfrage bestimmt werden.

Da für das fixierte Maximum einer der Fälle eintreten muss, entspricht das Maximum aus $(l_1, \vec{c}, \bar{r}), (\bar{l}, \vec{c}, r_1), (\bar{l}_{r_2}, \vec{c}, r_2)$ und $(\bar{l}_{r_3}, \vec{c}, r_3)$ der gesuchten maximalen verfügbaren $\mathcal{A}_7$-Alternative und kann mit amortisiert konstantem Zeitaufwand gefunden werden. Da die jeweilige Erweiterung x_l von $G_i(\vec{c}) = (l, \vec{c}, r)$ in $Ext_{l,i} \setminus \{x^1_l, x^2_l, x^3_{l,r}\}$ beliebig gewählt werden kann und $Ext_{l,i}$ nach Gewinn sortiert ist, kann $G_i(\vec{c})$ auch mit der kleinstmöglichen, sogenannten **provisorischen Erweiterung** zu $\boldsymbol{G^+_i(\vec{c})}$ erweitert werden. Die Erweiterung ist „provisorisch“, da sie später bei eventueller Nicht-Verfügbarkeit durch eine andere, größere ersetzt wird. □

Lemma 27 `GetGood` *berechnet, falls vorhanden, mit amortisiert konstantem Zeitaufwand und einmaligem Zeitaufwand von* $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$ *eine verfügbare* $\mathcal{A}_7$*-Alternative* $G_i^+(\vec{c})$*, so dass für alle verfügbaren* $\mathcal{A}_7$*-Alternativen* A *mit Zentrum* $\vec{c}$ *und Erweiterung vom Rang* i *gilt:*

$$\text{gain}_7^{\vec{c}}(G_i^+(\vec{c})) \geq \frac{\text{gain}_7^{\vec{c}}(A) - \beta\,\omega(M \cap A) - \frac{\alpha}{n}\,\omega(M^*)}{1+\beta}.$$

Beweis. Sei A eine verfügbare $\mathcal{A}_7$-Alternative mit Zentrum c. Mit Lemma 2 (S. 5) kann A zerlegt werden in $(x_l, l, \vec{c}, r)$, d. h. in das Zentrum $\vec{c}$ mit einem linken und rechten Arm, sowie einer Erweiterung des linken Arms.

1. Fall: $r(x_l) = 0$

Dann folgt mit Korollar 11 (S. 11):

$$0 \geq \frac{\text{gain}(x_l) - \frac{1}{n}\,\alpha\, w_{max}}{1+\beta}.$$

Da $G_0^+(\vec{c}) = G_0(\vec{c})$ folgt zusammen mit Korollar 3 (S. 6) und Lemma 25 (S. 26):

$$\text{gain}_7^{\vec{c}}(G_0^+(\vec{c})) = \text{gain}(G_0(\vec{c})) \geq \text{gain}(l, \vec{c}, r) = \text{gain}(l) + \text{gain}(r) - \omega(\vec{c}).$$

Beide Ungleichungen zusammen ergeben:

$$\begin{aligned}
\text{gain}_7^{\vec{c}}(G_0^+(\vec{c})) &\geq \frac{\text{gain}(x_l) - \frac{1}{n}\,\alpha\, w_{max}}{1+\beta} + \text{gain}(l) + \text{gain}(r) - \omega(\vec{c}) \\
&= \frac{\text{gain}_7^{\vec{c}}(A) + \beta\,(\text{gain}(l) + \text{gain}(r) - \omega(\vec{c})) - \frac{1}{n}\,\alpha\, w_{max}}{1+\beta} \\
&\geq \frac{\text{gain}_7^{\vec{c}}(A) - \beta\,\omega(M \cap A) - \frac{1}{n}\,\alpha\,\omega(M^*)}{1+\beta}, \quad \text{da } w_{max} \leq \omega(M^*).
\end{aligned}$$

2. Fall: $r(x_l) = i > 0$

Dann besitzt $G_i^+(\vec{c})$ die provisorische Erweiterung x_G vom gleichen Rang $i > 0$ und mit Lemma 10 (S. 11) gilt $\text{gain}(x_G) > 0$. Dann folgt mit Korollar 11 (S. 11):

$$\text{gain}(x_G) \geq \frac{\text{gain}(x_l) - \frac{1}{n}\,\alpha\, w_{max}}{1+\beta}.$$

Nun ist nach Definition von Gewinn$_7$ und Lemma 26 (S. 26):

$$\text{gain}_7^{\vec{c}}(G_i^+(\vec{c})) = \text{gain}(x_G) + \text{gain}(G_i(\vec{c})) \geq \text{gain}(x_G) + \text{gain}(l, \vec{c}, r).$$

Beide Ungleichungen zusammen ergeben

$$\text{gain}_7^{\vec{c}}(G_i^+(\vec{c})) \geq \frac{\text{gain}(x_l) - \frac{1}{n}\,\alpha\, w_{max}}{1+\beta} + \text{gain}(l) + \text{gain}(r) - \omega(\vec{c}).$$

Der Rest der Rechnung verläuft wie im ersten Fall.

Die Laufzeit ergibt sich aus Lemma 25 (S. 26) und Lemma 26 (S. 26). □

Es wird nun zu beiden Orientierungen einer jeden Matchingkante c und jedem möglichen Rang i mit `GetGood` die $\mathcal{A}_7$-Alternative $G_i^+(\vec{c})$ und deren Gewinn$_7$ berechnet. Dies sind maximal $\mathcal{O}(n \log_{1+\beta} \frac{n}{\alpha})$ Alternativen, da es höchstens $n/2$ viele Matchingkanten und $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ viele Ränge gibt. Sie werden nun ebenfalls nach Gewinn$_7$ partioniert mit dem gleichen α, β und n wie die Erweiterungen zuvor. Weiterhin wird $f(a) := \text{gain}_7^{\vec{c}_a}(a)$ benutzt und damit $f_{max} \leq 3\,\omega_{max}$. Dabei wird jede Partitionsklasse als Liste $\boldsymbol{\mathcal{P}_i}$ angelegt. Da die Elemente aus Partitionsklasse $\mathcal{P}_0$ nicht mehr vonnöten sind, werden sie gelöscht, d.h. $\mathcal{P}_0 = \emptyset$. Mit Korollar 12 (S. 11) und Lemma 27 folgt ein Gesamtaufwand von $\mathcal{O}((n+m)\log_{1+\beta} \frac{n}{\alpha})$.

4.2 Der Algorithmus

Greedy-$\mathcal{A}_7$ ($G = (V, E)$, *maximales* M, $\omega : E \mapsto \mathbb{R}_+$)

```
(1)  Wiederhole
(2)      P_max := nicht leere Klasse P_i mit max. Rang
(3)      P := erste Alternative aus P_max; die provisorische
             Erweiterung wird durch eine verfügbare
             Erweiterung gleichen Ranges ersetzt
(4)      C := erster verfügbarer Kreis aus C
(5)      Resultat := Resultat ⊎ Max. von C und P bzgl. Gewinn
(6)      Für jeden nicht mehr verfügbaren Knoten u:
(7)          ∀v, i : Entferne l aus L_{v,i} falls u ∈ l
(8)          ∀v : Entferne l aus Arms_v falls u ∈ l
(9)          ∀v : Entferne x aus Ext_{v,r(x)} falls u ∈ x
(10)     Für jeden nicht mehr verfügbaren oder erweiterbaren
         Pfad G_i(c)^8:
(11)         Entferne G_i^+(c) aus P_{r(G_i^+(c))} und berechne neuen
             verfügbaren G_i^+(c), falls vorhanden, und füge
             G_i^+(c) in P_{r(G_i^+(c))} ein, falls r(G_i^+(c)) > 0
(12) Bis C = P = ∅
(13) M := MaxMatch(M \ Resultat ⊎ Resultat \ M)
```

Der Algorithmus sammelt knotendisjunkte Alternativen, die allesamt positiven Gewinn haben, da sowohl in $\mathcal{P}$ keine Elemente der Partitionsklasse Null und in $\mathcal{C}$ nur positive Kreise enthalten sind. Nachdem in Zeile 5 eine Alternative ausgewählt wurde, sind ihre Knoten nicht mehr **verfügbar**. Entsprechend sind auch alle anderen Alternativen, die solch einen Knoten beinhalten, nicht mehr verfügbar. Anfangs sind alle provisorischen Erweiterungen verfügbar, später wird in Zeile 3 der Pfad, falls die provisorische Erweiterung nicht mehr verfügbar ist, durch eine andere erweitert. Der Pfad $G_i(\vec{c})$ ist nicht mehr **erweiterbar**, wenn $i > 0$ und keine Erweiterung des linken Arms vom Rang i mit $G_i(\vec{c})$ eine verfügbare $\mathcal{A}_7$-Alternative bildet.[9]

Lemma 28 *Der Algorithmus wählt nur knotendisjunkte Alternativen aus.*

[8] Verfügbarkeit wird in Zeile 10 nur für $G_i(\vec{c})$ getestet, nicht jedoch für die provisorische Erweiterung (s. dazu Zeile 3).

[9] D. h. es tritt Situation 1, 2 oder 3 aus dem Beweis von Lemma 26 (S. 26) ein.

Beweis. Das gewählte C ist mit Zeile 4 stets verfügbar. Die Alternative $G_i^+(\vec{c}) \in \mathcal{P}_{r(G_i^+(\vec{c}))}$ besteht aus $G_i(\vec{c})$ und einer provisorischen Erweiterung. $G_i^+(\vec{c})$ ist zu Beginn verfügbar. Ist $G_i(\vec{c})$ nicht mehr verfügbar, so wird die Alternative in Zeile 10 und 11 durch eine verfügbare Alternative ersetzt, ist hingegen die provisorische Erweiterung nicht mehr verfügbar, so bleibt sie bestehen. Wird sie in Zeile 3 ausgewählt, so wird die nicht mehr verfügbare provisorische Erweiterung durch eine verfügbare gleichen Ranges ersetzt. Die Existenz einer solchen Erweiterung wird mit Zeile 10-11, also der Erweiterbarkeit, garantiert. Damit ist auch P stets verfügbar, d. h. knotendisjunkt zu allen zuvor gewählten Alternativen. □

4.2.1 Die Laufzeit

Völlig analog zum Lemma 16 und 17 (S. 16) folgen Lemma 29 und 30:

Lemma 29 *Es gibt höchstens $\mathcal{O}(n)$ Schleifendurchläufe.* □

Lemma 30 *Das Berechnen von C benötigt insgesamt $\mathcal{O}(m)$ Laufzeit. C ist stets der größte verfügbare C_4.* □

Lemma 31 *Das Berechnen von P benötigt insgesamt $\mathcal{O}(n \cdot \log_{1+\beta} \frac{n}{\alpha})$ Laufzeit. P entstammt der nicht leeren Partitionsklasse mit größtmöglichem Rang.*

Beweis. $\mathcal{P}_{max}$ kann pro Schleifendurchlauf mit Lemma 12 (S. 11) in $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ gefunden werden, mit Lemma 29 folgt ein Gesamtaufwand von $\mathcal{O}(n \log_{1+\beta} \frac{n}{\alpha})$. Die Alternative P entstammt $\mathcal{P}_{max} = \mathcal{P}_{r(P)}$ und hat somit größtmöglichen Rang. Ist ihre provisorische Erweiterung nicht mehr verfügbar, so wird diese durch eine andere Erweiterung gleichen Ranges ersetzt. Diese kann, wie im Beweis von Lemma 26 (S. 26) festgestellt, beliebig aus $Ext_{l,i} \setminus \{x_l^1, x_l^2, x_{l,r}^3\}$ mit konstantem Zeitaufwand gewählt werden. Die Existenz einer solchen Erweiterung ist mit Zeile 10-11 gegeben. Da die provisorische Erweiterung die kleinstmögliche Erweiterung war, vergrößert die neue Erweiterung den Gewinn$_7$ bzw. den Rang mitunter noch. □

Lemma 32 *Die Aktualisierung benötigt insgesamt $\mathcal{O}((n+m) \log_{1+\beta} \frac{n}{\alpha})$ Laufzeit.*

Beweis. Analog zum Beweis von Lemma 19 (S. 17) können alle Arme, die einen gemeinsamen Knoten mit der ausgewählten Alternative besitzen, mit einer Laufzeit von insgesamt $\mathcal{O}(m)$ enumeriert werden. Ein jeder Arm kommt mit Lemma 22, 23 und 24 (S. 24) in genau einer $Arms_v$-Liste, in genau einer $Ext_{v,i}$-Liste und höchstens $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ vielen $L_{v,i}$-Listen vor. Unter Benutzung der Datenstrukturen in Abbildung 4.3 (S. 28) können die Listeneinträge mit konstantem Zeitaufwand aus diesen (doppelt verketteten) Listen gelöscht werden, soweit die Arme in den Listen noch enthalten sind. Daraus folgt ein Gesamtaufwand von $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$ Laufzeit für die Aktualisierung in den Zeilen 6-9 und die Listen enthalten anschließend nur verfügbare Arme.

Eine Alternative $G_i(\vec{c})$ ist nicht mehr verfügbar, wenn ihr Zentrum oder einer ihrer Arme nicht mehr verfügbar ist. Ein Zentrum $\vec{c}$ wird höchstens einmal direkt

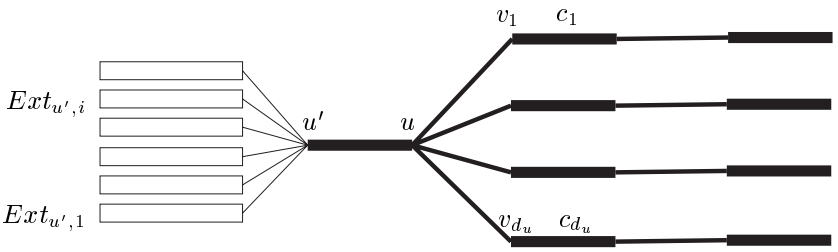

Abbildung 4.4: Aktualisierung bei Nichterweiterbarkeit

ausgewählt. In diesem Fall müssen alle $G_i^+(\vec{c})$-Einträge entfernt werden. Es gibt maximal $n/2$ Zentren mit je zwei Orientierungen und pro gerichtetem Zentrum gibt es höchstens $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ viele Einträge in $\mathcal{P}$, es folgt ein Gesamtaufwand von $\mathcal{O}(n \log_{1+\beta} \frac{n}{\alpha})$. Ebenso ist jeder der $2m$ Arme höchstens einmal betroffen, und mit jedem Arm höchstens ein Zentrum mit höchstens $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha})$ $G_i^+(\vec{c})$-Einträgen zu beiden Orientierungen. Da mit Lemma 27 (S. 30) ein neues $G_i^+(\vec{c})$ mit amortisiert konstantem Zeitaufwand berechenbar ist, kann jedes $G_i^+(\vec{c})$ mit nicht mehr verfügbarer Alternative $G_i(\vec{c})$ mit konstantem Zeitaufwand erneuert werden, wenn beim Zentrum Zeiger auf die Partitionseinträge existieren und die Partitionsklassen selbst doppelt verkettet sind. Es folgt ein Gesamtaufwand von $\mathcal{O}((m+n) \cdot \log_{1+\beta} \frac{n}{\alpha})$.

Ein $G_i(\vec{c}) = (l, \vec{c}, r)$ ist nicht mehr erweiterbar, wenn $i > 0$ und keine Erweiterung vom Rang i mit $G_i(\vec{c})$ eine verfügbare $\mathcal{A}_7$-Alternative bildet. Dies kann also nur eintreten, wenn eine Erweiterung aus $Ext_{l,i}$ gelöscht wird. Wie im Beweis von Lemma 26 (S. 26) festgestellt, wird es auch nur in den Situationen 1, 2 und 3 kritisch, also wenn $|Ext_{l,i}| \leq 3$ ist. Pro Liste $Ext_{u',i}$ von u' müssen also höchstens viermal alle betroffenen $G_i^+(\vec{c})$ erneuert werden, wobei die Alternative $G_i^+(\vec{c})$ genau dann betroffen ist, wenn ihr linker Arm gerade die Matchingkante uu' enthält und in u' endet. Hat also u' den Matchingnachbar u und dieser wiederum wie in Abbildung 4.4 gerade d_u Nachbarn v, die jeweils in einem Zentrum c_v enthalten sind, so wird jedes $G_i^+(\vec{c}_v)$ getestet, ob der linke Arm u' benutzt, und dann erneuert. Mit Lemma 23 (S. 25) ergibt dies pro Knoten u' gerade $\mathcal{O}(4 \cdot \log_{1+\beta} \frac{n}{\alpha} \cdot d_u)$ Tests, insgesamt also $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$, eine Erneuerung kann mit `GetGood` jeweils nach Lemma 27 (S. 30) mit amortisiert konstantem Zeitaufwand geschehen. □

Satz 33 *Der* `Greedy-`$\mathcal{A}_7$ *hat eine Laufzeit von* $\mathcal{O}((n+m) \cdot \log_{1+\beta} \frac{n}{\alpha})$.

Beweis. Die Initialisierung benötigt einen Aufwand von $\mathcal{O}((m+n) \cdot \log_{1+\beta} \frac{n}{\alpha})$. Der Algorithmus selbst benötigt mit Lemma 29, 30, 31 und 32 insgesamt $\mathcal{O}((m+n) \cdot \log_{1+\beta} \frac{n}{\alpha})$. □

Der $\log_{1+\beta} \frac{n}{\alpha}$-Faktor könnte an einigen Stellen durch eine aufwändigere Analyse vermieden werden, indem z.B. die C_4-Kreise partitioniert anstatt sortiert werden. Andererseits konnte der Faktor in dieser Arbeit nicht vermieden werden. So gibt es $\mathcal{O}(n \log_{1+\beta} \frac{n}{\alpha})$ Listen $Ext_{v,i}$ und $L_{v,i}$. Letztere werden in $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$ erstellt. Ebenso gibt es $\mathcal{O}(n \log_{1+\beta} \frac{n}{\alpha})$ partitionierte Alternativen $G_i^+(\vec{c})$. `GetGood`

benötigt einen Gesamtaufwand von $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$ zum Berechnen dieser Alternativen. Und schließlich wird für die Aktualisierung der Datenstrukturen, also die Sicherstellung der Verfügbarkeit und Erweiterbarkeit, ein Zeitaufwand von $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$ benötigt.

4.2.2 Die Korrektheit

Satz 34 *Der vom Algorithmus erzielte Gewinn ist*

$$\sum_{A_{alg} \in Resultat} \text{gain}(A_{alg}) \geq \frac{1}{4\,(1+\beta)^2} \left(\left(\frac{3}{4} - \frac{8+6\,\beta}{4} \alpha \right) \omega(M^*) - (1+\beta)\,\omega(M) \right)$$

Beweis. Der Beweis verläuft ähnlich zum Beweis von Satz 21 (S. 18). Auch hier wird die disjunkte Zerlegung von $\mathcal{A}^*$ in $\mathcal{A}^*_{C_4}$ und $\mathcal{A}^*_\infty$ betrachtet und jede Alternative $A \in \mathcal{A}^*$ durch die erste sie überlappende, im Resultat enthaltene Alternative $\text{Cut}(A)$ abgeschätzt. Wird A nicht überlappt, so ist $\text{Cut}(A) = \emptyset$. Da im Resultat nur positive Alternativen vorkommen, ist $\text{gain}(\text{Cut}(A)) \geq 0$.

1. Fall: $\mathcal{A}^*_{C_4}$

Sei also C ein alternativer C_4-Kreis aus $\mathcal{A}^*_{C_4}$. Ist sein Gewicht kleiner gleich Null, so folgt:

$$\text{gain}(\text{Cut}(C)) \geq 0 \geq \text{gain}(C)/(1+\beta)^2.$$

Ist hingegen sein Gewicht größer Null, so ist er im Rahmen der Initialisierung in der nach Gewicht sortierten Liste $\mathcal{C}$ eingefügt worden. Solange C verfügbar und damit in $\mathcal{C}$ enthalten ist, ist das Gewicht des jeweils ersten verfügbaren Kreises $C_1 \in \mathcal{C}$ damit größer gleich dem von C. Weiterhin ist die vom Algorithmus ausgewählte Alternative wegen Zeile 5 größer als bzw. gleich dem momentan ersten Kreis und damit:

$$\text{gain}(\text{Cut}(C)) \geq \text{gain}(C_1) \geq \text{gain}(C) \geq \text{gain}(C)/(1+\beta)^2.$$

Beide Ungleichungen ergeben zusammen:

$$\sum_{C \in \mathcal{A}^*_{C_4}} \text{gain}(\text{Cut}(C)) \geq \sum_{C \in \mathcal{A}^*_{C_4}} \frac{\text{gain}(C)}{(1+\beta)^2}$$

Nun ist $\sum_{C \in \mathcal{A}^*_{C_4}} \text{gain}(C) = \omega(M^* \cap \mathcal{A}^*_{C_4}) - \omega(M \cap \mathcal{A}^*_{C_4})$ und damit (zur späteren Verwendung etwas abgeschwächt):

$$3 \cdot \sum_{C \in \mathcal{A}^*_{C_4}} \text{gain}(\text{Cut}(C)) \geq \frac{3 \cdot \omega(M^* \cap \mathcal{A}^*_{C_4}) - 4 \cdot \omega(M \cap \mathcal{A}^*_{C_4})}{(1+\beta)^2} \tag{4.1}$$

2. Fall: $\mathcal{A}^*_\infty$

Betrachtet werden nun die Alternativen in $\mathcal{A}^*_\infty$. Mit Lemma 7 (S. 8) gibt es eine Überdeckung Ω_7 von $\mathcal{A}^*_\infty$ durch $\mathcal{A}_7$-Alternativen, so dass

$$\sum_{A \in \Omega_7} \text{gain}_7^{\vec{c}_A}(A) \geq 3 \cdot \omega(M^* \cap \mathcal{A}^*_\infty) - 4 \cdot \omega(M \cap \mathcal{A}^*_\infty). \tag{4.2}$$

Sei A solch eine Alternative mit Zentrum $\vec{c}_A$ und einer Erweiterung vom Rang i des linken Arms. Zu jeder (orientierten) Matchingkante $\vec{c}$ und jedem Rang i wurde im Rahmen der Initialisierung mittels `GetGood` eine (mitunter leere) Alternative $G_i^+(\vec{c})$ mit Rang $r(G_i^+(\vec{c}))$ berechnet, die eventuell im Laufe des Algorithmus mehrmals mittels `GetGood` aktualisiert wurde. Solange A verfügbar ist, gilt für jedes so berechnete $G_i^+(\vec{c}_A)$ mit Lemma 27 (S. 30):

$$\mathrm{gain}_7^{\vec{c}_A}(G_i^+(\vec{c}_A)) \geq \frac{\mathrm{gain}_7^{\vec{c}_A}(A) - \beta\,\omega(M \cap A) - \frac{\alpha}{n}\,\omega(M^*)}{1+\beta}.$$

Mit Korollar 8 (S. 10) gilt $\|\Omega_7\| \leq 2\,n$. Weiterhin gilt mit Korollar 9 (S. 10), dass jede Matchingkante höchstens viermal in der Überdeckung vorkommt, also $\sum_{A\in\Omega_7} \omega(M \cap A) \leq 4\,\omega(M)$. Solange sich jedes $G_i^+(\vec{c}_A)$ auf ein verfügbares A bezieht, folgt damit:

$$\sum_{A\in\Omega_7} \mathrm{gain}_7^{\vec{c}_A}(G_i^+(\vec{c}_A)) \geq \frac{\left(\sum_{A\in\Omega_7} \mathrm{gain}_7^{\vec{c}_A}(A)\right) - 4\,\beta\,\omega(M) - 2\,\alpha\,\omega(M^*)}{1+\beta}. \quad (4.3)$$

Ist $r(G_i^+(\vec{c}_A))$ gleich Null, so folgt mit Korollar 11.i (S. 11):

$$0 \geq \frac{\mathrm{gain}_7^{\vec{c}_A}(G_i^+(\vec{c}_A)) - \frac{\alpha}{n}\,3\,w_{max}}{1+\beta}. \quad (4.4)$$

Ist $r(G_i^+(\vec{c}_A))$ hingegen größer als Null, so ist die Partitionsklasse $\mathcal{P}_{r(G_i^+(\vec{c}_A))}$ nicht leer. Die in Zeile 3 jeweils ermittelte erste Alternative P hat dann mit Lemma 31 (S. 32) einen Rang größer gleich $r(G_i^+(\vec{c}_A))$. Durch Austausch ihrer provisorischen (also kleinstmöglichen) Erweiterung wird ihr Rang höchstens noch vergrößert. Mit Korollar 11.i (S. 11) :

$$\mathrm{gain}_7^{\vec{c}_P}(P) \geq \frac{\mathrm{gain}_7^{\vec{c}_A}(G_i^+(\vec{c}_A)) - \frac{\alpha}{n}\,3\,w_{max}}{1+\beta}. \quad (4.5)$$

Wird die Alternative $\mathrm{Cut}(A)$ in Zeile 5 ausgewählt, so ist mit Zeile 3 und 5 ihr Gewinn größer gleich dem Gewinn der im Moment ersten Alternative P, also folgt mit Korollar 3 (S. 6):

$$\mathrm{gain}(\mathrm{Cut}(A)) \geq \mathrm{gain}(P) \geq \mathrm{gain}_7^{\vec{c}_P}(P).$$

Zusammen mit Ungleichung 4.4 und 4.5 ergibt sich:

$$\begin{aligned}\sum_{A\in\Omega_7} \mathrm{gain}(\mathrm{Cut}(A)) &\geq \sum_{A\in\Omega_7} \frac{\mathrm{gain}_7^{\vec{c}_A}(G_i^+(\vec{c}_A)) - \frac{\alpha}{n}\,3\,w_{max}}{(1+\beta)} \\ &\geq \frac{\left(\sum_{A\in\Omega_7} \mathrm{gain}_7^{\vec{c}_A}(G_i^+(\vec{c}_A))\right) - 6\,\alpha\,\omega(M^*)}{(1+\beta)}\end{aligned}$$

da $w_{max} \leq \omega(M^*)$ und $\|\Omega_7\| \leq 2\,n$ mit Korollar 8 (S. 10). Setzt man hier nun Ungleichung 4.3 ein, so folgt:

$$\sum_{A\in\Omega_7} \mathrm{gain}(\mathrm{Cut}(A)) \geq \frac{\left(\sum_{A\in\Omega_7} \mathrm{gain}_7^{\vec{c}_A}(A)\right) - 4\,\beta\,\omega(M) - (8+6\,\beta)\,\alpha\,\omega(M^*)}{(1+\beta)^2}.$$

und mit Ungleichung 4.2 gilt dann:

$$\sum_{A\in\Omega_7} \text{gain}(\text{Cut}(A)) \geq \frac{3\cdot\omega(M^*\cap\mathcal{A}^*_\infty) - 4\cdot\omega(M\cap\mathcal{A}^*_\infty) - (8+6\,\beta)\,\alpha\,\omega(M^*) - 4\,\beta\,\omega(M)}{(1+\beta)^2}.$$

Es werden nun beide Fälle zusammengeführt. Zusammen mit der im ersten Fall hergeleiteten Ungleichung 4.1 für Kreise folgt:

$$\sum_{A\in\Omega_7} \text{gain}(\text{Cut}(A)) + \sum_{C\in\mathcal{A}^*_{C_4}} 3\cdot\text{gain}(\text{Cut}(C)) \geq \frac{3\cdot\omega(M^*\cap A^*) - 4\cdot\omega(M\cap A^*) - (8+6\,\beta)\,\alpha\,\omega(M^*) - 4\,\beta\,\omega(M)}{(1+\beta)^2}$$

Mit $\mathcal{A}^* = (M^*\cup M)\setminus(M^*\cap M)$ folgt $\omega(M^*\cap\mathcal{A}^*) = \omega(M^*\setminus M^*\cap M) = \omega(M^*) - \omega(M^*\cap M)$. Analog ist $\omega(M\cap\mathcal{A}^*) = \omega(M) - \omega(M^*\cap M)$ und damit

$$3\cdot\omega(M^*\cap\mathcal{A}^*) - 4\cdot\omega(M\cap\mathcal{A}^*) \geq 3\cdot\omega(M^*) - 4\cdot\omega(M) + \omega(M^*\cap M).$$

Enthält eine Alternative einen Knoten u, so auch die zugehörige Matchingkante m_u, falls diese existiert. Eine im Resultat enthaltene Alternative besitzt zusammengenommen höchstens vier nicht zum Matching gehörende Knoten oder aber Matchingkanten. Mit jedem dieser Knoten oder Matchingkanten kann sie entweder genau einen Kreis in $\mathcal{A}^*_{C_4}$ oder, mit Lemma 7 (S. 8) und Korollar 9 (S. 10), höchstens vier Alternativen aus Ω_7 überlappen. Damit kann jede im Resultat enthaltene Alternative höchstens 16-mal auf der linken Seite der obigen Ungleichung als $\text{Cut}(A)$ bzw. $\text{Cut}(C)$ auftauchen. Da jedes $\text{Cut}(A)$ entweder leer oder im Resultat enthalten ist und alle Alternativen des Resultats positiven Gewinn haben, folgt:

$$16\cdot\sum_{A\in Resultat}\text{gain}(A) \geq \frac{3\,\omega(M^*) - 4\,\omega(M) - (8+6\,\beta)\,\alpha\,\omega(M^*) - 4\,\beta\,\omega(M)}{(1+\beta)^2}.$$

□

4.3 Die Approximation von $(3/4-\epsilon)\,\omega(M^*)$

Dazu wird der `Greedy-`$\mathcal{A}_7$ wiederholt ausgeführt. Pro Aufruf erzielt er mit Satz 34 (S. 34) einen Gewinn von

$$\sum_{A_{alg}\in Resultat}\text{gain}(A_{alg}) \geq \frac{1}{4\,(1+\beta)^2}\left(\left(\frac{3}{4} - \frac{8+6\,\beta}{4}\alpha\right)\omega(M^*) - (1+\beta)\,\omega(M)\right)$$

Dann gilt analog zum Abschnitt 3.3 (S. 21) mit $\rho = \frac{1}{4\,(1+\beta)}$ und $\mu = \frac{\rho}{1+\beta}\left(\frac{3}{4} - \frac{8+6\,\beta}{4}\,\alpha\right)\omega(M^*)$:

$$\begin{aligned}\omega(M_k) &\geq \frac{1-(1-\rho)^k}{1-(1-\rho)}\cdot\mu = \frac{1-(1-\rho)^k}{\rho}\cdot\frac{\rho}{1+\beta}\left(\frac{3}{4}-\frac{8+6\,\beta}{4}\,\alpha\right)\omega(M^*)\\ \omega(M_k) &\geq \frac{1-\left(1-\frac{1}{4\,(1+\beta)}\right)^k}{1+\beta}\left(\frac{3}{4}-\frac{8+6\,\beta}{4}\,\alpha\right)\omega(M^*)\end{aligned}$$

Mit $\epsilon < 3/4$ werden nun $0 < \alpha, \beta \leq \epsilon/11 < 3/44$ fixiert. Weiterhin sei $k > \log_{37/26} 1/\epsilon$, also $\epsilon \geq \left(\frac{26}{37}\right)^k = \left(1-\frac{1}{4\,(1+3/44)}\right)^k \geq \left(1-\frac{1}{4\,(1+\beta)}\right)^k$ und damit

$$\begin{aligned}\omega(M_k) &\geq \frac{1-\epsilon}{1+\epsilon/11}\left(\frac{3}{4}-\frac{8+6\,\epsilon/11}{4}\,\epsilon/11\right)\omega(M^*)\\ &= \frac{\frac{3}{4}-\frac{41}{44}\epsilon+\frac{41}{242}\epsilon^2+\frac{3}{242}\epsilon^3}{1+\epsilon/11}\,\omega(M^*)\\ &> \frac{(\frac{3}{4}-\epsilon)(1+\epsilon/11)}{1+\epsilon/11}\,\omega(M^*) = (3/4-\epsilon)\,\omega(M^*).\end{aligned}$$

Mit Satz 33 (S. 33) wird die erforderliche Güte in einer Laufzeit von $\mathcal{O}((n+m)\cdot\log_{1+\beta}\frac{n}{\alpha}\cdot k)$ erreicht und analog zu Abschnitt 3.3 (S. 21) folgt eine Gesamtlaufzeit von $\mathcal{O}((n+m)\cdot\frac{1}{\epsilon}\log\frac{n}{\epsilon}\log\frac{1}{\epsilon})$.

Im Vergleich zum linearen `Greedy-`$\mathcal{A}_5$ benötigt `Greedy-`$\mathcal{A}_7$ das $\frac{1}{\epsilon}\log\frac{n}{\epsilon}$-fache an Laufzeit, aber erreicht dafür eine Güte von $3/4-\epsilon$ anstatt $2/3-\epsilon$.

Kapitel 5

Empirische Evaluation

Die Algorithmen wurden auf einer Reihe von Graphenklassen getestet, die auch schon von Drake und Hougardy [9] benutzt wurden.

- `G.`r ist eine Gruppe von $(1000 \times r)$-Gittergraphen mit gleichmäßig zufällig erzeugten ganzzahligen Gewichten zwischen $0 \ldots 999$.
- `R10000.`r ist eine Gruppe zufälliger Graphen auf 10000 Knoten mit einer Kantenwahrscheinlichkeit von $r/10000$ und gleichmäßig zufällig erzeugten ganzzahligen Gewichten zwischen $1 \ldots 1000$.

Zu jedem benutzten Parameter r wurde eine Gruppe von 10 Graphen erzeugt und die ermittelten Werte über diese Gruppe gemittelt.

5.1 Das Start-Matching

Beide Algorithmen benötigen ein maximales Matching als Eingabe. Um dies zu erzeugen, wird eine Variante des linearen Algorithmus `MaxMatch` benutzt. Ist das Matching nicht maximal, so gibt es eine Kante, die mit keiner anderen Matchingkante inzident ist. Diese Eigenschaft wird **frei** genannt. `MaxMatch` findet zu jedem Knoten aus der Menge der zum Knoten inzidenten Kanten die größte freie Kante und fügt diese zum Matching hinzu. Obwohl `MaxMatch` keine Güte garantiert, weist das so generierte Start-Matching die in Tabelle 5.1 dargestellten Abweichungen vom Optimum auf.

5.2 Der `Greedy-`$\mathcal{A}_5$ im Vergleich zu Drake & Hougardy

Bezüglich diesen Startlösungen wurde der Algorithmus mit den Resultaten von Drake und Hougardy [9] verglichen. Von ihrem Algorithmus liegen Werte zur Güte nach einer Iteration und nach $MaxI$ Iterationen vor, wobei $\boldsymbol{MaxI}$ die Anzahl von Iterationen ist, nach der sich keine Änderungen mehr ergeben.

Da das Ergebnis mit dem benutzten ϵ variiert, wurde `Greedy-`$\mathcal{A}_5$ mit verschiedenen ϵ-Werten gestartet. Für jedes ϵ wird eine Güte von $2/3-\epsilon$ garantiert. Der Parameter ϵ entstammt daher dem (theoretischen) Bereich $(0 \ldots 2/3]$. Da

Graph	n	m	MM
G10	10000	18990	12,42
G20	20000	38980	12,97
G40	40000	78960	13,5
G60	60000	118940	13,7
G80	80000	158920	13,76
G100	100000	198900	13,92
R10000.5	10000	25009	14,42
R10000.10	10000	49960	12,71
R10000.20	10000	100046	9,55
R10000.30	10000	150075	7,53
R10000.40	10000	200011	6,34
R10000.60	10000	299933	4,86
R10000.80	10000	399994	3,95
R10000.100	10000	499882	3,39

Tabelle 5.1: Abweichung des Start-Matchings vom Optimum (in %)

sich der Wert von ϵ nur auf die Anzahl der Partitionsklassen ($\alpha = \beta = \epsilon/9$) und den damit verbundenen Fehler auswirkt, können auch größere ϵ-Werte benutzt werden.

Die Abbildung 5.1 und die Tabelle 5.2 zeigen die $MaxI$-Werte, also die Anzahl der Iterationen des Algorithmus, welche das Matching verbessern. Insbesondere bei den zufälligen Graphen fallen diese Werte für den `Greedy-`$\mathcal{A}_5$ wesentlich größer aus als für den Algorithmus von Drake und Hougardy. Insbesondere bei der Benutzung großer ϵ-Werte ergeben sich hohe Messwerte, was eine lange Laufzeit zur Folge hat.

Die Abbildungen 5.2 und 5.3 sowie Tabelle 5.3 stellen die Abweichung des Matchings vom Optimum dar, wobei sich auf das berechnete Matching nach einer Iteration und nach $MaxI$ Iterationen des Algorithmus bezogen wird.

Zunächst stellt man fest, dass sich in beiden Fällen die Werte für jedes $\epsilon \leq 2/3$ ähneln. Bei weitergehenden Untersuchungen wurde festgestellt, dass die Werte für $\epsilon \in \{0.01, 0.00\overline{6}, 0.005, 0.00\overline{3}, 0.001, 0.0001, 0.00001\}$ sogar identisch sind. Erklären lässt sich dies damit, dass mit kleinerem ϵ mehr Partitionsklassen erzeugt werden, sich jedoch durch die Ganzzahligkeit der Gewichte kaum Änderungen in der Reihenfolge beim Auslesen der Partitionsklassen ergeben.

Weiterhin fällt bei kleinem ϵ auf, dass die nach einer Iteration erzeugten Matchings allesamt besser sind als die entsprechenden Werte von Drake und Hougardy, insbesondere auf den Gittergraphen unterscheiden sie sich um ca. ein Prozent. Nach $MaxI$ Iterationen sind sie auf den Gittergraphen immer noch geringfügig besser, auf den zufälligen Graphen gleichen sich die Werte mit Drake und Hougardys Algorithmus.

Interessanterweise verhält es sich bei größerem ϵ etwas anders. Nach einer Iteration sind die Werte schlechter als bei kleinem ϵ, auf den zufälligen Graphen sogar schlechter als Drake und Hougardys Werte. Nach $MaxI$ Iterationen jedoch liegen die Werte in der Nähe oder unterhalb den Messwerten für kleine ϵ.

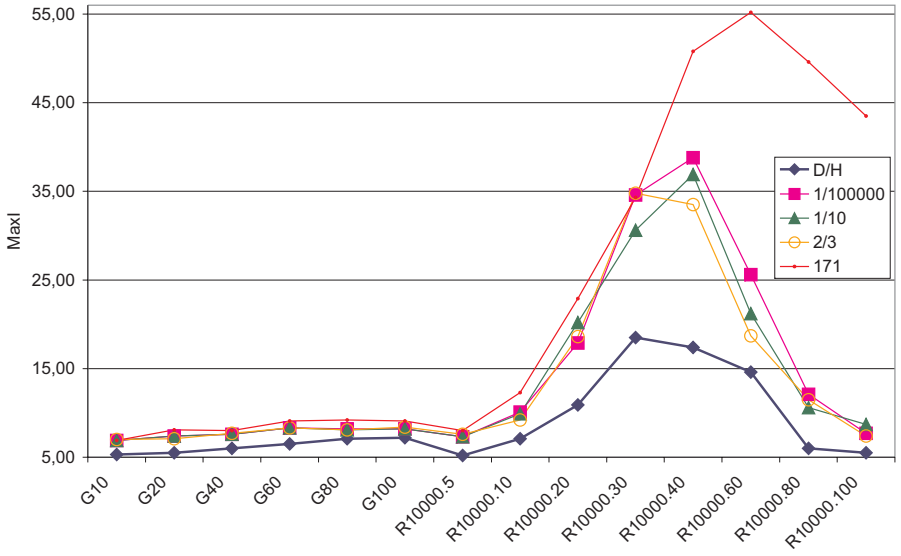

Abbildung 5.1: Anzahl gewinnbringender Iterationen ($MaxI$) für Drake & Hougardy und `Greedy`-$\mathcal{A}_5$ mit verschiedenen ϵ-Werten (s. Legende)

Graph	D/H	`Greedy`-$\mathcal{A}_5$ (bzgl. ϵ)			
		$1/10^5$	1/10	2/3	171
G10	5,30	6,9	6,9	7,0	6,9
G20	5,50	7,4	7,4	7,1	8,1
G40	6,00	7,6	7,6	7,7	8,0
G60	6,50	8,3	8,3	8,3	9,1
G80	7,10	8,2	8,1	8,1	9,2
G100	7,20	8,2	8,2	8,4	9,1
R10000.5	5,20	7,3	7,3	7,6	8,0
R10000.10	7,10	10,1	9,9	9,2	12,3
R10000.20	10,90	17,9	20,2	18,6	22,9
R10000.30	18,50	34,6	30,6	34,8	34,3
R10000.40	17,40	38,8	36,9	33,5	50,8
R10000.60	14,60	25,6	21,2	18,7	55,2
R10000.80	6,00	12,1	10,6	11,5	49,6
R10000.100	5,50	7,7	8,7	7,4	43,5

Tabelle 5.2: Anzahl gewinnbringender Iterationen ($MaxI$)

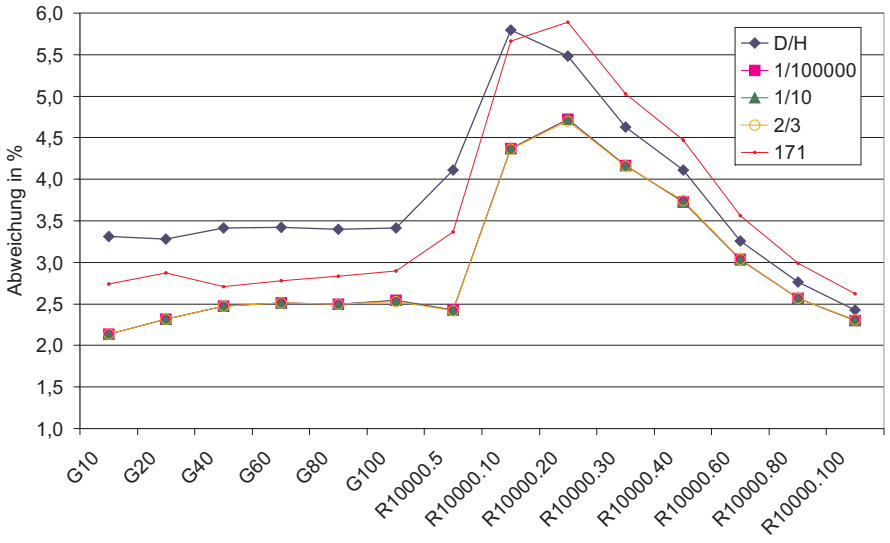

Abbildung 5.2: Abweichung vom Optimum nach einer Iteration

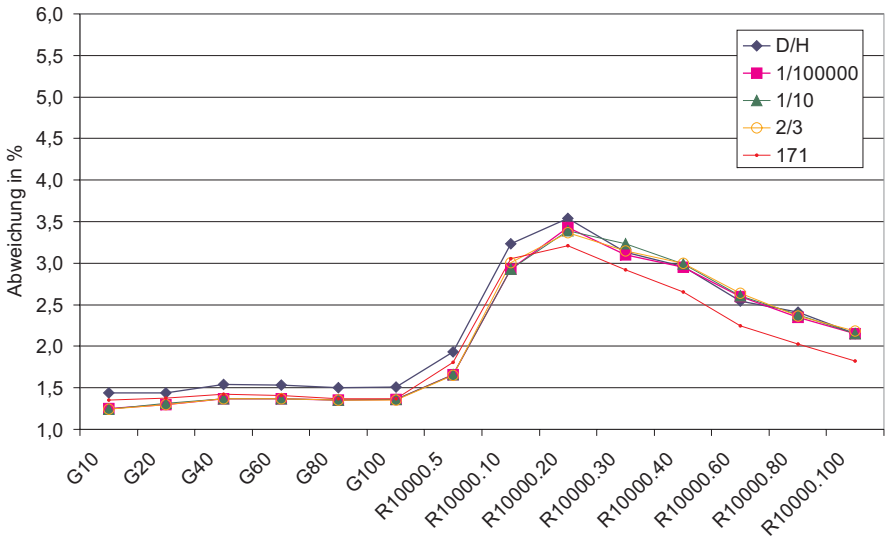

Abbildung 5.3: Abweichung vom Optimum nach $MaxI$ Iterationen

Graph	nach einer Iteration					nach $MaxI$ Iterationen				
	D/H	Greedy-$\mathcal{A}_5$ (bzgl. ϵ)				D/H	Greedy-$\mathcal{A}_5$ (bzgl. ϵ)			
		$1/10^5$	$1/10$	$2/3$	171		$1/10^5$	$1/10$	$2/3$	171
G10	3,31	2,14	2,14	2,14	2,74	1,44	1,25	1,24	1,24	1,35
G20	3,28	2,32	2,32	2,32	2,87	1,44	1,30	1,31	1,30	1,38
G40	3,41	2,47	2,47	2,47	2,71	1,54	1,37	1,37	1,37	1,42
G60	3,42	2,51	2,51	2,51	2,78	1,53	1,37	1,37	1,37	1,41
G80	3,40	2,50	2,50	2,50	2,83	1,50	1,35	1,35	1,35	1,37
G100	3,41	2,54	2,54	2,53	2,90	1,51	1,36	1,36	1,35	1,37
R10000.5	4,11	2,43	2,43	2,42	3,37	1,93	1,66	1,66	1,65	1,81
R10000.10	5,80	4,37	4,37	4,36	5,66	3,23	2,93	2,93	3,01	3,05
R10000.20	5,48	4,72	4,72	4,70	5,89	3,54	3,43	3,39	3,37	3,21
R10000.30	4,63	4,17	4,17	4,16	5,03	3,13	3,10	3,23	3,15	2,92
R10000.40	4,11	3,73	3,73	3,74	4,47	2,96	2,95	2,99	3,00	2,65
R10000.60	3,26	3,04	3,03	3,04	3,56	2,54	2,60	2,61	2,64	2,25
R10000.80	2,76	2,57	2,57	2,57	2,99	2,41	2,35	2,38	2,36	2,03
R10000.100	2,43	2,30	2,30	2,31	2,62	2,15	2,15	2,15	2,18	1,82

Tabelle 5.3: Abweichung vom Optimum (in %)

5.3 Sinnvolle Iterationswerte für den Greedy-$\mathcal{A}_5$

Mitunter ist es nicht sinnvoll, alle $MaxI$ Iterationen auszuführen, da die dafür benötigte Zeit nicht im Verhältnis zum Gewinn steht. Um eine Güte von $2/3-\epsilon$ zu garantieren, müssten theoretisch $\log_{29/20} 1/\epsilon$ Iterationen durchgeführt werden (s. Abschnitt 3.3, S. 21), bei den untersuchten Beispielen reichen jedoch bereits wenige Iterationen um eine gute Approximation zu erhalten. Die theoretischen Mindest-Iterationen sind in folgender Tabelle aufgeführt.

ϵ	$1/10^5$	$1/10^4$	$1/10^3$	$1/10^2$	$1/10$	$1/3$	$1/2$	$\geq 2/3$
$\log_{29/20} 1/\epsilon$	30,9	24,8	18,5	12,4	6,2	3,0	1,9	0[1]

Bevor man sich auf eine Iterationsgröße festlegt, muss man einen ebenfalls sinnvollen Parameter ϵ wählen, da die Genauigkeit und Laufzeit einer einzelnen Iteration davon abhängt. Je kleiner ϵ ist, desto kleiner sind α und β und umso größer ist die Anzahl benutzter Partitionsklassen. Mehr Partitionsklassen wiederum versprechen kleinere Fehler, kosten im Gegenzug jedoch mehr Laufzeit.

Die Abbildungen 5.5 bis 5.10 zeigen die Abweichung vom Optimum in Abhängigkeit von ϵ nach einer bis sechs Iterationen. Es zeigt sich, dass innerhalb der ersten sechs Iterationen ein ϵ kleiner als 1 ziemlich gleichwertige Lösungen verspricht, jedoch ab 1 die Qualität der Lösungen auf den benutzten Graphenklassen nachlässt. Tabelle 5.4 zeigt, dass der Zeitaufwand mit größerem ϵ (mitunter nur minimal) fällt. Aus diesen Gründen scheint ein ϵ von 3/4 für die benutzten Daten angebracht.

Die Abbildung 5.4 zeigt die Abweichung vom Optimum nach jeder Iteration bei $\epsilon = 3/4$. MM bezeichnet dabei die mit MaxMatch berechnete Startlösung.

[1] $\epsilon \geq 2/3$ führt zu einer Approximationsgüte von $2/3-\epsilon \leq 0$, wofür keine Iterationen nötig sind.

	0,01	$0,0\overline{3}$	0,05	$0,0\overline{6}$	0,1	$0,\overline{3}$	0,5	$0,\overline{6}$	0,75	1	9	63	171
G10	0,76	0,74	0,74	0,74	0,74	0,74	0,74	0,75	0,74	0,74	0,74	0,74	0,74
G20	1,56	1,55	1,54	1,54	1,54	1,54	1,53	1,53	1,53	1,53	1,49	1,49	1,49
G40	3,10	3,06	3,08	3,06	3,07	3,05	3,06	3,05	3,04	3,04	2,96	2,92	2,92
G60	4,65	4,62	4,61	4,61	4,60	4,58	4,59	4,60	4,57	4,54	4,41	4,35	4,33
G80	6,25	6,19	6,18	6,18	6,19	6,13	6,14	6,12	6,09	6,08	5,87	5,80	5,78
G100	7,89	7,78	7,84	7,82	7,81	7,75	7,73	7,73	7,68	7,64	7,38	7,26	7,24
R10000.5	1,12	1,10	1,11	1,09	1,09	1,08	1,09	1,08	1,09	1,09	1,11	1,10	1,10
R10000.10	2,09	2,07	2,10	2,11	2,06	2,06	2,06	2,07	2,06	2,05	2,05	2,05	2,06
R10000.20	3,95	3,94	3,93	3,93	3,95	3,94	3,94	3,93	3,90	3,89	3,89	3,92	3,96
R10000.30	5,82	5,77	5,77	5,76	5,80	5,80	5,80	5,77	5,74	5,73	5,71	5,74	5,78
R10000.40	7,68	7,65	7,64	7,65	7,69	7,67	7,64	7,63	7,62	7,62	7,54	7,59	7,66
R10000.60	11,59	11,59	11,58	11,58	11,57	11,59	11,53	11,58	11,53	11,49	11,39	11,29	11,41
R10000.80	15,80	15,75	15,74	15,79	15,84	15,74	15,78	15,74	15,65	15,64	15,36	15,22	15,22
R10000.100	19,92	19,83	19,75	19,83	20,07	19,87	19,93	19,86	19,76	19,52	19,24	18,81	18,86

Tabelle 5.4: Zeitaufwand (in s) für sechs Iterationen in Abhängigkeit von ϵ (AMD Athlon 1GHz 650MB)

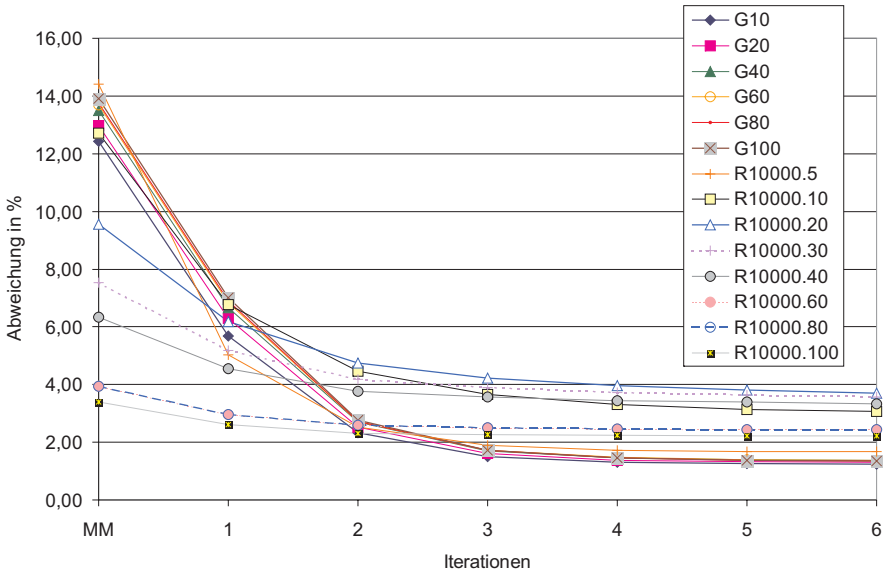

Abbildung 5.4: Abweichung vom Optimum nach jeder Iteration bei $\epsilon = 3/4$

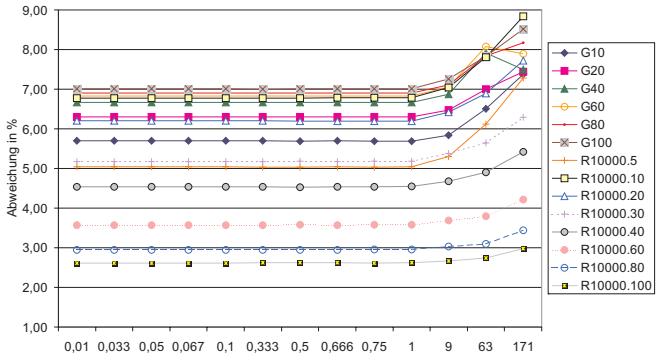

Abbildung 5.5: Abweichung vom Optimum nach einer Iteration

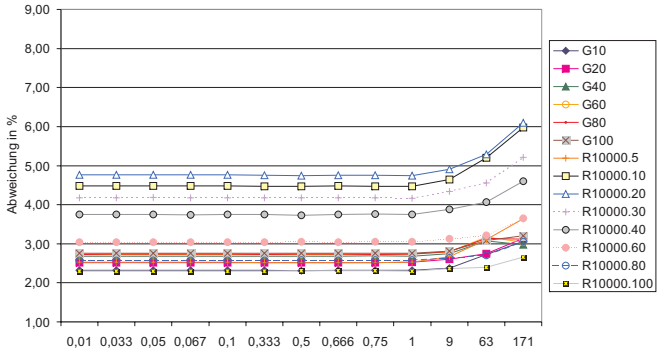

Abbildung 5.6: Abweichung vom Optimum nach zwei Iterationen

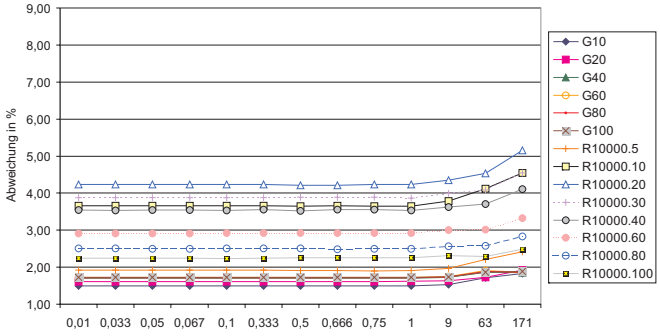

Abbildung 5.7: Abweichung vom Optimum nach drei Iterationen

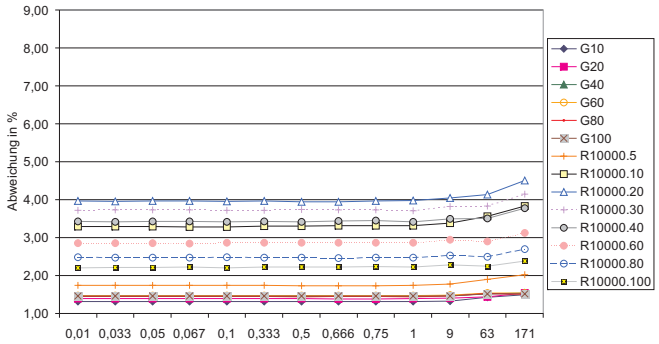

Abbildung 5.8: Abweichung vom Optimum nach vier Iterationen

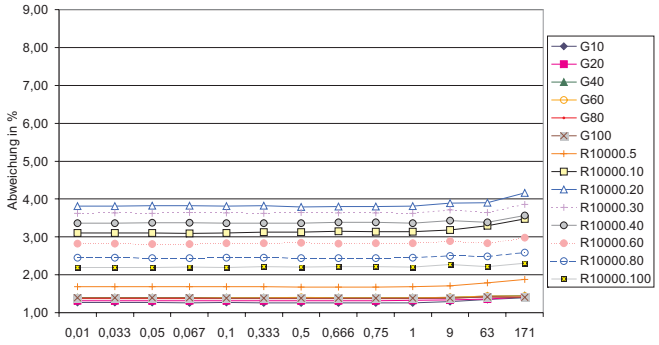

Abbildung 5.9: Abweichung vom Optimum nach fünf Iterationen

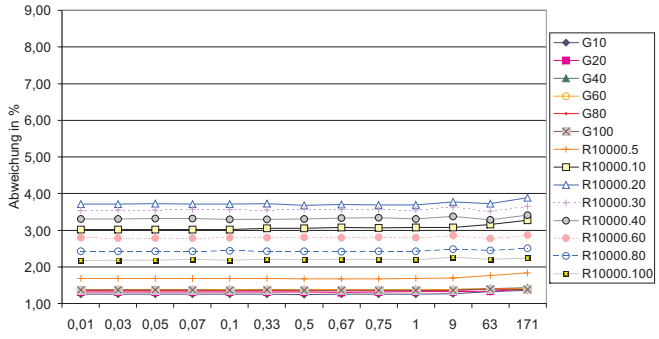

Abbildung 5.10: Abweichung vom Optimum nach sechs Iterationen

	1	2	3	4	5	6
G10	0,13	0,13	0,12	0,12	0,12	0,12
G20	0,27	0,27	0,25	0,25	0,25	0,24
G40	0,54	0,54	0,51	0,49	0,48	0,48
G60	0,82	0,83	0,76	0,73	0,72	0,71
G80	1,09	1,11	1,02	0,97	0,95	0,95
G100	1,38	1,41	1,29	1,23	1,19	1,18
R10000.5	0,21	0,18	0,18	0,17	0,17	0,18
R10000.10	0,37	0,36	0,34	0,33	0,33	0,33
R10000.20	0,65	0,67	0,66	0,64	0,64	0,64
R10000.30	0,93	0,99	0,96	0,96	0,95	0,95
R10000.40	1,20	1,31	1,29	1,28	1,27	1,27
R10000.60	2,31	2,70	2,67	2,66	2,66	2,65
R10000.80	2,31	2,70	2,67	2,66	2,66	2,65
R10000.100	2,84	3,37	3,40	3,39	3,37	3,39

Tabelle 5.5: Zeitaufwand (in s) für jede einzelne Iterationen bei $\epsilon = 3/4$

Man sieht, dass die Gewinne sich bei jeder Iteration verkleinern und ab der dritten Iteration nur noch sehr gering sind. Da der zeitliche Aufwand laut Tabelle 5.5 jedoch für jede Iteration ungefähr gleich ist, sind wohl drei Iterationen angebracht, da danach der zeitliche Aufwand nicht mehr im Verhältnis zum Gewinn steht.

5.4 Der `Greedy`-$\mathcal{A}_7$ im Vergleich

Die Abbildung 5.11 und Tabelle 5.6 vergleichen den `Greedy`-$\mathcal{A}_7$ bzgl. dem Parameter $\epsilon \in \{0.75, 1, 11\}$ mit dem `Greedy`-$\mathcal{A}_5$ und dem Algorithmus von Drake und Hougardy [9]. Da die Berechnung für $\epsilon < 11$ für große Graphen sehr lange dauert, fehlen einige Einträge. Aus der Grafik wird deutlich, dass nach $MaxI$ Iterationen der `Greedy`-$\mathcal{A}_7$ auf den Gittergraphen wesentlich besser abschneidet als `Greedy`-$\mathcal{A}_5$ oder Drake und Hougardys Algorithmus. Auf den zufälligen Graphen sind die Lösungen gleichwertig.

In Abbildung 5.12 und Tabelle 5.7 sind die dafür durchgeführten Iterationen aufgelistet. Es fällt auf, dass es erheblich weniger gewinnbringende Iterationen als beim `Greedy`-$\mathcal{A}_5$ gibt, auf den zufälligen Graphen wird sogar das Minimum der verglichenen Algorithmen angenommen.

Wie angesprochen benötigt der `Greedy`-$\mathcal{A}_7$ auf großen Graphen sehr viel Laufzeit. Die theoretisch benötigte Laufzeit für eine Iteration beträgt $\mathcal{O}((n + m) \cdot \log_{1+\beta} \frac{n}{\alpha})$ (Satz 33, S. 33). Da die Arme sortiert werden, ergibt sich eine Laufzeit von $\mathcal{O}((n + m) \cdot \log(n + m))$, also ist sowohl beim Ansteigen der Knoten als auch beim Ansteigen der Kanten eine superlineare Laufzeit zu erwarten. Hinzu kommt, dass durch mehr Knoten und Kanten mehr Alternativen in die Partitionsklassen eingefügt werden, welche aktualisiert werden müssen. Viele Operationen sind zwar amortisiert konstant, haben jedoch im schlimmsten Fall einen einmaligem Aufwand von $\mathcal{O}(m \log_{1+\beta} \frac{n}{\alpha})$. Je mehr Kanten bzw. Alternativen in der Datenstrukur enthalten sind, umso mehr fällt dieser einmalige

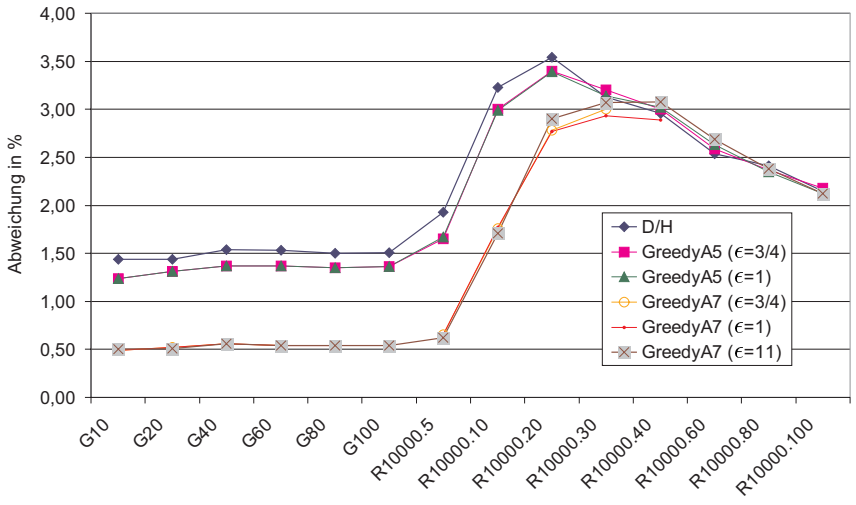

Abbildung 5.11: Abweichung vom Optimum nach $MaxI$ Iterationen

	D/H	Greedy-$\mathcal{A}_5$		Greedy-$\mathcal{A}_7$		
		3/4	1	3/4	1	11
G10	1,44	1,24	1,24	0,49	0,49	0,50
G20	1,44	1,31	1,31	0,52	0,52	0,51
G40	1,54	1,37	1,37	0,56	0,56	0,56
G60	1,53	1,37	1,37	0,54	0,54	0,54
G80	1,50	1,35	1,35		0,54	0,54
G100	1,51	1,36	1,36			0,54
R10000.5	1,93	1,65	1,67	0,65	0,65	0,62
R10000.10	3,23	3,00	2,99	1,76	1,77	1,71
R10000.20	3,54	3,40	3,39	2,78	2,77	2,90
R10000.30	3,13	3,20	3,14	3,00	2,93	3,07
R10000.40	2,96	3,00	3,02		2,89	3,08
R10000.60	2,54	2,59	2,63			2,69
R10000.80	2,41	2,37	2,35			2,38
R10000.100	2,15	2,18	2,12			2,12

Tabelle 5.6: Abweichung vom Optimum (in %) nach $MaxI$ Iterationen

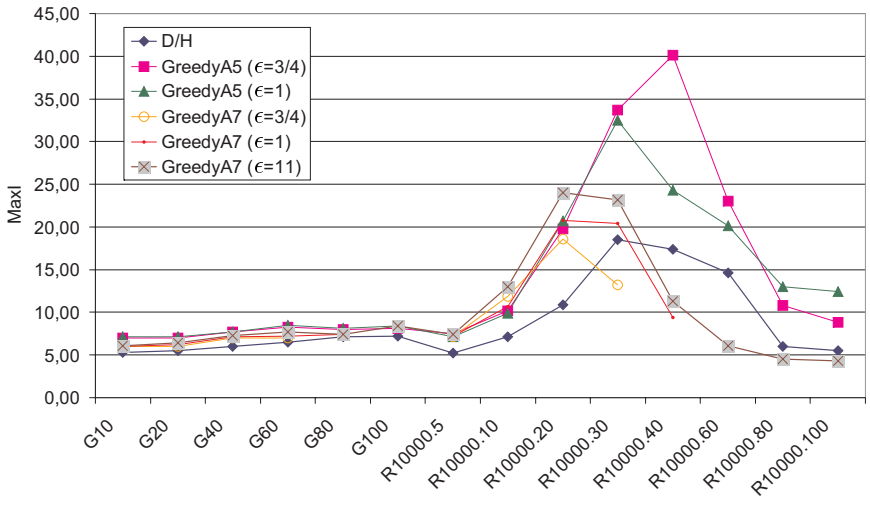

Abbildung 5.12: Anzahl gewinnbringender Iterationen ($MaxI$)

	D/H	Greedy-$\mathcal{A}_5$		Greedy-$\mathcal{A}_7$		
		3/4	1	3/4	1	11
G10	5,3	7,0	7,1	6,0	6,0	6,1
G20	5,5	7,0	7,1	6,0	6,2	6,4
G40	6,0	7,7	7,7	7,0	7,1	7,3
G60	6,5	8,3	8,5	7,0	7,2	7,7
G80	7,1	8,0	8,1		7,4	7,4
G100	7,2	8,1	8,4			8,4
R10000.5	5,2	7,5	7,1	7,1	7,2	7,4
R10000.10	7,1	10,2	9,9	11,9	10,7	13,0
R10000.20	10,9	19,8	20,7	18,6	20,8	24,0
R10000.30	18,5	33,7	32,5	13,2	20,4	23,2
R10000.40	17,4	40,1	24,3		9,4	11,3
R10000.60	14,6	23,0	20,1			6,1
R10000.80	6,0	10,8	13,0			4,5
R10000.100	5,5	8,8	12,4			4,3

Tabelle 5.7: Anzahl gewinnbringender Iterationen ($MaxI$)

Aufwand ins Gewicht.

In Abbildung 5.13 und Tabelle 5.8 werden die Laufzeiten auf den Graphen mit dem Greedy-$\mathcal{A}_5$ verglichen. Man sieht, wie zu erwarten, dass der Greedy-$\mathcal{A}_7$ deutlich mehr Zeit benötigt als Greedy-$\mathcal{A}_5$, theoretisch unterscheiden sie sich um den Faktor $\mathcal{O}(\frac{1}{\epsilon} \cdot \log \frac{n+m}{\epsilon})$. In Tabelle 5.1 (S. 39) sind die Knoten- und Kantenzahlen der Graphen aufgelistet. Auf den Gittergraphen steigen sowohl m und n mit wachsendem Index, auf den zufälligen Graphen steigt nur die Kantenzahl. Der Greedy-$\mathcal{A}_5$ weist in der Abbildung wie erwartet einen linearen Laufzeitaufwand auf, der Greedy-$\mathcal{A}_7$ hingegen hat wie festgestellt einen superlinearen Aufwand.

Mit Abschnitt 3.3, S. 21, ist für ϵ eine Abhängigkeit von $\mathcal{O}(\log_{1+\beta} \frac{n}{\alpha}) \subseteq \mathcal{O}(\frac{1}{\epsilon} \cdot \ln(\frac{n}{\epsilon}))$ zu erwarten. Die Anzahl der Partitionsklassen und damit die Anzahl partitionierter Alternativen steigt mit der Verkleinerung von ϵ, nicht zuletzt gibt es nun auch mehr Alternativen, die nicht zur Partitionsklasse Null gehören. Wie bei den Knoten und Kanten fällt bei kleinerem ϵ somit der einmalige Aufwand der amortisiert konstanten Funktionen mehr ins Gewicht.

Die Abbildung 5.14 und Tabelle 5.8 vergleicht die Laufzeiten des Greedy-$\mathcal{A}_7$ mit unterschiedlichem Parameter ϵ. Wie festgestellt steigen sie mit fallendem Parameter stark an.

Die Abbildung 5.15 zeigt die Abweichung vom Optimum nach jeder Iteration bei $\epsilon = 11$. Analog zum Greedy-$\mathcal{A}_5$ verkleinert sich der Zugewinn mit jeder Iteration. Je nach Iterationsdauer rentieren sich zwei bis drei Iterationen. Der Zeitaufwand für jede einzelne Iteration ist in Tabelle 5.9 dargestellt und ist pro Graphenklasse nahezu gleich groß für alle sechs Iterationen. Die theoretische Mindestanzahl von Iterationen, die durchgeführt werden müssten, um eine Güte von $3/4 - \epsilon$ zu garantieren, ist in folgender Tabelle dargestellt.

ϵ	1/10	1/2	2/3	$\geq 3/4$
$\log_{37/26} 1/\epsilon$	6,5	1,9	1,1	0[2]

5.5 Zusammenfassung

Aufgrund der hier untersuchten Graphen scheint die Benutzung des Greedy-$\mathcal{A}_5$ vorteilhafter zu sein, da dieser erheblich schneller als der Greedy-$\mathcal{A}_7$ ist. Ebenso ist er einfacher zu implementieren. Folgende Paramter stellen sich auf den benutzten Graphen als günstig heraus:

Greedy-$\mathcal{A}_5$ mit drei Iterationen und $\epsilon = 3/4$
Greedy-$\mathcal{A}_7$ mit zwei bis drei Iterationen und $\epsilon = 11$

Der Greedy-$\mathcal{A}_5$ erreicht bei dieser Untersuchung nach einer Iteration ein Ergebnis, welches im Schnitt ungefähr $1,5$ Prozent besser ist als das nach einer Iteration des Algorithmus von Drake und Hougardy. Nach $MaxI$ Iterationen unterscheiden sie sich nur noch um ca. $0,2$ Prozent. Nach drei Iterationen ist also mit ca. einem Prozent Unterschied zu rechnen. Der Greedy-$\mathcal{A}_7$ ist mit ungefähr zwei Prozent weniger als Drake und Hougardys Algorithmus zwar noch besser, benötigt aber auch mitunter das Zehnfache an Laufzeit gegenüber Greedy-$\mathcal{A}_5$.

[2] $\epsilon \geq 3/4$ führt zu einer Approximationsgüte von $3/4 - \epsilon \leq 0$, wofür keine Iterationen nötig sind.

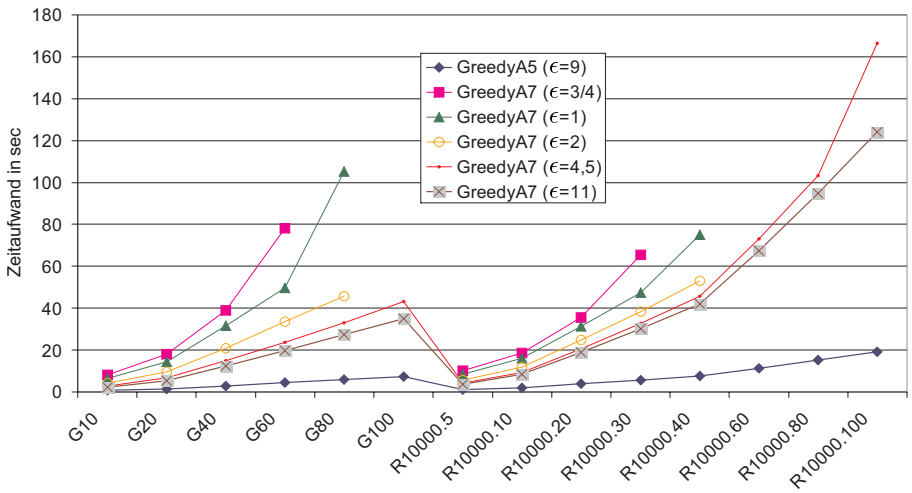

Abbildung 5.13: Zeit für sechs Iterationen (AMD Athlon 1GHz 650MB)

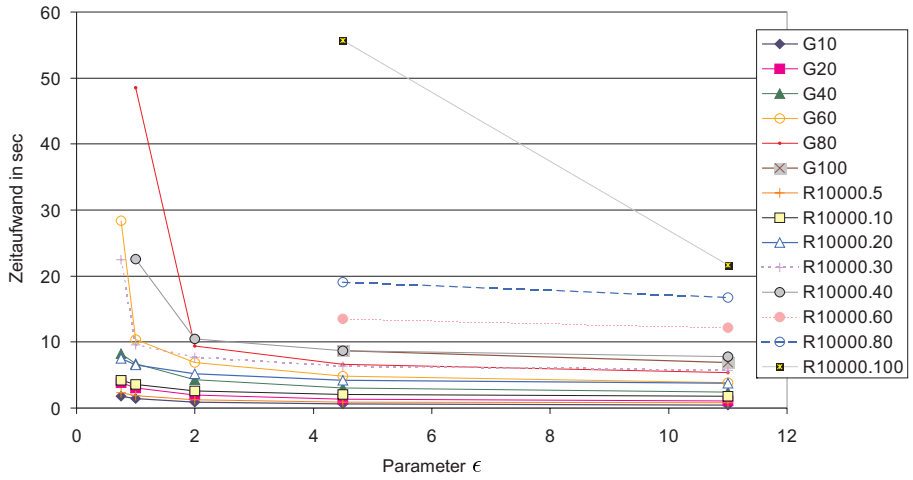

Abbildung 5.14: Zeit für die erste Iteration bzgl. ϵ (AMD Athlon 1GHz 650MB)

Graph	für eine Iteration				für sechs Iterationen			
	Greedy-$\mathcal{A}_5$	Greedy-$\mathcal{A}_7$			Greedy-$\mathcal{A}_5$	Greedy-$\mathcal{A}_7$		
	9	0,75	1	11	9	0,75	1	11
G10	0,13	1,77	1,40	0,48	0,74	8,26	6,59	2,36
G20	0,26	3,85	3,06	1,09	1,49	18,00	14,45	5,42
G40	0,52	8,25	6,68	2,44	2,96	38,81	31,54	12,30
G60	0,78	28,39	10,42	3,86	4,41	78,23	49,59	19,62
G80	1,04		48,57	5,37	5,87		105,33	27,28
G100	1,31			6,89	7,38			35,09
R10000.5	0,21	2,33	1,89	0,77	1,11	10,27	8,38	3,55
R10000.10	0,37	4,17	3,54	1,76	2,05	18,73	15,99	8,42
R10000.20	0,65	7,50	6,57	3,73	3,89	35,64	31,36	18,92
R10000.30	0,92	22,49	9,62	5,75	5,71	65,47	47,53	30,26
R10000.40	1,19		22,53	7,78	7,54		74,97	41,97
R10000.60	1,74			12,20	11,39			67,57
R10000.80	2,29			16,74	15,36			94,92
R10000.100	2,83			21,63	19,24			124,09

Tabelle 5.8: Zeit (in s) für eine und sechs Iterationen in Abhängigkeit von ϵ

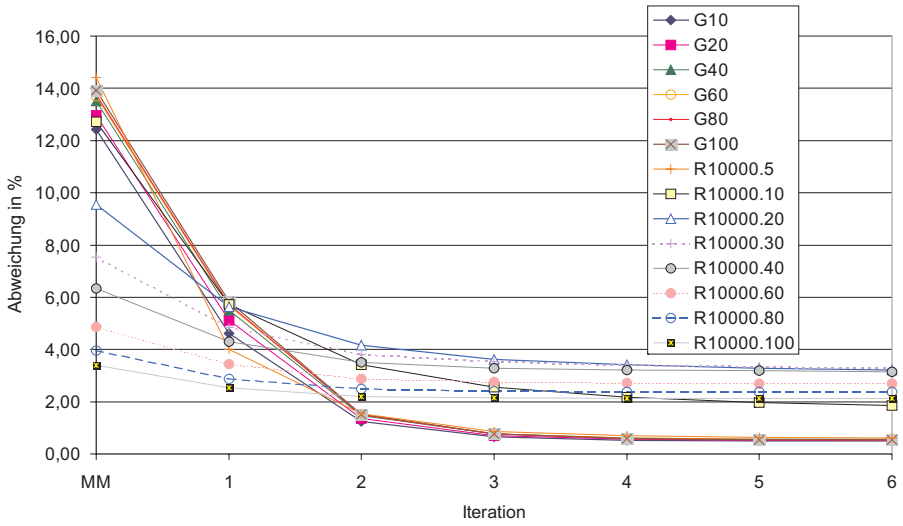

Abbildung 5.15: Abweichung vom Optimum nach jeder Iteration bei $\epsilon = 11$

	1	2	3	4	5	6
G10	0,48	0,42	0,38	0,36	0,36	0,36
G20	1,09	0,97	0,87	0,84	0,83	0,82
G40	2,44	2,21	1,99	1,91	1,88	1,87
G60	3,86	3,53	3,19	3,05	3,01	2,98
G80	5,37	4,90	4,43	4,24	4,18	4,16
G100	6,89	6,30	5,70	5,46	5,39	5,35
R10000.5	0,77	0,63	0,56	0,54	0,53	0,52
R10000.10	1,76	1,49	1,36	1,30	1,26	1,25
R10000.20	3,73	3,37	3,08	2,95	2,91	2,88
R10000.30	5,75	5,38	4,94	4,78	4,72	4,69
R10000.40	7,78	7,47	6,87	6,66	6,61	6,58
R10000.60	12,2	11,99	11,08	10,83	10,75	10,72
R10000.80	16,74	16,83	15,65	15,29	15,23	15,18
R10000.100	21,63	22,07	20,48	20,04	19,93	19,94

Tabelle 5.9: Zeitaufwand (in s) für jede einzelne Iterationen bei $\epsilon = 11$

Literaturverzeichnis

[1] D. Avis; *A Survey of Heuristics for the Weighted Matching Problem*; Networks, Vol. 13 (1983); 475-493

[2] C.E. Bell; *Weighted matching with vertex weights: an application to scheduling training sessions in NASA space shuttle cockpit simulators*; European Journal of Operational Research 73 (1994); 443-449

[3] M. Campêlo-Neto, S. Klein; *Maximal Vertex-Weighted Matching in Strongly Chordal Graphs*; Discrete Applied Mathematics 84, Issue 1-3 (May 1998); 71 - 77

[4] W. Cook, A. Rohe; *Computing minimum-weight perfect matchings*; INFORMS Journal on Computing 11 (1999); 138-148

[5] J. Dénes, A.D. Keedwell; *Latin squares and their applications*; Academic Press, 1974

[6] D.E. Drake, S. Hougardy; *A Simple Approximation Algorithm for the Weighted Matching Problem*; Information Processing Letters 85 (2003); 211-213

[7] D.E. Drake, S. Hougardy; *Linear Time Local Improvements for Weighted Matchings in Graphs*; In: Workshop on Efficient Algorithms (WEA) 2003, K. Jansen, M. Margraf, M. Mastrolli, J.D.P. Rolim (eds.), LNCS 2647, Springer 2003; 107-119

[8] D.E. Drake, S. Hougardy; *Improved Linear Time Approximation Algorithms for Weighted Matchings*; In: Approximation, Randomization, and Combinatorial Optimization, (Approx/Random 03), S.Arora, K.Jansen, J.D.P.Rolim, A.Sahai (Eds.), LNCS 2764, Springer 2003; 14-23

[9] D.E. Drake, S. Hougardy; *A Linear Time Approximation Algorithm for Weighted Matchings in Graphs*; Preprint Humboldt-Universität zu Berlin, Dec 2003

[10] J. Edmonds; *Maximum matching and a polyhedron with 0,1-vertices*; J. Res. Nat. Bur. Standards 69B (1965); 125-130

[11] J. Edmonds, E.L. Johnson; *Matching, Euler tours and the Chinese postman*; Math. Progr. 5, 1973; 88-124

[12] M.L. Fredmann, R.E. Tarjan; *Fibonacci Heaps and their uses in improved network optimization problems*; J. ACM 34, 1987; 596-615

[13] H.N. Gabow; *An efficient implementation of Edmond's algorithm for maximum matching on graphs*; Journal of the ACM 23 (1976); 221-234

[14] H.N. Gabow; *Data Structures for Weighted Matching and Nearest Common Ancestors with Linking*; SODA 1990; 434-443

[15] H.N. Gabow, Z. Galil, T.H. Spencer; *Efficient implementation of graph algorithms using contraction*; J. ACM 36 (1989); 815-853

[16] H.N Gabow, R.E. Tarjan; *Faster Scaling Algorithms for General Graph-Matching Problems*; J ACM 38:4 (1991); 815-853

[17] Z. Galil, S. Micali, H.N. Gabow; *An O(EV log V) algorithm for finding a maximal weighted matching in general graphs*; SIAM Journal on Computing 15 (1986); 120-130

[18] Th. Emden-Weinert, S. Hougardy, B. Kreuter, H.J. Prömel, A. Steger; *Einführung in Graphen und Algorithmen*; URL=`http://www.informatik.hu-berlin.de/Institut/struktur/algorithmen/ga/chp4.ps`; 94-127

[19] S.P. Fekete, H. Meijer, A. Rohe, W. Tietze; *Solving a „Hard" Problem to Approximate an „Easy" One: Heuristics for Maximum Matchings and Maximum Traveling Salesman Problems*; Journal of Experimental Algorithms 7 (2002); article 11

[20] B. Kalyanasundaram, K. Pruhs; *Online Weighted Matching*; Journal of Algorithms 14 (1993); 478-488

[21] G. Karypis, V. Kumar; *A fast and high quality multilevel scheme for partitioning irregular graphs*; SIAM J. Sci. Comput. 20:1 (1998); 359-392

[22] H.W. Kuhn; *The Hungarian method for the assignment problem*; Naval Res. Logist. Quart. 2, 1955; 83-97

[23] H.W. Kuhn; *Variants of the Hungarian method for assignment problems*; Naval Res. Logist. Quart. 3, 1956; 253-258

[24] E.L. Lawler; *Combinatorical Optimization: Networks and Matroids*; Holt, Rinehart and Winston, New York, 1976

[25] R.J. Lipton, R.E. Tarjan; *Applications of a planar separator theorem*; SIAM Journal on Computing 9 (1980); 615-627

[26] K. Mehlhorn, G. Schäfer; *Implementation of $\mathcal{O}(nm \log n)$ Weighted Matchings in General Graphs: The Power of Data Structures*; In: S. Näher, D. Wagner (eds.), 4th International Workshop on Algorithm Engineering (WAE) 2000, Saarbrücken, Germany, September 5-8, LNCS 1982, Springer2001; 23-38

[27] K. Mei-Ko; *Graphic programming using odd or even points*; Chinese Math. 1, 1962; 273-277

[28] S. Micali and V.V. Vazirani; *An $\mathcal{O}(\sqrt{V}E)$ Algorithm for Finding Maximum Matching in General Graphs*; Proc. 21st Annual IEEE Symposium on Foundations of Computer Science (1980); 17-27

[29] B. Monien, R. Preis, R. Diekmann; *Quality Matching and Local Improvement for Multilevel Graph-Partitioning*; Parallel Computing, 26(12), 2000; 1609-1634

[30] M. Müller-Hannemann, A. Schwartz; *Implementing Weighted b-Matching Algorithms: Towards a Flexible Software Design*; ACM Journal of Experimental Algorithmics, Volume 4 (1999); Article 7

[31] S. Pettie, P. Sanders; *A Simpler Linear Time* $2/3 - \epsilon$ *Approximation for Maximum Weight Matching*; Information Processing Letters (2004) in press

[32] A. Pinar, A. Pothen; *Matroids and Signed Graphs: The nice basis problem*; FoCM'02, URL=`http://www.math.umn.edu/~focm/c_/Pothen.pdf`

[33] R. Preis; *Linear Time* $1/2$*-Approximation Algorithm for Maximum Weighted Matching in General Graphs*; Symposium on Theoretical Aspects of Computer Science (STACS) 1999, C. Meinel, S. Tison (eds.), LNCS 1563, Springer 1999; 259-269

[34] R. Preis; *Analytical Methods for Multilevel Graph-Partitioning*; GI Jahrestagung (1999); 223–230

[35] E.A. Riskin, R. Ladner, R.-Y. Wang, L.E. Atlas; *Index assignment for progressive transmission of full-search vector quantization*; IEEE Transactions on Image Processing 3 (1994); 307-312

[36] R.E. Tarjan; *Data Structures and Network Algorithms*; CBMS-NSF Regional Conference Series in Applied Mathematics, Vol. 44, SIAM 1983

[37] R. Uehara, Z. Chen; *Parallel Approximation Algorithms for Maximum Weighted Matching in General Graphs*; Information Processing Letters 76(1-2) (2000); 13-17

[38] V.V. Vazirani; *A Theory of Alternating Paths and Blossoms for Proving Correctness of the* $\mathcal{O}(\sqrt{V}E)$ *Maximum Matching Algorithm*; Combinatorica 14:1 (1994); 71-109

[39] M. Wattenhofer, R. Wattenhofer; *Distributed Weighted Matching*; Distributed Computing, 18th International Conference, DISC 2004

[40] M. Wattenhofer, R. Wattenhofer; *Fast and Simple Algorithms for Weighted Perfect Matching*; Cologne-Twente Workshop on Graph Theory, CTW 2004

[41] M. Yannakakis, F. Gavril; *Edge dominating sets in graphs*; SIAM J. Appl. Math., 38(3), 1980; 364-372

[42] M. Zito; *Linear Time Maximum Induced Matching Algorithm for Trees*; Nordic Journal of Computing 7 (2000); 58-63