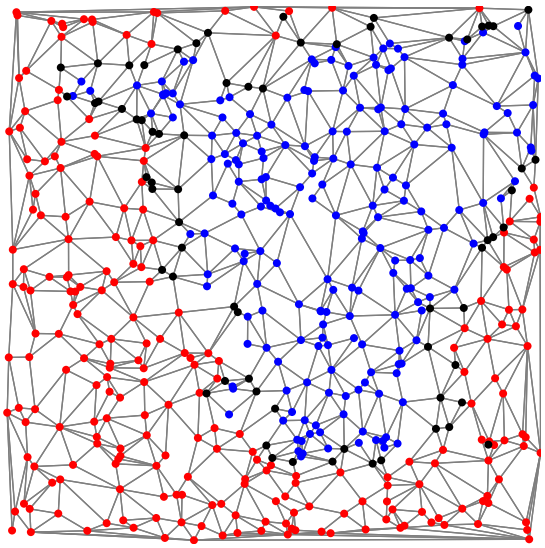
Institut für Theoretische Informatik
Fakultät für Informatik
Universität Karlsruhe (TH)

Experimentelle Analyse von Freiheitsgraden im Planar-Separator-Theorem

Studienarbeit
von

Tirdad Rahmani



Hauptreferentin : Prof. Dr. Dorothea Wagner

Betreuer : Dipl.-Math. Martin Holzer

Inhaltsverzeichnis

Zusammenfassung

Diese Arbeit befasst sich mit dem *Planar-Separator-Theorem* (PST) von Lipton & Tarjan [1]. In diesem Theorem wird ein planarer Graph durch das Wegnehmen einer bestimmten Anzahl von Knoten (Separatorknoten) in zwei Teilkomponenten zerlegt. Dabei werden bei der Separierung die Schranken $4 \cdot \sqrt{n}$ für die Separatorgröße und $2/3 \cdot n$ für die Balanciertheit der Zerlegung eingehalten, wobei n die Knotenanzahl des Eingabegraphen kennzeichnet. Das Theorem kann man generell in drei Phasen einteilen. In der ersten Phase wird versucht, den Graphen anhand eines Breitensuchbaums zu separieren. Falls diese Separation die Schranken des Theorems nicht einhält, so wird ein Teilgarph des Gesamtgraphen anhand von einigen Regeln ausgewählt und trianguliert. Darin wird dann ein Separator gefunden, der in der dritten Phase zu einem Separator des Gesamtgraphen, mit Hilfe des Breitensuchbaums in der ersten Phase, ergänzt wird.

In dieser Studie untersuchen wir, welche Auswirkungen die im PST verwendeten aufspannenden Bäume auf die Separatorgröße und Balance haben und ob eine Verbesserung dieser Werte durch geeignete Konstruktion von Bäumen, angepasst an die verschiedenen Phasen des Theorems, möglich ist. Desweiteren wird der Einfluss der Triangulierung auf die algorithmische Performanz analysiert und eine neue Variante der Triangulierung untersucht, die aus Kombination der Breitensuche mit der ursprünglichen Triangulierung entstanden ist. Aus diesen Untersuchungen ergaben sich einige Heuristiken, die auf verschiedene Graphklassen angewendet und getestet werden. Durch die Anwendung dieser Heuristiken konnten deutlich kleinere Separatoren mit einem geringen Zusatzaufwand gefunden werden, was dazu beigetragen hat, dass der Erwartungswert der Separatorgröße in praktischen Anwendungen sich stark verringert.

1 Einleitung

1.1 Vorstellung

Wenn man von den *Planar-Separator-Algorithms* spricht, dann versteht man darunter die Angabe eines Algorithmus, der in einem gegebenen planaren Graphen eine kleine Anzahl von Knoten (Separator) auswählt und entfernt, so dass der Graph in zwei möglichst gleich große Teilgraphen zerfällt. Solche Separatoren spielen bei Graphalgorithmen, die auf dem *Divide-and-Conquer* Prinzip beruhen, eine wichtige Rolle. Bei diesem Prinzip, wird ein Graph durch die Wegnahme der Separatorknoten in zwei Teilgraphen zerlegt und der Algorithmus rekursiv auf die so entstandenen Teilgraphen weiter angewendet. Die Gesamtlösung setzt sich dann aus diesen Teillösungen zusammen. Hierbei geht die Separatorgröße entscheidend in die Laufzeit des Zusammensetzens ein. Deshalb eignen sich typischerweise eher kleine Separatoren für diesen Ansatz als große Separatoren. Ein weiterer Faktor, welcher in die Gesamtlaufzeit mit eingeht, ist die Rekursionstiefe des Verfahrens. Die Rekursionstiefe hängt von der Größe der entstandenen Teilgraphen ab. Im Idealfall hätte man balancierte Zerlegungen, bei denen die Teilgraphen etwa gleich groß sind. Aus dieser Feststellung ergibt sich das folgende Optimierungsproblem.

Problem: *Minimum-Balanced-Separator-Problem.*
Gegeben: Graph $G = (V, E)$.
Gesucht: Eine Partition von V in drei Mengen V_1, V_2, und S, wobei S ein Separator mit minimaler Kardinalität ist, der V_1 und V_2 trennt, so dass:

$$|V_1|,\ |V_2| \leq \alpha \cdot |V| \quad \textit{und} \quad 1/2 \leq \alpha < 1.$$

Dieses Optimierungsproblem ist für beliebige Graphen NP-schwer. Für $\alpha = 1/2$ nennt man das Problem *Minimum-Bisection-Problem*. Es ist nicht bekannt, ob das *Minimum-Bisection-Problem* auch für planare Graphen NP-schwer ist. Allerdings wurde 1977 von Lipton & Tarjan das *Planar-Separator-Theorem* (abgekürzt PST) bewiesen [1], in dem gezeigt wurde, dass in linearer Zeit ein Separator gefunden werden kann, dessen Größe kleiner als $4 \cdot \sqrt{n}$ ist und die zerlegten Teilkomponenten maximal $2/3 \cdot n$ Knoten besitzen. Hierbei bezeichnet n die Anzahl der Knoten des Graphen.

Planar-Separator-Theorem (PST):
Die Knotenmenge eines zusammenhängenden planaren Graphen $G = (V, E)$, $n = |V| \geq 5$, kann so in linearer Zeit $\mathrm{O}(n)$ in drei Mengen $V_1,\ V_2,\ S \subseteq V$ partitioniert werden, dass

1. S Separator, der V_1 und V_2 trennt,
2. $|V_1|,\ |V_2| \leq \frac{2}{3} \cdot n$,
3. $|S| \leq 4 \cdot \sqrt{n}$.

Das *Planar-Separator-Theorem* kann man, was das algorithmische Verfahren angeht, in drei Phasen einteilen. In der ersten Phase wird versucht, den Graphen anhand eines Breitensuchbaums zu separieren. Falls diese Separation die Schranken des Theorems nicht einhält, so wird ein Teilgarph des

Gesamtgraphen, anhand von einigen Regeln ausgewählt und trianguliert. Darin wird dann ein Separator gefunden, der in der dritten Phase zu einem Separator des Gesamtgraphen, mit Hilfe des Breitensuchbaums in der ersten Phase, ergänzt wird. Man sieht also, dass einige aufspannende Bäume im Planar-Separator-Theorem benötigt werden, deren Struktur einen direkten Einfluss auf die Separation haben könnte.

1.2 Motivation

Eine Hauptmotivation für diese Arbeit war die Untersuchung des PST in [10], bei der sich der Einfluss einiger Freiheitsgrade auf die Qualität des Separators als bedeutend herausgestellt hat, sowie einfache Heuristiken, die sich ebenfalls als sehr performant erwiesen haben. Mit der Qualität eines Separators werden die Charakteristika einer Separation gemeint. Zum Beispiel die Separatorgröße und Balanciertheit der Zerlegung, anhand denen man entscheiden kann, wie geeignet eine Separation hinsichtlich ihres Anwendungsgebietes ist. Diese Ergebnisse legten es nahe, weitere Freiheitsgrade und Heuristiken zu untersuchen. Es stellt sich somit die Frage, ob nicht eine genauere Analyse der restlichen noch nicht untersuchten Freiheitsgrade auch zu einer Verbesserung der Qualität des Separators führen könnten. In dieser Arbeit werden einige dieser noch offen stehenden Freiheitsgrade, nämlich die Konstruktion aufspannender Bäume und die Tringulierung einer festen Einbetteung des Graphen, näher untersucht und ihre Auswirkungen auf die Qualität des Separators experimentell getestet.

1.3 Ziele

Hauptaugenmerk dieser Arbeit ist die Untersuchung aufspannender Bäume für das *Planar-Separator-Theorem*, sowie Triangulierungen zu diesem Zwecke. Grob gesehen kann man das PST, was die Konstruktion aufspannender Bäume betrifft, in zwei Phasen einteilen, die den ersten beiden Phasen in der Vorstellung des Theorems zugeordnet werden können. Die Intuition deutet darauf hin, dass in der ersten Phase sich ein hoher Breitensuchbaum für die Separation eignet und in der zweiten Phase ein niedriger aufspannender Baum. Eine genaue Analyse der Strukturen aufspannender Bäume könnte es uns ermöglichen, festzustellen, in welchem Maße wir in der Lage sind, geeignete Bäume im Hinblick auf die verschiedenen Phasen des Theorems zu konstruieren. Außerdem werden wir die Triangulierung wegen ihres Einflusses auf den Radius des Graphen bei der Konstruktion eines niedrigen aufspannenden Baumes einbeziehen. Aus diesen Studien wollen wir einige Heuristiken zusammenstellen und auf unterschiedlichen Graphklassen testen und feststellen, ob insgesamt eine Verbesserung erzielt werden kann.

2 Aufspannende Bäume

Einer der Hauptziele in dieser Arbeit ist die Konstruktion aufspannender Bäume für die verschiedenen Phasen des PST. Wie bereits erwähnt, deutet die Intuition darauf hin, dass in der ersten Phase des PST hohe Breitensuchbäume und in der zweiten Phase aufspannende Bäume mit einer geringen Höhe von Vorteil sein könnten. Die Untersuchungen und Experimente dieses Abschnitts gehen dieser Intuition nach. Wir werden versuchen hohe Breitensuchbäume und tiefe aufspannende Bäume zu konstruieren und uns noch anschauen, welchen Einfluss die Wurzelwahl in einem aufspannenden Baum auf die Baumhöhe hat. Insbesondere werden wir die Zentrumsknoten genauer analysieren, da aus diesen Knoten die niedrigsten Breitensuchbäume und damit auch die niedrigsten aufspannenden Bäume konstruiert werden können. Die besten Ergebnisse aus dieser Studie werden dann, nachdem sie vorher ausreichend getestet worden sind, im PST eingesetzt werden.

2.1 Grundbegriffe

Für die Untersuchung aufspannender Bäume benötigen wir einige grundlegende Definitionen. Dabei sei im Folgenden $G = (V, E)$ ein einfacher ungerichteter zusammenhängender Graph.

Ein Weg $W = (v_0, v_1, ..., v_k)$, mit $k = 0$ oder bei dem falls $l \geq 1$ alle Knoten $v_i,\ i = 0, \ldots, k$ paarweise verschieden sind, heißt Pfad. Ein Pfad aus $n \in \mathbb{N}$ Knoten hat die Länge $n - 1$ und besteht aus $n - 1$ Kanten. Sei $P = (v_0, \ldots, v_l)$ ein Pfad mit der Länge k, dann ist v_0 der Anfangsknoten und v_k der Endknoten und alle anderen Knoten heißen die inneren Knoten des Pfades. Ein u-v-Pfad ist ein Pfad mit Anfangsknoten u und Endknoten v.

Den Abstand von zwei Knoten $u, v \in V$ definieren wir als die Länge eines kürzesten u-v-Pfades in G.

$$\boxed{\text{dist}(u, v) = \min_{k \in \mathbb{N}_0} |u\text{-}v\text{-Pfad} = (u, l_1 \ldots, l_k, v)| \text{ mit } l_i \in V \text{ mit } i = 1, \ldots, k.}$$

Ein *maximaler u-v-Pfad* ist ein Pfad zwischen den Knoten $u, v \in V$, dessen Länge maximal in der Menge aller u-v-Pfade ist. Ein *maximaler Pfad* in G ist ein Pfad, dessen Länge maximal in der Menge aller existierenden Pfade in G ist.

Die *Exzentrizität* $\text{ex}(v)$ eines Knoten $v \in V$ ist der Knotenabstand des entferntesten Knotens zu v.

$$\boxed{\text{ex}(v) = \max_{u \in V} \text{dist}(v, u).}$$

Der *Durchmesser* $\text{diam}(G)$ eines einfachen ungerichteten zusammenhängenden Graphen ist als die Länge eines längsten kürzesten Pfades in G definiert.

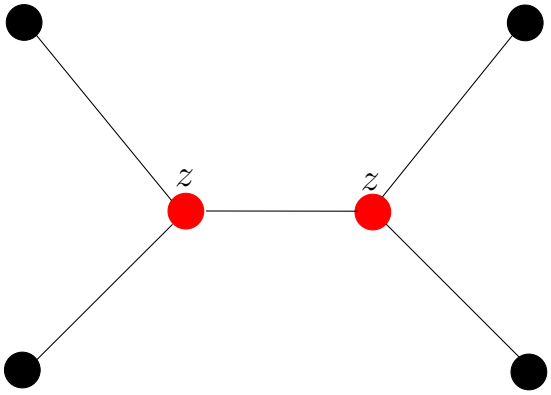
$$\boxed{\text{diam}(G) = \max_{v \in V} \text{ex}(v) = \max_{u,v \in V} \text{dist}(u, v).}$$

Der *Radius* $\text{rad}(G)$ ist definiert als das Minimum der Exzentrizität über alle Knoten des Graphen G.

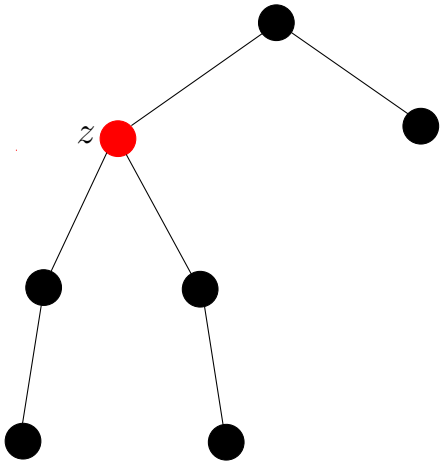
$$\boxed{\text{rad}(G) = \min_{v \in V} \text{ex}(v).}$$

Die Menge $\mathrm{Z}(G) \in V$, die alle Knoten aus V enthält, deren Exzentrizität gleich dem Radius ist, nennt man *Zentrum* des Graphen G.

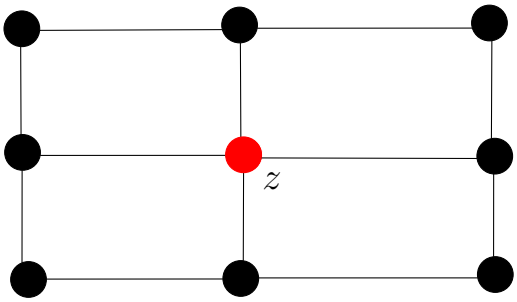
$$\mathrm{Z}(G)=\{v \in V \mid \mathrm{rad}(G) = \mathrm{ex}(v)\}.$$



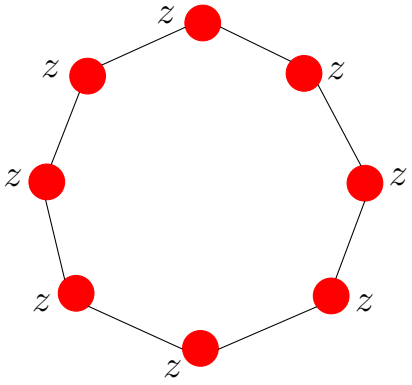
$rad(G) = 2,\ diam(G) = 3,\ \#Z(G) = 2$



$rad(G) = 2,\ diam(G) = 4,\ \#Z(G) = 1$



$rad(G) = 2,\ diam(G) = 4,\ \#Z(G) = 1$



$rad(G) = 4,\ diam(G) = 4,\ \#Z(G) = 8$

Abbildung 1: Zentren in Graphen.

Aus den obigen Abbildungen ist ersichtlich, dass die Anzahl der Zentrumsknoten von Graph zu Graph sehr stark variieren kann. Wie wir später aber noch sehen werden, kann man für Bäume dennoch eine Aussage über die Kardinalität des Zentrums Z treffen.

2.2 Zentren in Bäumen

Viele Algorithmen zur Konstruktion aufspannender Bäume wie z.B. die Breitensuche und Tiefensuche benötigen einen Knoten als Startknoten (Wurzelknoten), aus dem der Baum konstruiert wird. Es wird untersucht, welchen Einfluss die Wahl der Wurzel auf die Baumstruktur hat. Hierfür zieht man vor allem die Zentrumsknoten des Graphen in Betracht. Für diesen Zweck muss ebenfalls untersucht werden, mit welchem zeitlichen Aufwand das Finden eines Zentrumsknoten verbunden ist. Ein zu großer Zeitaufwand könnte die lineare Gesamtlaufzeit des PST beeinflussen und vielleicht die Aussage des Theorems verletzen. Es ist also sinnvoll, diese offenen Fragen etwas genauer zu behandeln.

Für allgemeine Graphen müsste man für die Zentrumsuche quadratischen Aufwand investieren. In Bäumen aber, die eine sehr spezielle Graphstruktur haben, gibt es einige interessante Feststellungen, anhand denen man einen Algorithmus mit linearem Zeitaufwand zur Zentrumssuche angeben kann. Durch das Finden eines Zentrumsknoten kann dann ganz leicht auch der Radius bestimmt werden. Der Radius ist ein wichtiges strukturelles Merkmal von Graphen, durch dessen Kenntnis man sich ein besseres Bild vom Graphen verschaffen kann. Ebenfalls von Interesse ist die Kardinalität des Zentrums. Im folgenden werden wir die Zentrumssuche in Bäumen aus einer ausführlichen theoretischen Sicht behandeln und zum Algorithmus der Zentrumssuche in Bäumen übergehen. Hierfür benötigen wir einige Hilfssätze, die später Verwendung finden.

Lemma 2.1 *Zwei maximale Pfade in einem Baum haben mindestens einen gemeinsamen Knoten.*

Beweis: Wir nehmen an, es existieren zwei maximale Pfade $(v, \ldots, w)$ und $(x, \ldots, y)$ in einem Baum, deren Knotenschnittmenge leer ist. Wähle nun einen Knoten u auf dem Pfad $(v, \ldots, w)$ bzw. auf dem Pfad $(x, \ldots, y)$, dessen Abstand in diesem Pfad zum Knoten w bzw. y größer gleich der Hälfte der Pfadlänge ist. Da ein Baum insbesondere ein zusammenhägender Graph ist, existiert ein kürzester Pfad $(u, \ldots, z)$ zwischen den Knoten u und z. Wenn wir nun aber den Pfad $(w, \ldots, u, \ldots, z, \ldots, y)$ betrachten, ist dessen Länge größer als die Länge eines maximalen Pfades und daraus folgt, dass der $(w, \ldots, v)$ Pfad und der $(x, \ldots, y)$ Pfad nicht maximal gewesen sind, was der anfänglichen Annahme widerspricht. □

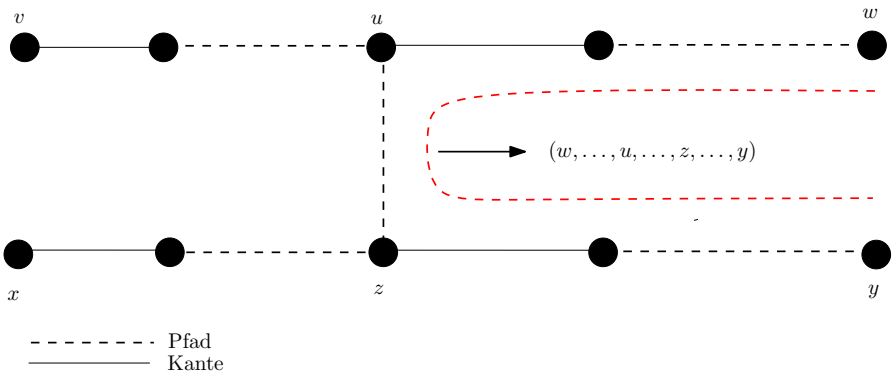


Abbildung 2: Zwei maximale Pfade in einem Baum haben mindestens einen gemeinsamen Knoten.

Jeden Knoten u eines $(v, \ldots, w)$ Pfades, dessen Abstand zu einen der beiden Endknoten v oder w des Pfades gleich $\lfloor L/2 \rfloor$ ist, bezeichnen wir als *Mittelknoten* des Pfades. Je nachdem ob ein Pfad eine gerade oder ungerade Länge hat, besitzt er zwei oder einen Mittelknoten.

Lemma 2.2 *Die Mittelknoten zweier maximaler Pfade eines Baumes sind identisch.*

Beweis: Seien der $(v, \ldots, w)$ Pfad und $(x, \ldots, y)$ Pfad zwei maximale Pfade eines Baumes mit der Knotenschnittmenge $\{u_1, \ldots, u_k\}$ mit k aus $\mathbb{N}$. Aus dem vorherigen Satz wissen wir, dass die Knotenschnittmenge nicht leer ist. Nun gilt ohne Beschränkung der Allgemeinheit:

$$\begin{aligned} \text{dist}(v, w) &= \text{dist}(x, y) && (1) \\ \text{dist}(w, u_k) &\geq \text{dist}(v, u_1) && (2) \\ \text{dist}(y, u_k) &\geq \text{dist}(x, u_1) && (3) \end{aligned}$$

Unmittelbar aus dieser Feststellung kann man folgern, dass der $(w, \ldots, u_k)$ Pfad und der $(y, \ldots, u_k)$ Pfad die gleiche Länge haben. Entsprechendes gilt auch für den $(v, \ldots, u_1)$ Pfad und $(x, \ldots, u_1)$ Pfad. Denn wäre dies nicht der Fall und wäre ohne Einschränkung der $(w, \ldots, u_k)$ Pfad echt länger als der $(y, \ldots, u_k)$ Pfad, so müsste der $(x, \ldots, u_1)$ Pfad echt länger als der $(v, \ldots, u_1)$ Pfad sein, da ansonsten die beiden maximalen Pfade nicht die gleiche Länge hätten. Nun würde aber der $(w, \ldots, u_k, \ldots, u_1, \ldots, x)$ Pfad ein noch längerer Pfad sein als der maximale Pfad des Baumes, was ein Widerspruch zu der Annahme wäre, dass die beiden Pfade $(w, \ldots, v)$ und $(y, \ldots, x)$ maximale Pfade sind.

Somit gilt:

$$\text{dist}(w, u_k) = \text{dist}(y, u_k) \quad \text{und} \quad \text{dist}(v, u_1) = \text{dist}(x, u_1).$$

Außerdem gilt:

$$\text{dist}(w, u_k) = \text{dist}(y, u_k) \leq \lfloor n/2 \rfloor .$$

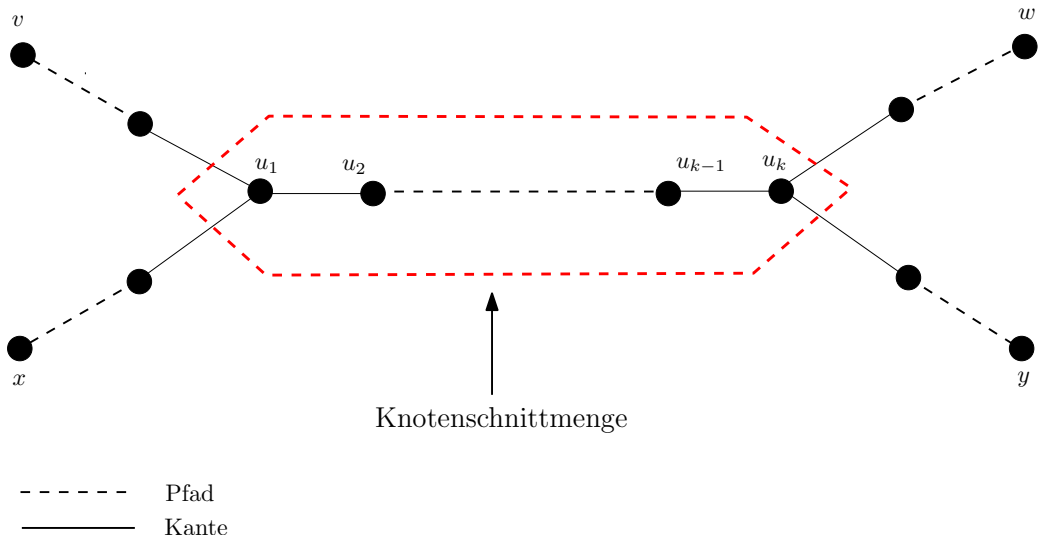


Abbildung 3: Maximale Pfade und die Knotenschnittmenge.

Mit Hilfe der Ungleichungen (2) und (3) ist nun offensichtlich, dass die Mittelknoten der beiden maximalen Pfade in der Knotenschnittmenge $\{u_1, \ldots, u_k\}$ enthalten sind und miteinander übereinstimmen. □

Satz 2.1 *Für die Anzahl der Zentrumsknoten in einem Baum $G = (V, E)$ gilt:*

$$\boxed{1 \leq \#Z(G) \leq 2.}$$

Beweis:

Es reicht zu zeigen, dass die Mittelknoten eines maximalen Pfades die minimalste Exzentrizität über alle Knoten des Baumes haben. Sei z_1 ein Mittelknoten eines maximalen Pfades $(v \ldots w)$ im Baum und der Knoten z_2, je nachdem ob zwei Mittelknoten existieren oder nicht, ein Mittelknoten oder ein normaler Pfadknoten. Die Länge des maximalen Pfades im Graphen G bezeichnen wir mit L.

Es gilt für alle Knoten u aus V:

$$\text{dist}(z_1, u) \leq \lceil L/2 \rceil \, .$$

Speziell für die Endknoten des maximalen v-w-Pfades ergibt sich somit folgendes:

$$\begin{array}{ll} & [(\text{dist}(z_1, v) = \lfloor L/2 \rfloor) \wedge (\text{dist}(z_1, w) = \lceil L/2 \rceil)] \\ \vee & [(\text{dist}(z_1, v) = \lceil L/2 \rceil) \wedge (\text{dist}(z_1, w) = \lfloor L/2 \rfloor)] \, . \end{array}$$

Für jeden Knoten q auf dem v-w-Pfad, der kein Mittelknoten ist, ist der Abstand zu einem der Endknoten v oder w größer als der Abstand des Mittelknotens z_1 zu einem Endknoten.

$$\text{dist}(q, v) > \text{dist}(z_1, v) \quad \text{oder} \quad \text{dist}(q, w) > \text{dist}(z_1, w)$$

Sei nun u ein Knoten der nicht auf diesem maximalen Pfad enthalten ist. Dann gibt es immer einen Knoten k auf dem v-w-Pfad, so dass:

$$\text{dist}(u, v) \geq (\text{dist}(k, v) + 1) \quad \text{oder} \quad \text{dist}(u, w) \geq \text{dist}(k, w) + 1.$$

Die folgende Abbildung veranschaulicht diesen Sachverhalt, wobei der Knoten u einem Knoten aus der Menge $\{u_1,\ u2,\ u_3,\ u_4\}$ entsprechen kann und der Knoten k einem der Knoten aus der Menge $\{k_1,\ z_1,\ z_2,\ k_2\}$.

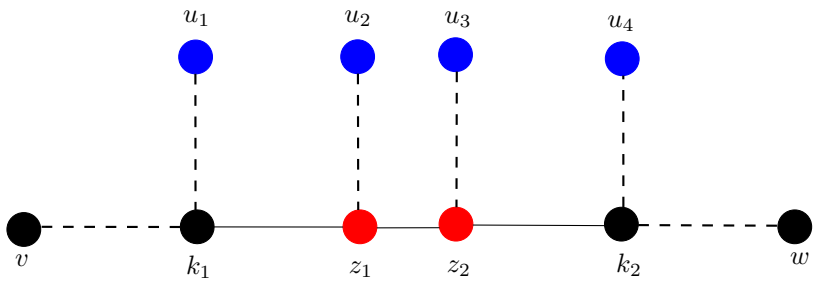


Abbildung 4: Maximale Pfade in Bäumen

Da bereits bewiesen wurde, dass die Mittelknoten maximaler Pfade identisch sind, folgt somit, dass das Zentrum eines Baumes genau den Mittelknoten eines maximalen Pfades entspricht. Ein maximaler Pfad hat mindestens einen Mittelknoten und höchstens zwei Mittelknoten. Daraus ergibt sich die Behauptung über die Kardinalität des Zentrums von Bäumen. □

Mit Hilfe der bewiesenen Sätze kann man nun einen Algorithmus angeben, mit dem in linearer Zeit $O(|E|)$ die Zentren eines Baumes bestimmt werden können. Es wird hier von den Implementierungsdetails abgesehen und nur die Vorgehensweise des Algorithmus geschildert. Im Folgenden bezeichnen wir einen Knoten mit Grad 1 als Blattknoten. Dabei entspricht der Grad eines Knotens der Anzahl nicht markierter Nachbarknoten.

Für den folgenden Algorithmus setzen wir voraus, dass ein aufspannender Baum gegeben ist. Desweiteren seien Q_1 und Q_2 zwei Warteschlangen, die die Funktionen pop(), append(v) und clear(v) unterstützen. Die Funktion pop() liefert das erste Element in der Warteschlange zurück und append(v) hängt das Element v an das Ende der Warteschlange. Die Funktion clear() löscht alle Elemente der Warteschlange.

Die Idee hinter dem Algorithmus ist die Tatsache, dass ein Mittelknoten auf einem maximalen Pfad als letzter Knoten den Grad 1 annimmt und bis zum Schluß einen Mindestgrad von 2 hat.

In Abbildung 5 wird der Ablauf des Algorithmus verdeutlicht. Die maximalen Pfade des Baumes sind hierbei die gestrichelt eingezeichneten Pfade. Die Blattknoten werden in jedem Schritt rot markiert bis zwei unmarkierte Knoten übrig bleiben. Diese unmarkierten Knoten sind genau die Zentrumsknoten des Baumes.

Algorithmus 1 : Zentrumsuche in Bäumen

Eingabe : Aufspannender Baum B
Ausgabe : Zentrum $Z(B)$ von B

```
1  Z(B) ← Alle Knoten aus B
2  Für alle Knoten v aus B
3      Grad[v] ← Anzahl nicht markierter Nachbarknoten
4  Für alle Knoten v aus B
5      Wenn Grad[v] == 1
6          Q_1.append(v)
7          mark(v)
8          Z(B).delete(v)
9  Solange #Z(B) > 2 wiederhole
10     Für alle Nachbarknoten v von Knoten aus Q_1
11         Grad[v]= Anzahl nicht markierter Nachbarknoten
12         Wenn Grad[v] == 1
13             Q_2.append(v)
14             mark(v)
15             Z(B).delete(v)
16     Q_1 ← Q_2
17     Q_2.clear()
18 return Z(B)
```

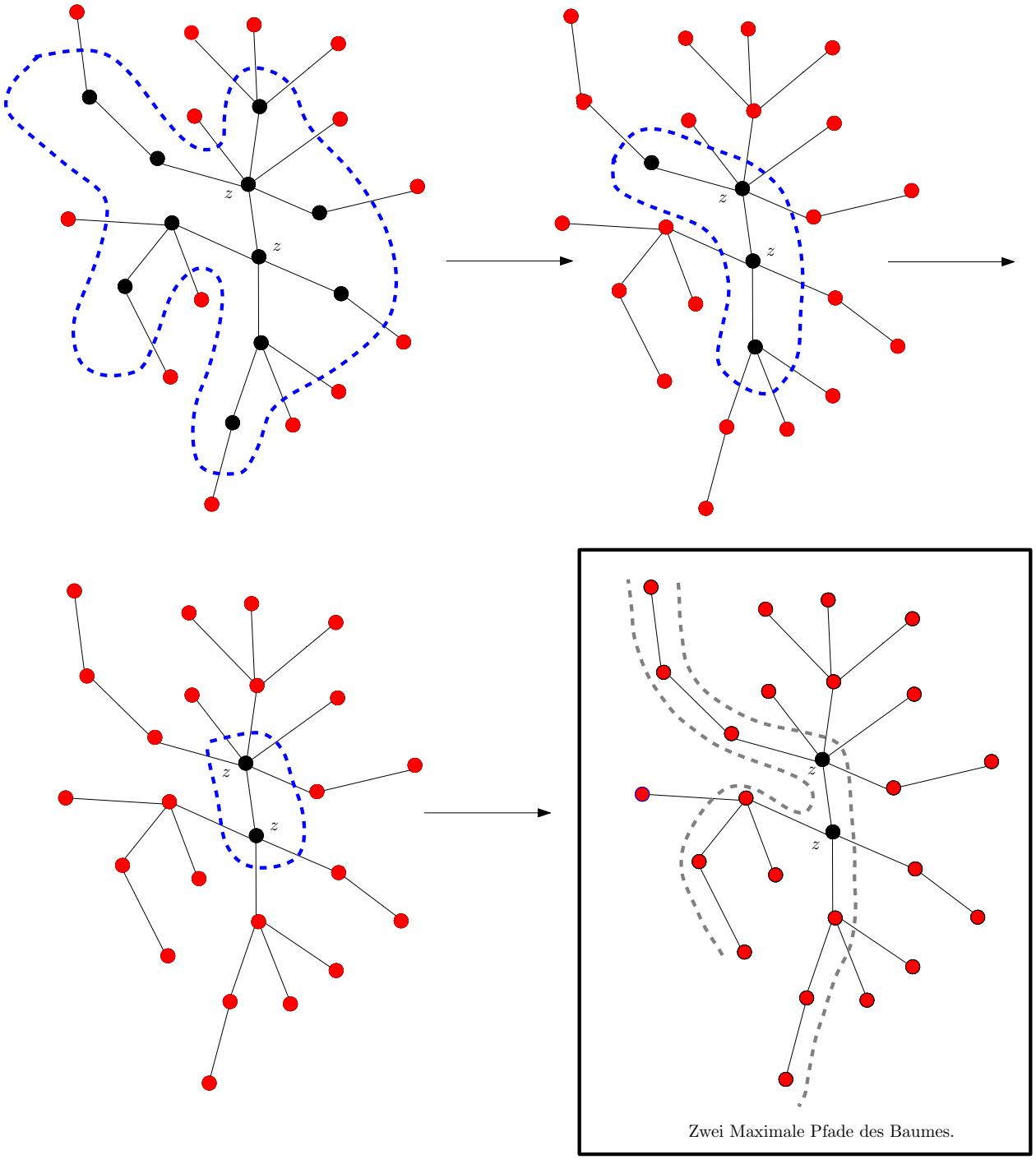


Abbildung 5: Anwendungsbeispiel zur Zentrumssuche in Bäumen.

2.3 Breitensuchbaum

Die Definition eines Breitensuchbaums wird direkt aus dem Konstruktionsalgorithmus dieses Baumes abgeleitet. Dennoch definieren wir den Breitensuchbaum etwas allgemeiner, um eine größere Klasse von Bäumen aufgrund ihrer besonderen Eigenschaft, die äquivalent zu einem Breitensuchbaum sind, als Breitensuchbaum bezeichnen zu können.

Definition 2.1 *Ein Breitensuchbaum im Graphen $G = (V, E)$ ist ein Baum, in dem der Abstand jeden Knotens zur Baumwurzel der Länge des kürzesten Pfades zwischen diesen beiden Knoten entspricht.*

Breitensuchbäume erfüllen einige besondere Eigenschaften, die ein beliebig konstruierter aufspannender Baum im allgemeinen nicht erfüllt. Wir werden einige dieser Eigenschaften erläutern, die im Beweis des *Planar-Separator-Theorem* [1] benutzt werden.

2.3.1 Level-Einteilung

Sei $H_B = (V, E_B)$ ein Breitensuchbaum im Graphen $G = (V, E)$ mit der Höhe h und dem Knoten v als die Wurzel des Breitensuchbaums. Dann kann man die Knoten des Graphen anhand ihrem Abstand zur Wurzel in Mengen Level(i) $i = 0, \ldots, h$ einteilen, wobei ein Knoten, dessen Abstand zur Wurzel i beträgt, im Level(i) enthalten ist. Somit sind die Mengen Level(i) und Level(j) für i verschieden von j disjunkt und die Vereinigung aller Level ergibt V.

- Level$(i) \cap$ Level$(j) = \emptyset$
- $\bigcup_{i=0}^{h}$ Level$(i) = V$

Oft bezeichnen wir Level(i) auch als das i-te Level. Es is offensichtlich, dass jeder andere Breitensuchbaum mit der selben Wurzel, exakt die gleiche Level-Einteilung liefert, da der Abstand der Knoten zur Wurzel unverändert bleibt. Diese Einteilung ist also nur von der Wahl des Wurzelknotens abhängig.

2.3.2 Separationseigenschaft

Gegeben sei ein Breitensuchbaum $B = (V, E_B)$ im Graphen $G = (V, E)$. Zwischen zwei Knoten, die im Breitensuchbaum B einen Levelunterschied größer gleich zwei haben, existiert keine Kante im Graphen G, da ansonsten ein echt kürzerer Pfad von einem Knoten zur Wurzel existieren würde als der Pfad zur Wurzel, der im Breitensuchbaum durchquert wird, was im Widerspruch zur Definition des Breitensuchbaums steht.

Für alle $v, w \in G$ $v \in$ Level(i) und $w \in$ Level(j) mit $\lvert i - j \rvert \geq 2$ gilt : $\{v, w\} \notin E$.

2.3.3 Baumhöhe

Aus den Definitionen für Radius und Durchmesser eines Graphen G des vorherigen Abschnitts kann man ableiten, dass diese Parameter stets die folgende Ungleichung erfüllen.

$$\boxed{\text{rad}(G) \leq \text{diam}(G) \leq 2 \cdot \text{rad}(G).}$$

Die Ungleichung $\text{rad}(G) \leq \text{diam}(G)$ ist direkt aus der Definition ableitbar. Die zweite Ungleichung kann man damit begründen, dass von einem beliebigen Knoten aus, ein Pfad zu jedem Zentrumsknoten des Graphen existiert, dessen Länge maximal gleich dem Radius ist. Also hat jeder kürzeste Pfad zwischen zwei Knoten maximal eine Länge gleich $2 \cdot \text{rad}(G)$. Da auch ein längster kürzester Pfad diese Gleichung erfüllt, folgt hieraus die Behauptung.

Nun sei $B = (V, E_B)$ ein Breitensuchbaum in G mit Wurzelknoten v und der Höhe H_B. Der Abstand von v zu jedem anderen Knoten des Graphen ist, nach den Definitionen von Radius und Durchmesser, größer gleich dem Radius und kleiner gleich dem Durchmesser. Damit gilt folgendes für die Baumhöhe H_B.

$$\boxed{\text{rad}(G) \leq H_B \leq \text{diam}(G).}$$

Diese Ungleichung hebt die Tatsache hervor, dass die Höhe der Breitensuchbäume durch den Radius und Durchmesser des Graphen beschränkt werden. Dabei wird der Radius oder der Durchmesser als Höhe durchaus auch angenommen. Man kann daher in jedem Graphen Breitensuchbäume konstruieren oder suchen, die den Radius oder den Durchmesser als Höhe haben. Die Wahl der Wurzel ist hierbei entscheidend. Würde man zum Beispiel aus einem Zentrumsknoten des Graphen einen Breitensuchbaum konstruieren, so hätte dieser eine Höhe gleich dem Radius.

2.3.4 Konstruktion

Als nächstes wollen wir uns den Konstruktionsalgorithmen von Breitensuchbäumen zuwenden. Die allgemeine Variante verwendet als Datenstruktur eine Warteschlange Q, in der die Knoten nach ihrer Abarbeitungsreihenfolge gespeichert werden. Diese Datenstruktur unterstützt die Operationen *append*(x) und *pop*(x). Bei *append*(x), wird das Element x hinten an die Q angehängt, und mit *pop*$()$ bekommt man das erste Element von Q zurück.

Analyse: Sei $G = (V, E)$ ein zusammenhängender Graph. Jede Kante des Graphen G wird genau einmal von jeden seiner Endknoten betrachtet. Daher ist die Worst-Case-Laufzeit T^{worst} aus $O(|E|)$ für zusammenhängende Graphen. Der Speicherplatzbedarf S^{worst} ist ebenfalls aus $O(|E|)$ für zusammenhängende Graphen, da der dünnste zusammenhängende Graph ein Baum ist und genau $|V| - 1$ Kanten hat. Jeder zusammmenhängende Graph, der kein Baum ist, hat mindestens $|V|$ Kanten.

Wenn wir uns Schritt 2 im Algorithmus 2 anschauen, stellen wir fest, dass nicht festgelegt ist, in welcher Reihenfolge die ausgehenden Kanten aus einem Knoten betrachtet werden. Diese Freiheit kann man sich zu nutzen machen, um verschiedene Breitensuchbäume zu konstruieren. Eine unterschiedliche Reihenfolge der Abarbeitung der ausgehenden Kanten verursacht insbesondere eine andere Platzierung der Knoten in der Warteschlange Q. Man kann nicht nur die ausgehenden Kanten

Algorithmus 2 : Breitensuchbaum

Eingabe : Zusammenhängender Graph $G = (V, E)$ mit der Knotenmenge V und der Kantenmenge E und einem Knoten $w \in V$ als Wurzel des Breitensuchbaums

Ausgabe : Breitensuchbaum $B = (V, E_B)$ mit $E_B \subseteq E$ mit Wurzel

```
1  Q.append(w)
2  Solange Q nicht leer ist wiederhole
3      v ← Q.pop()
4      Für alle ausgehenden Kanten e von v
5          u ← Inzidenter Knoten zu e ungleich v.
6          Wenn u nicht markiert ist
7              markiere u.
8              Q.append(u)
9              E_B = E_B ∪ {e}
10 return E_B
```

eines bestimmten Knotens, sondern alle Kanten, die zwei aufeinander folgende Levelknoten miteinander verbinden, betrachten. Unser Ziel ist es, diese Freiheitsgrade zu untersuchen und festzustellen, welchen Einfluss es auf die Baumstruktur haben kann.

Standardbreitensuche: Bei dieser Art der Breitensuche werden die Kanten eines Knotens in der Reihenfolge, in der sie intern in der Datenstruktur des Graphen gespeichert sind, durchlaufen. Falls wir zum Beispiel die Koordinaten der Knoten eines in der Ebene eingebetteten Graphen betrachten, können wir über die ausgehenden Kanten der Knoten im Uhrzeigersinn oder Gegenuhrzeigersinn beginnend bei 12: 00 Uhr iterieren. Diese Art der Breitensuche liefert für einen bestimmten Wurzelknoten immer exakt den gleichen Breitensuchbaum, falls die Koordinaten der Knoten und damit ihre Anordnung sich nicht ändert.

Ordnungsbehaftete Breitensuche: Um eine andere Abarbeitungsreihenfolge der Knoten eines bestimmten Levels zu bestimmen, kann für diese Knoten eine partielle oder totale Ordnugsrelation definiert werden. Anhand dieser Relation sortiert man die Knoten des Levels und bearbeitet sie wie im Algorithmus 2 beschrieben. Um nicht eine zu hohe Laufzeit zu verursachen, sollte man die Ordnungsrelation geschickt wählen, so dass das Sortieren effizienter (in linearer Laufzeit) möglich wird. Der Knotengrad z.B. eignet sich besonders gut für diesen Ansatz.

Permutierende Breitensuche: Oft ist es nicht wünschenswert, von einem bestimmten Knoten als Wurzel immer exakt den gleichen Breitensuchbaum zu konstruieren. Es gibt Fälle, wo man eher einen zufälligen Baum haben möchte, statt immer den gleichen. Hierfür bietet sich die Möglichkeit an, die Knoten eines Levels zu permutieren und dann abzuarbeiten. Die gleiche Idee kann für alle Kanten zweier aufeinanderfolgender Levels verwendet werden. Es ist erwähnenswert, dass mit dieser Art der Breitensuche alle möglichen Breitensuchbäume aus einem bestimmten Wurzelknoten konstruiert werden können.

Balancierte Breitensuche: Bei der balancierten Breitensuche wird versucht, einen Breitensuchbaum zu konstruieren, der so wenig wie möglich Blattknoten hat. Hierbei werden die Knoten eines Levels aufsteigend nach ihrem Knotengrad in die Warteschlange Q eingefügt. Dann wird eine Kante $\{v, u\}$ vom ersten Knoten v in Q zu einem noch nicht markierten Knoten u gewählt. Falls eine solche Kante existiert, so wird die Kante zum Breitensuchbaum hinzugenommen und der Knoten v kommt ans

Ende von Q. Falls eine solche Kante nicht existiert, so wird v aus Q entfernt. Analog wendet man auf die Knoten des neu hinzugekommenen Levels das selbe Verfahren an, bis man schließlich einen Breitensuchbaum hat. Zusammengefasst erlauben wir jedem Knoten eines Levels genau eine Kante zu einem Knoten des darauf folgenden Levels zu ziehen und machen analog weiter mit den nächsten Knoten. Die Sortierung nach dem Knotengrad erhöht hierbei die Wahrscheinlichkeit, dass Kanten von Knoten mit niedrigem Grad zu Knoten des darauf folgenden Levels gewählt werden. Dadurch wird verhindert, dass diese Knoten frühzeitig zu Blattknoten werden.

Um dem Leser zu demonstrieren, wie unterschiedliche Breitensuchbäume von dem gleichen Wurzelknoten ausgehend konstruiert werden können, wenden wir die verschiedenen Varianten der Breitensuche auf einen Beispielgraphen an. Gegeben sei ein Graph, der in der folgenden Abbildungen 6 zu sehen ist. Die zum Breitensuchbaum gehörenden Kanten sind jeweils dicker und rot eingezeichnet. Die Blattknoten sind jeweils mit einem b markiert.

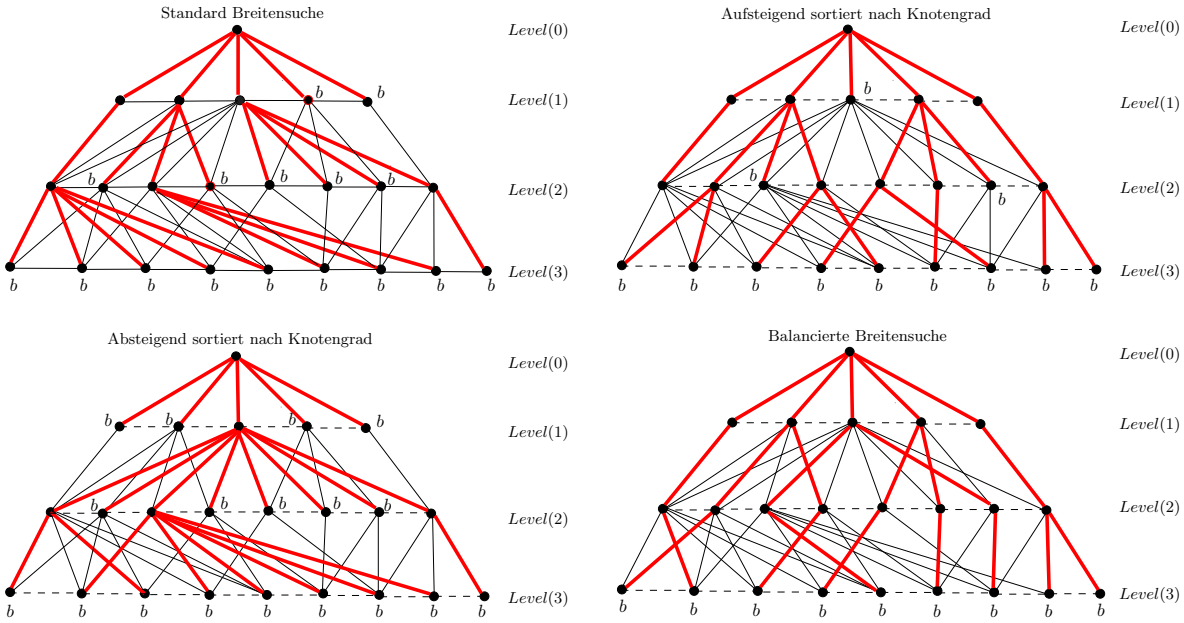


Abbildung 6: Oben (links): S tandard Breitensuche im Gegenuhrzeigersinn. Oben (rechts): Ordnungsbehafteter Breitensuchbaum, in dem die Knoten aufsteigend nach ihrem Knotengrad behandelt werden. Unten (links): Ordnungsbehafteter Breitensuchbaum, in dem die Knoten absteigend nach ihrem Knotengrad behandelt werden. Unten (rechts): Balancierter Breitensuchbaum.

Es ist deutlich zu erkennen, dass die Anzahl der Blattknoten der konstruierten Bäume sich stark unterscheiden. Vor allem können Knoten in einem Baum Blattknoten sein und in einem anderen keine. Wenn wir die balancierte Breitensuche betrachten, so sind in diesem Beispiel nur die Knoten im höchsten Level Blattknoten und der Baum ist ausbalanciert. Bei allen anderen Varianten der Breitensuche sind Knoten im ersten oder zweiten Level frühzeitig zu Blattknoten geworden. Was die ordnungsbehaftete Breitensuche betrifft, so können Knoten mit einem hohen oder niedrigen Grad als letzte Knoten in einem Level behandelt werden und genau diese Knoten könnten frühzeitig zu Blattknoten werden.

2.4 Heuristiken

In den verschiedenen Phasen des *Planar-Separator-Theorems* werden zwei Sorten von Bäumen gebraucht. In der ersten Phase wird die Separationseigenschaft des Breitensuchbaums verwendet. Unsere Intuition deutet darauf hin, dass durch einen hohen Breitensuchbaum eventuell die Levels spärlicher besetzt sein könnten, was uns eine leichtere Separation ermöglichen würde und zusätzlich noch Auswirkungen auf die Balance haben könnte. Mehr dazu im Abschnitt 2.4.2. In der zweiten Phase wird ein aufspannender Baum benötigt, durch dessen Höhe h eine obere Schranke $(2h + 1)$ für die Separatorgröße angegeben wird (Siehe 4.1). Je niedriger der Baum also in dieser Phase konstruiert wird, desto kleiner wird auch die obere Schranke für die Separatorgröße. Ziel dieses Abschnittes ist es, Heuristiken anzugeben und zu untersuchen, mit denen man einen relativ hohen bzw. niedrigen Breitensuchbaum, in einem einfachen zusammenhängenden Graphen finden kann, ohne dabei quadratische Laufzeit spendieren zu müssen. Sowohl bei der Höhenminimierung als auch bei der Höhenmaximierung will man eine quadratische Laufzeit vermeiden, um nicht die lineare Gesamtlaufzeit des PST zu verletzen.

2.4.1 Höhenminimierung

Problem:
Gegeben: Ein einfacher zusammenhängender ungerichteter Graph $G = (V, E)$.
Gesucht: Ein niedriger Breitensuchbaum.

Wir wissen, dass der Breitensuchbaum mit der geringsten Höhe, eine Höhe gleich dem Radius des Graphen hat. Optimal wäre es, den Knoten zu finden, der als Wurzel des Baumes betrachtet, genau einen Baum liefert, dessen Höhe $\text{rad}(G)$ ist. Man könnte über alle Knoten von G iterieren und jedesmal einen Breitensuchbaum bilden und dessen Höhe abspeichern. Der Knoten, der als Wurzel eines Breitensuchbaums die geringste Baumhöhe hervorgerufen hat, ist somit der gesuchte Knoten.

Laufzeitanalyse Die Laufzeit T^{worst} ist hierbei in $\text{O}(|V| \cdot |E|)$. Da die Anzahl der Kanten $|E|$ in $\text{O}(|V|^2)$ ist, bekämen wir bei diesem Ansatz eine Laufzeit $T^{\text{worst}} \in \text{O}(|V|^3)$. Speziell für planare Graphen jedoch, wäre $|E|$ nach dem Satz von Euler maximal $3|E|-6$ groß, was eine Laufzeit $T^{\text{worst}} \in \text{O}(|V|^2)$ ergibt.

Da das *Planar-Separator-Theorem* eine lineare Gesamtlaufzeit hat, möchten man Verfahren und Heuristiken entwickeln, für die nur lineare Zeit gebraucht wird. Für größere Graphen sind quadratische Laufzeiten viel zu groß und können daher in praktischen Anwendungen nicht verwendet werden. Wir stellen nun eine Heuristik vor, mit der in vielen Fällen, falls der Graph planar ist, in linearer Zeit ein Knoten, der sich als Wurzel eines Breitensuchbaums eignet, gefunden werden kann. Die Eignung bezieht sich hier auf die resultierende Breitensuchhöhe aus einem gegebenen Wurzelknoten. Je geringer die Höhe ist, desto geeigneter bezeichnen wir den Wurzelknoten. Wie schon erwähnt, soll diese Heuristik in der zweiten Phase des *Planar-Separator-Theorem* Verwendung finden, in der die Separatorgröße anhand der Baumhöhe nach oben abgeschätzt wird.

Algorithmus 3 : Heuristik zur Höhenminimierung

Eingabe : Ein zusammenhängender Graph $G = (V, E)$ mit einem Startknoten w und Parameter i.

Ausgabe : Geeigneter Wurzelknoten für die Konstruktion eines niedrigen Breitensuchbaums.

1 Bilde einen Breitensuchbaum $B_w = (V, E_w)$ mit w als Wurzel. Suche einen Zentrumsknoten z in B_w.
2 Bilde nun einen neuen Breitensuchbaum $B_z = (V, E_z)$ über G mit z als Wurzel.
3 Setze w auf z und führe das Verfahren i-mal analog aus.
4 Gib w zurück.

Lemma 2.3 *Die Behauptung ist, dass der so gewonnene Breitensuchbaum höchstens so hoch wie der anfängliche Breitensuchbaum B_w ist.*

Beweis:
Wenn man einen aufspannenden Baum hat und eine Wurzel bestimmen will, aus dem die Höhe des aufspannenden Baumes minimal wird, so wählt man einen Knoten aus dem Zentrum des Baumes, da dieser die minimale Exzentrizität über alle Knoten des Grapen hat. Die Höhe des Baumes ist dann gleich dem Radius des Baumes. Somit ist die Höhe des Baumes B_w, wenn man ihn als Wurzelbaum mit dem Knoten z als Wurzel betrachtet, höchstens so groß wie die Höhe von B_w mit w als Wurzel. Sei $T = (V, E_T)$ ein beliebiger aufspannender Baum im Graphen $G = (V, E)$, der z als Wurzel hat. Dieser Baum ist mindestens so hoch wie ein aufspannender Breitensuchbaum mit z als Wurzel. Also hat jeder Breitensuchbaum mit z als Wurzel eine kleinere Höhe als die Höhe von T. Daraus kann man schließen:

$$\text{Höhe}(B_v, v) \geq \text{Höhe}(B_v, z) \geq \text{Höhe}(B_z, z).$$

Also tritt insgesamt durch die Anwendung der Heuristik keine Verschlechterung ein und die Höhe könnte eventuell verbessert werden.

Genauere Analyse Die Verbesserung ist stark von der Struktur des Breitensuchbaums (B_w, w) abhängig. Falls w nämlich bereits im Zentrum von B_w enthalten ist, so kann nach den Schritten 1, 2 und 3 keine Verbesserung erzielt werden. Das heißt aber nicht, dass kein anderer Breitensuchbaum existiert, in dem w nicht im Zentrum des Baumes liegt. Um festzstellen zu können, ob ein solcher Breitensuchbaum existiert oder nicht, nutzen wir die Eigenschaft der Leveleinteilung von Breitensuchbäumen aus. Im Folgenden betrachten wir einen Breitensuchbaum mit der Höhe h und der Wurzel w.

Satz 2.2 *Eine Verbesserung kann genau dann erzielt werden, wenn ein Breitensuchbaum mit der Wurzel w existiert, so dass der Unterbaum eines Nachbarknotens v von w im Breitensuchbaum bereits alle Knoten im Level h und $h-1$ abgedeckt hat.*

Beweis:

$(\Rightarrow)$:
Falls eine Verbesserung erzielt werden kann, so existiert ein Breitensuchbaum B, so dass $w \notin Z(B)$.

Wir wählen einen Knoten $z \in Z(B)$ und verfolgen seinen Pfad zur Wurzel in B, bis wir zu einem Nachbarknoten v von w gelangen. Der Knoten v existiert, da ansonsten kein Pfad zwischen z und w existieren würde. Der Unterbaum von v enthält z und alle Knoten im Level h und $h-1$. Denn wenn ein Knoten u im Level h oder $h-1$ existieren würde, das nicht im Unterbaum von w liegt, so wäre $\text{dist}(z,u) \geq h$. Damit wäre aber w auch im Zentrum, da es mindestens die gleiche Exzentrizität wie ein Knoten aus dem Zentrum hätte. Es gilt aber $w \notin Z(B)$. Somit enthält der Unterbaum vom Knoten w bereits alle Knoten aus den Leveln h und $h-1$.

$(\Leftarrow)$:
Falls der Unterbaum von einem Nachbarkonten v von w in einem Breitensuchbaum B, der w als Wurzel hat, alle Knoten aus den Leveln h und $h-1$ enthält, so hat der Knoten v in B die Exzentrizität $h-1$ und liefert somit einen niedrigeren Baum. Daher liegt w nicht in $Z(B)$ und es existiert garantiert ein Knoten $z \in Z(B)$, von dem aus ein noch besserer Breitensuchbaum konstruiert werden kann. Insgesamt ist also eine Höhenverbesserung durch die vorgestellte Heuristik möglich.

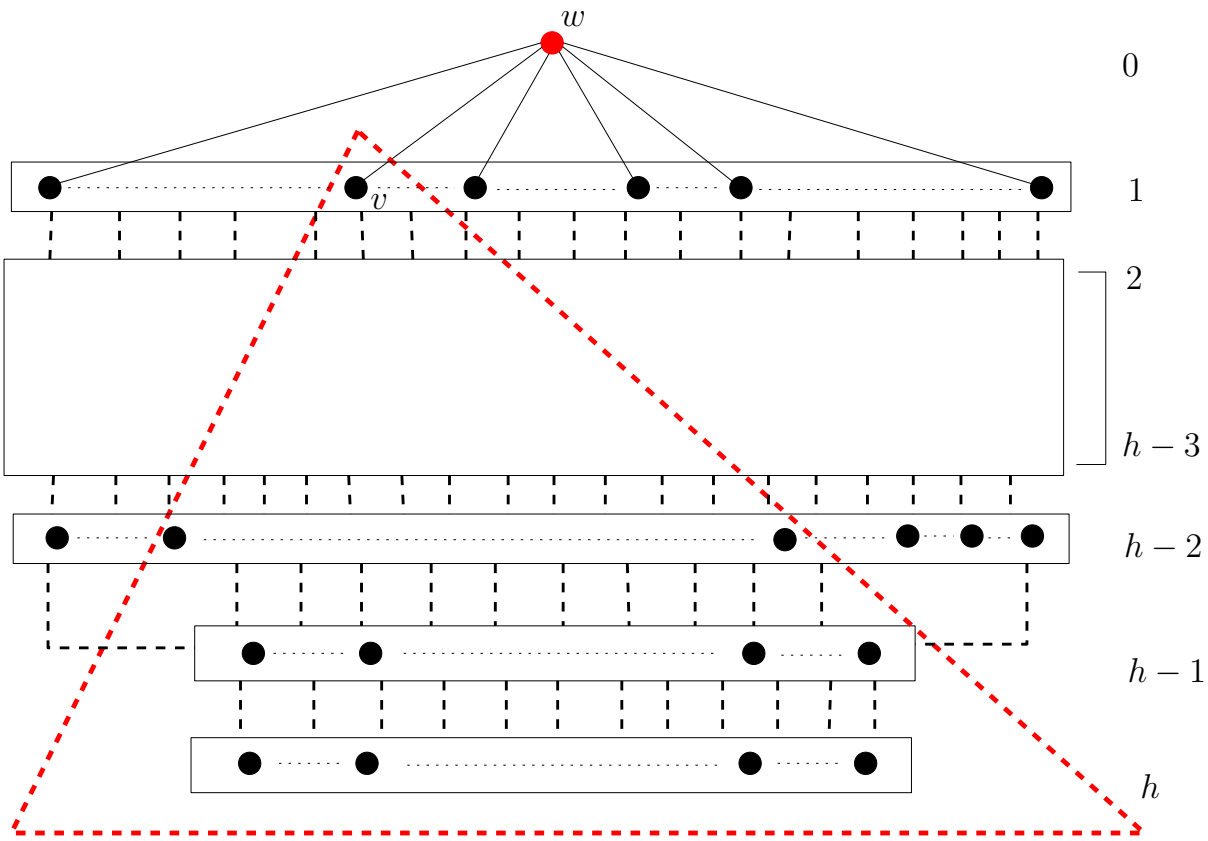


Abbildung 7: Fall indem eine Verbesserung der Höhe erzielt werden kann.

Zur Demonstration wenden wir die Heuristik auf einem Gittergraphen mit 36 Knoten an. Als Breitensuche wird dabei in diesem Beispiel die permutierende Breitensuche verwendet. Hierbei wird davon ausgegangen, dass man als Anfangsknoten einen ungünstign Knoten gwählt hat. Mit ungünstig meinen wir hier, dass ein Breitensuchbaum mit diesem Knoten als Wurzel sehr hoch wäre.

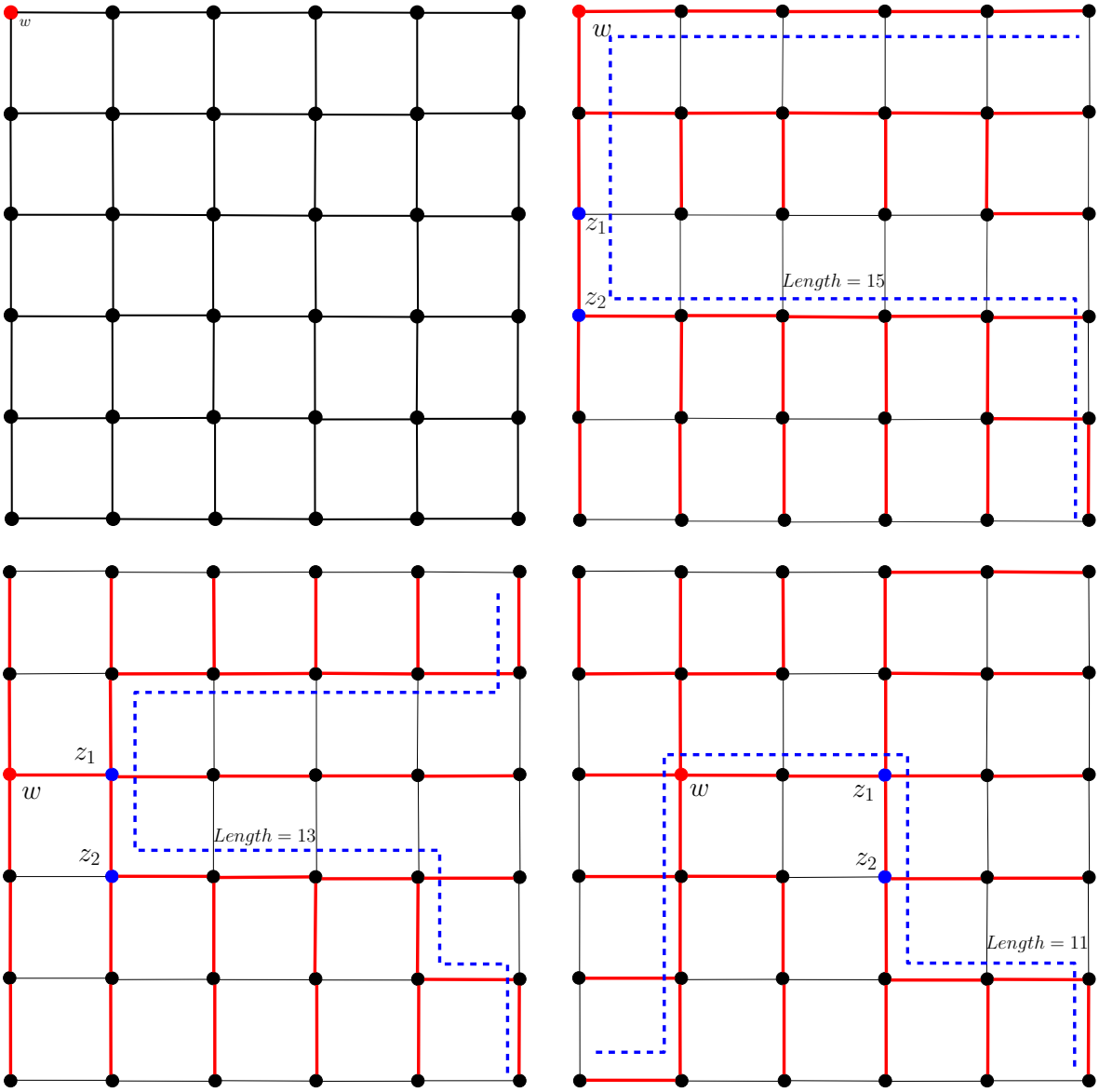


Abbildung 8: Demonstration der Höhenminimierungsheuristik anhand einem Gittergraphen mit 36 Knoten. Der Startknoten ist hierbei ein Knoten auf der äußeren Facette mit Grad 2.

Im obigen Anwendungsbeispiel sieht man ganz deutlich, wie sich die Länge eines maximalen Pfades im Baum verringert. Die Länge eines solchen Pfades reduziert sich von 15 auf 11 und die Baumhöhe reduziert sich von anfänglich 9 auf 6. Man kann also eine Korrespondenz zwischen der Baumhöhe und die Länge eines maximalen Pfades im Baum beobachten. Man sieht auch, dass nach vier Durchläufen ein Zentrumsknoten im Baum gefunden wird, der auch Zentrum des Graphen ist. Aus diesem Knoten läßt sich dann ein Breitsuchbaum mit der geringsten Höhe konstruieren. Eine derart gute Verbesserung ist, wie wir später noch sehen werden, nicht für alle Graphen möglich. Außerdem werden wir noch sehen, dass die Art der Breitensuche dabei auch eine starke Rolle spielen kann, wie gut die Heuristik funktioniert.

2.4.2 Höhenmaximierung

Problem:
Gegeben: Ein einfacher zusammenhängender ungerichteter Graph $G = (V, E)$.
Gesucht: Ein hoher Breitensuchbaum.

Der höchste Breitensuchbaum in G hat eine Höhe gleich dem Durchmesser. Genau wie im Fall der Höhenminimierung könnte man auch hier von jedem Knoten aus einen Breitensuchbaum konstruieren und jeweils die Höhe abspeichern und daraus den höchsten Baum wählen.

Laufzeitanalyse Wie im Fall der Höhenminimierung ergibt sich für allgemeine Graphen eine Laufzeit T^{worst} aus $O(|V|^3)$ und speziell für planare Graphen T^{worst} aus $O(|V|^2)$.

Auch hier wird eine Heuristik vorgestellt, mit der in linearer Zeit versucht wird, einen geeigneten Knoten zu finden, der als Wurzel eines Breitensuchbaums einen hohen Baum liefern würde. Mehr als linearer Zeit kann man wegen der linearen Gesamtlaufzeit des *Planar-Separator-Theorem* nicht für diesen Zweck investieren.

Algorithmus 4 : Heuristik zur Höhenmaximierung durch geeignete Wurzelwahl

Eingabe : Ein zusammenhängender Graph $G = (V, E)$ mit einem Startknoten w und einem Parameter i.
Ausgabe : Geeigneter Wurzelknoten für die Konstruktion eines hohen Breitensuchbaums.

1 Bilde einen Breitensuchbaum $B_w = (V, E_w)$ mit w als Wurzel.
2 Wähle einen Knoten s aus dem höchsten Level von B_w .
3 Setze w auf s und führe das ganze i-mal aus .
4 Gib w zurück.

Lemma 2.4 *Die Behauptung ist, dass der so gewonnene Breitensuchbaum mindestens so hoch ist wie der anfängliche Breitensuchbaum B_w.*

Beweis:
Gegeben sei ein Breitensuchbaum B_w in Graphen $G = (V, E)$ mit der Höhe h und sei s ein Konten der im Level(h) von B_w liegt. Es existiert ein kürzester Pfad P zwischen w und s in B_w mit der Länge h, der in G ebenfalls ein kürzester Pfad zwischen diesen beiden Knoten ist. Somit gilt für die Höhe eines Breitensuchbaums B_s, der s als Wurzel hat:

$$\text{Höhe}(B_s) \geq h.$$

Also kann nach einer konstanten Anzahl von Durchläufen die Höhe sich nicht verschlechtern und es ist ziemlich Wahrscheinlich, dass ein Knoten v existiert, so dass der kürzester Pfad P' zwischen s und v in G größer als h ist.

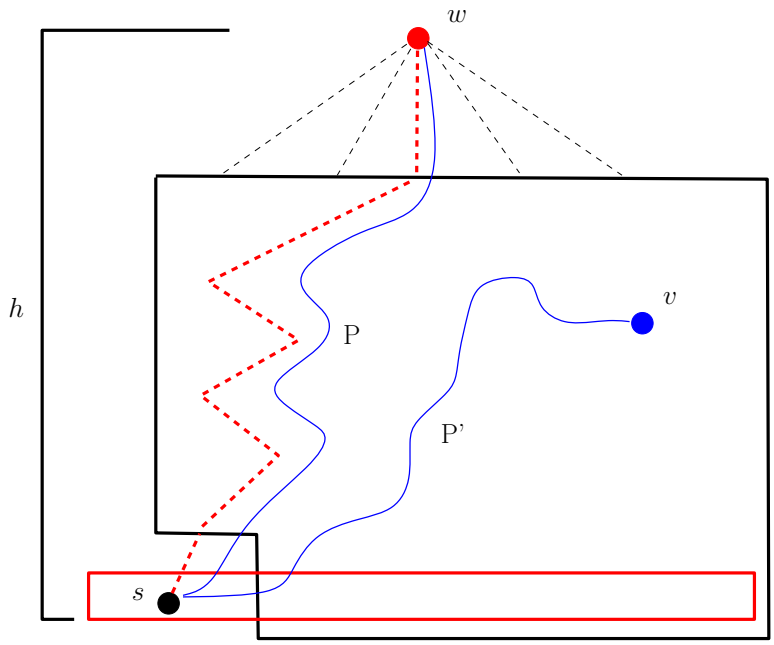


Abbildung 9: Veranschaulichung der eventuell möglichen Höhenmaximierung.

2.4.3 Höhenabschätzung

Eine andere Möglichkeit, die sich sowohl statt der Höhenminimierungsheuristik als auch der Höhenmaximierungsheuristik anbietet, ist die Höhenabschätzungsheuristik. In dieser Heuristik versucht man, in einem gegebenen Graphen durch eine konstante Anzahl von Breitensuchen, für jeden Knoten so genau wie möglich abzuschätzen, welche Höhe ein Breitensuchbaum mit diesem Knoten als Wurzel mindestens hätte. Hierfür wird die Level-Eigenschaft von Breitensuchbäumen verwendet. Nach dem für jeden Knoten die Breitensuchhöhe aus diesem Knoten abgeschätzt wird, kann je nach entsprechender Phase des *Planar-Separator-Theorem* ein Knoten mit niedriger bzw. hoher Abschätzung gewählt werden.

Problem:
Gegeben: Einfacher zusammenhängender Graph $G = (V, E)$. Gesucht: Möglichst genaue Abschätzung der Mindesthöhe eines Breitensuchbaums aus jedem Knoten des Graphen G.

Wir bezeichnen mit $B_w = (V, E_{B_w})$ einen Breitensuchbaum im Graphen $G = (V, E)$, dessen Wurzel w ist und die Höhe h hat. Analog bezeichnen wir mit B_v einen beliebigen Breitensuchbaum mit v als Wurzel. Es gilt stets:

$$\text{Höhe}(B_v) \geq \max\{(h - \text{dist}(w, v)), \text{dist}(w, v)\}.$$

Aus dieser Tatsache ergibt sich eine Heuristik, mit der man versucht, für jeden Knotens v in G die Höhe eines Breitensuchbaums mit v als Wurzel anhand eines bereits konstruierten Breitensuchbaums abzuschätzen. Diese Abschätzung bezeichnen wir mit $\text{app}_G(v)$.

Algorithmus 5 : Heuristik zur Abschätzung der Mindesthöhe von Breitensuchbäumen.

Eingabe : Ein zusammenhängender Graph $G = (V, E)$ und ein Parameter i.
Ausgabe : Abschätzung der Breitensuchhöhe für jeden Knoten des Graphen.

1. Setze anfänglich $\text{app}_G(v)$ für alle Knoten $v \in G$ auf Null.
2. Wähle zufällig einen Knoten w in G, der noch nicht markiert ist.
3. Konstruiere einen beliebigen Breitensuchbaum mit w als Wurzel und markiere w.
4. Aktualisiere nun für alle v aus V $\text{app}_G(v)$ wie folgt:

$$\text{app}_G(v) = \max\{(h - \text{dist}(w, v)),\ \text{dist}(w, v),\ \text{app}_G(v)\}.$$

5. Wiederhole die Schritte 2, 3 und 4 i-mal.

Lemma 2.5 *Zu jedem Zeitpunkt ist die folgende Invariante I erfüllt, wobei B_v einen beliebigen Breitensuchbaum bezeichnet, der v als Wurzel hat.*

$$I := \textit{f.a } v \in G\ :\ \textit{app}_G(v) \leq \textit{Höhe}(B_v).$$

Analyse Würde man in der oben angegebenen Heuristik aus allen Knoten des Graphen einen Breitensuchbaum konstruieren ($i = |V|$), so würde die resultierende Abschätzung für jeden Knoten v in $G = (V, E)$ mit der Höhe eines Breitensuchbaums, der v als Wurzel hat, übereinstimmen und man hätte eine Laufzeit T^{worst} in $\text{O}(|V|^3)$.

Es ist jedoch zu hoffen, dass bereits eine geringe Anzahl von Breitensuchbäumen existieren, mit Hilfe derer man aufgrund des abgeschätzten Wertes $\text{app}_G(v)$ eines beliebigen Knotens v in G feststellen kann, ob dieser Knoten als Wurzel eines Breitensuchbaums geeignet ist oder nicht. Falls wir zum Beispiel die Höhenminimierung im Abschnitt 2.4.1 betrachten, würden wir nach der Anwendung der Heuristik einen Knoten wählen, dessen abgeschätzte Breitensuchhöhe minimal ist. Denn die Knoten, die unmittelbar nahe an einem Zentrumsknoten des Graphen liegen, hätten einen niedrigeren abgeschätzten Wert als Mindesthöhe. Analog würde man bei der Höhenmaximierung im Abschnitt 2.4.2 eher einen Knoten v wählen, dessen abgeschätzter Wert $\text{app}_G(v)$ maximal ist. Da man weiß, dass ein Breitensuchbaum aus v mindestens die Höhe $\text{app}_G(v)$ hätte.

Ein weiterer Aspekt, der sehr stark die Abschätzung beeinflusst, ist die Wahl der zufälligen Knoten, von denen aus ein Breitensuchbaum konstruiert werden soll. Würden nämlich diese Knoten nahe beieinander liegen, so würde man einiges an Laufzeit für die Breitensuchen investieren, aber keine genaue Abschätzung erhalten. Es wäre also von Vorteil, wenn diese zufällig gewählten Knoten über den Graphen verteilt wären und einen möglichst weiten Abstand von einander hätten. Diese Beobachtung motiviert die folgende Modifikation.

Modifikation Man kann die Knotenwahl der Heuristik verändern und nicht wie vorher zufällig eine Anzahl von Knoten wählen. Dabei wird immer ein Knoten, der von der Wurzel des bereits konstruierten Vorgängerbaumes am weitesten entfernt ist, als neue Wurzel ausgesucht. Dann von diesem

Knoten erneut eine Breitensuche gestartet und die Baumhöhen aller Knoten abgeschätzt. Durch diese Strategie der Knotenwahl wird, wie bei der Höhenmaximierungsheuristik schon bewiesen, die Höhe des neuen Breitensuchbaums eventuell höher sein und es ist möglich die Baumhöhen dadurch besser abzuschätzen.

Algorithmus 6 : Modifizierte Heuristik zur Abschätzung der Mindesthöhe

---

Eingabe : Ein zusammenhängender Graph $G = (V, E)$ und ein Parameter i.
Ausgabe : Abschätzung der Breitensuchhöhe für jeden Knoten des Graphen.

1. Setze anfänglich $\text{app}_G(v)$ für alle Knoten $v \in G$ auf 0.
2. Wähle zufällig einen Knoten w in G, der noch nicht markiert ist.
3. Konstruiere einen beliebigen Breitensuchbaum B_w mit w als Wurzel und markiere w.
4. Aktualisiere nun für alle v aus V $\text{app}_G(v)$ wie folgt:

$$\text{app}_G(v) = \max\{(h - \text{dist}(w, v)), \text{dist}(w, v), \text{app}_G(v)\}\,.$$

5. Wähle nicht markierten Knoten s im höchstmöglichen Level bezüglich B_w.
6. Konstruiere einen Breitensuchbaum B_s mit s als Wurzel und markiere s.
7. Setze w auf s.
8. Wiederhole die Schritte 3 bis 7 i-mal.

2.4.4 Sternbäume

Alle Heuristiken, die wir bisher betrachtet haben, sind von einem Wurzelknoten zur Konstruktion eines geeigneten aufspannenden Baumes ausgegangen. Es stellt sich die Frage, ob es ein wurzelunabhängiges Verfahren gibt, Bäume mit niedriger Höhe zu konstruieren. Diese Frage wurde näher untersucht und daraus das folgende Verfahren abgeleitet.

Idee Man kann sich den Grad eines Knotens zu Nutzen machen, indem man aus Knoten mit hohem Grad lokale Bäume konstruiert und sie danach geschickt zu einem aufspannenden Baum des Graphen verbindet. Mit dem geschickten Verbinden der lokalen Bäume ist hierbei ein Verfahren gemeint, das aus einem konstruierten Wald einen aufspannenden Baum konstruiert, in dem keine sehr lange maximale Pfade zwischen zwei Knoten existieren. Anschließend könnte man im konstruierten aufspannenden Baum einen Zentrumsknoten als Wurzel wählen, damit die Höhe des Baumes minimal wird.

Für diesen Ansatz eignen sich UNION-FIND Techniken zum Vereinigen von Knotenmengen, die auch im Algorithmus von Kruskal [8] verwendet werden. Es gibt sehr viele Möglichkeiten solche Bäume zu bilden und den Knotengrad zu nutzen, daher werden wir hier kein konkretes Verfahren angeben und lediglich einige Möglichkeiten und Ideen und die dazugehörigen Testergebnisse aus den Experimenten vorstellen.

Veranschaulichung Eine Möglichkeit einen aufspannenden Baum zu konstruieren wäre es, Knoten zu suchen, die am meisten lokale Bäume miteinander verbinden. Durch die Wahl einiger Kanten dieses Knotens würden dann sehr viele Bäume in einem Schritt zusammenwachsen und es hätte keine negative Auswirkung auf die Längen der maximalen Pfade im aufspannenden Baum. In der Abbildung 10 wird nochmal die Idee des Sternbaums verdeutlicht. Nachdem die lokalen Bäume konstruiert sind, stehen zwei Knoten zur Wahl, die die meisten lokalen Bäume mit einander verbinden. Wir wählen Kandidat 1 und erhalten somit einen aufspannenden Baum, dessen Zentrum den Kandidat 1 enthält. Falls wir Kandidat 1 als Wurzel wählen sollten, so bekämen wir einen Baum der Höhe 3.

Man könnte natürlich auch ziemlich ungeschickt die lokalen Bäume miteinander verbinden, so dass der daraus resultierende aufspannende Baum trotz der Wahl eines Zentrumsknoten als Wurzel sehr hoch wäre (viel höher als der Durchmesser des Graphen). In der Abbildung 11 sieht man ganz deutlich, was passieren kann, wenn man nicht berücksichtigt, welche Kanten zum Verbinden der Teilbäume zu einem Ganzen auswählt werden. Ein Zentrumsknoten als Wurzel des aufspannenden Baumes liefert die Höhe 6, welches doppelt so groß ist wie die Höhe, die wir beim aufspannenden Baum in Abbildung 10 erhalten haben.

Experimentelle Beobachtung Es wurden einige Experimente mit planaren Eingabegraphen durchgeführt, in denen der Erwartungswert der Baumhöhe von Sternbaümen mit der Baumhöhe von Breitensuchbäumen verglichen wurden. Was die praktische Anwendung dieses Verfahrens betrifft, so haben die Experimente mit unterschiedlichen Graphklassen jedoch gezeigt, dass man sehr viel schlechtere Bäume hinsichtlich der Minimalität der Baumhöhen erhält als ein ganz normaler Breitensuchbaum aus einem beliebigen Knoten des Graphen, da ein Breitensuchbaum immer eine Höhe kleiner gleich dem Radius des Graphen hat und das bei Sternbäumen nicht garantiert ist. Durch diese Feststellung

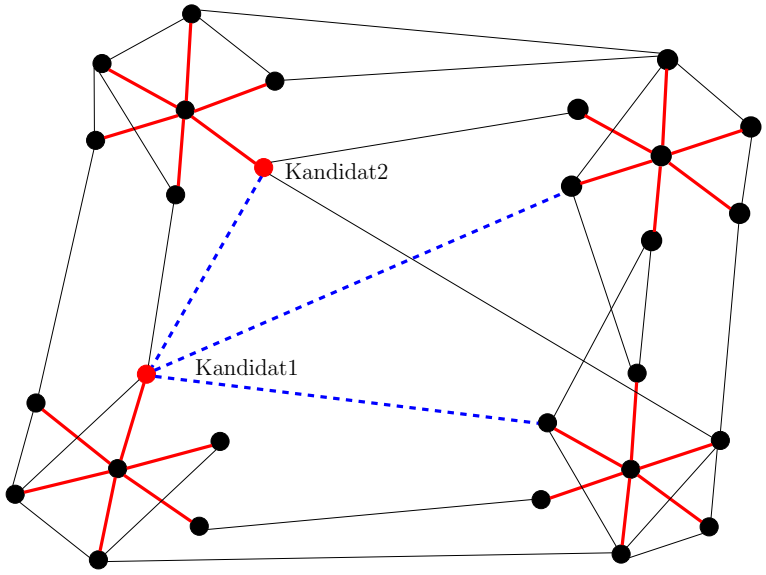


Abbildung 10: Geschicktes Verbinden der lokalen Bäume zu einem aufspannenden Baum.

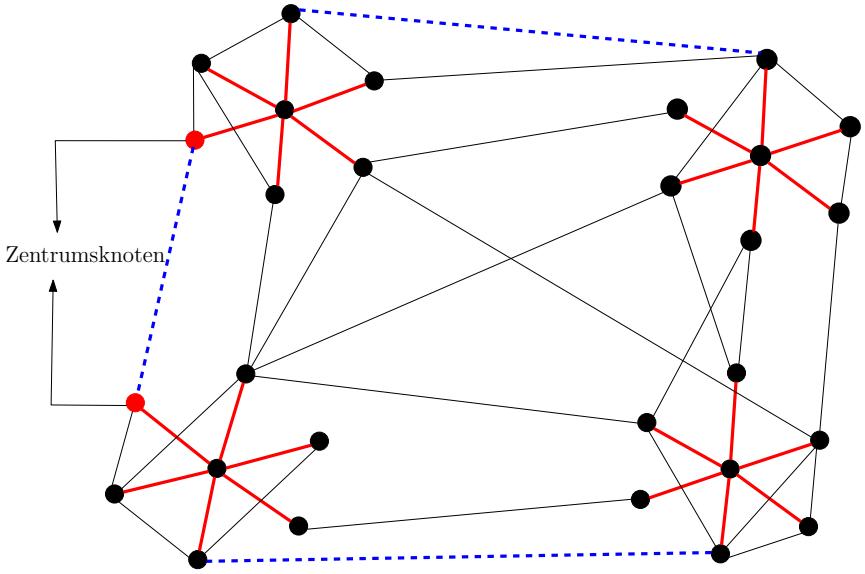


Abbildung 11: Ungeschicktes Verbinden der lokalen Bäume zu einem aufspannenden Baum.

und Beobachtung wird der wurzelabhängige Optimierungsansatz bei der Höhenminimierungsheuristik bekräftigt, deshalb wird für die spätere Optimierung des PST der wurzelabhängige Ansatz weiterverfolgt.

2.4.5 Experimente

Nachdem einige Heuristiken im Abschnitt 2.4 vorgestellt wurden, möchten man diese auch testen und statistisch bewerten, damit man besser einschätzen kann, wie gut bzw. schlecht sie in der Praxis funktionieren und wie weit sie von der optimalen Lösung abweichen. Aus diesen Beobachtungen werden dann die besten Verfahren ausgesucht und später im PST als Optimierung eingesetzt, um die praktische Performanz des Theorems zu erhöhen.

Freiheitsgrade Bei diesen Heuristiken gab es einige Wahlmöglichkeiten wie die Wahl eines Startknotens, Wahl einer Breitensuchvariante, Anzahl der Durchläufe und die Auswahl der Testgraphen für die Experimente. Somit hat man die Qual der Wahl und muß sich für einige bestimmte Testreihen entscheiden und kann, so leid es einem auch tuen mag, nicht alle Möglichkeiten ausschöpfen. Unser bemühen ist es jedoch, eine sinnvolle Testreihe aufzustellen, in der die wesentlichen Aspekte zum Vorschein kommen.

Testreihe Es werden alle Breitensuchvarianten, die in Abschnitt 2.3.4 behandelt wurden, untersucht. Dabei wird ein Testlauf für jeder dieser Breitensuchen durchgeführt, bei dem jeder Knoten eines Eingabegraphen genau einmal als Startknoten gewählt wird. Beobachtungen haben gezeigt, dass bereits bei 10 Durchläufen eine deutliche Verbesserung erzielt werden kann. Daher hat man in den Testläufen genau 10 Iterationen den Heuristiken für die *Höhenmaximierung* und *Höhenminimierung* spendiert. Die Ergebnisse werden schließlich mit dem nicht optimierten Ansatz verglichen.

Testgraphen Was die Testgraphen betrifft, so hat man sich für einige planare Graphklassen entschieden, die als Eingabegraphen verwendet werden. Unser Ziel ist es, die Eignung der verschiedenen Breitensuchkonstruktion in der Heuristik zu analysieren, in dem verschiedene Varianten der Heuristik auf unterschiedliche Graphklassen getestet und daraus die Varianten ausgewählt werden, die angewendet auf den meisten dieser Eingabegraphen ein gutes Ergebnis erbracht haben. Ebenso soll festgestellt werden, welche Breitensuchbäume eher nicht für diesen Ansatz geeignet sind. Insgesamt werden die folgenden Graphklassen betrachtet.

Graphklasse	Besondere Merkmale
Gittergraphen	Graphen, die eine Gitterstruktur aufweisen.
Diametergraphen	Graphen, die bereits trianguliert sind und einen hohen Durchmesser haben.
LEDA-Graphen	Graphen, die durch einer LEDA-Triangulierung entstanden sind.
Delaunay-Graphen	Graphen, die durch eine DELAUNAY-Triangulierng entstanden sind.
Straßengraphen	Graphen, die aus Straßenkarten von Städten entstehen, indem man Straßen durch Polygonzüge ersetzt.

Diagramme Im Folgenden werden wir einige Diagramme, die wir aus den Experimentdaten erstellt haben präsentieren, die uns vermitteln, wie die jeweiligen Verfahren in der Praxis funktionieren. Dabei erläutern wir jeweils kurz um welche Graphklasse es sich handelt und welche besondere Eigenschaften die Graphklassen jeweils haben. Um besser auf die Diagramme und Verfahren bezugnehmen zu können, werden die folgenden Bezeichnungen eingeführt, die alle für eine bestimmte Art der Breitensuche stehen, die in den Heuristiken Verwendung findet. Desweiteren bezeichnet bei den Boxplots

ein Kreuz immer den Erwartungswert der Baumhöhen und eine Linienzug den Median der Baumhöhen. Ein anderen Begriff, den wir hier einführen, ist das Höhenspektrum. Damit bezeichnen wir den Intervall zwischen der minimalen Breitensuchhöhe und der maximalen Breitensuchhöhe, die eine Heuristik angewendet auf den Graphen liefert.

- WB: Ohne Anwendung einer Heuristik.(**W**ithout Heuristik)
- SB: Heuristik mit der Standardbreitensuche.(**S**tandard)
- VB: Heuristik mit der Knoten permutierenden Breitensuche.(**V**ertex permuted)
- EB: Heuristik mit der Kantenpermutierenden Breitensuche.(**E**dge permuted)
- AB: Heuristik mit der aufsteigend sortiereten Breitensuche.(**A**scending sorted)
- DB: Heuristik mit der absteigend sortierten Breitensuche.(**D**escending sorted)
- BB: Heuristik mit der balancierten Breitensuche.(**B**alance)
- PB: Heuristik mit den permutierenden Breitensuche VB oder EB.(**P**ermuted)

Gittergraphen Wie der Name schon sagt, sind Gittergraphen Graphen, die eine Gitterstruktur aufweisen. Diese Graphen haben eine sehr übersichtliche Struktur und finden sehr viel Verwendung in der Praxis. Es gibt quadratische Gitter und rechteckige Gitter. Wir bezeichnen einen Gittergraphen als rechteckig, wenn die Knoten in einer Einbettung, die einem realen Gitter entspricht, in horizontaler und vertikaler Ausrichtung nicht gleichverteilt sind. Die Bezeichnungen rechteckiges und quadratisches Gitter dienen nur dem Zweck zu unterscheiden, wie sich die Heuristiken bei unterschiedlicher Verteilung der Knoten in x und y Ausrichtung verhalten.

Wenn man das Spektrum der Breitensuchhöhen in Gittergraphen betrachtet, so sieht man, dass Höhen die dem Radius oder dem Durchmesser näher sind weniger oft vorkommen als Höhen die irgendwo dazwischen liegen. In einer Einbettung eines Gittergraphen, die einem Gitter entspricht, ist zu erkennen, dass das Zentrum des Graphen in der Mitte des Gitters liegt. Aus Symmetriegründen gibt es oft mehrere Knoten, die die gleiche Breitensuchbaumhöhe liefern würden.

Zur Demonstration wurden jeweils ein quadratisches (40×40) und rechteckiges Gitter (80×20) mit 1600 Knoten untersucht und miteinander verglichen. Im quadratischen Gitter beträgt der Radius 40 und es existieren genau vier Knoten, die als Wurzel des Breitensuchbaums diese Höhe liefern. Der Durchmesser beträgt 78 und es existieren ebenfalls 4 Knoten, die als Wurzel des Breitensuchbaums den Durchmesser als Baumhöhe liefern. Außerdem kann man beobachten, dass die am meisten vorkommende Baumhöhe ungefähr dem Mittelwert vom Radius und Durchmesser entspricht. Die Länge unseres Höhenspektrums ist hier 39 (Siehe Abbildung 12 Unten (links)). Die Situation in einem rechteckigen Gitter mit gleicher Knotenzahl sieht ein bißchen anders aus. Hier kommen die Breitensuchbaumhöhen für die meisten Werte gleich oft vor bis auf die Höhen die nahe am Radius und Durchmesser liegen (Siehe Abbildung 14).

Insgesamt ist es unwahrscheinlich, dass man per Zufall einen Knoten in einem Gittergraphen wählt, die eine geeignete Breitensuchhöhe liefert, die je nachdem, ob man die Höhe minimieren oder maximieren möchte, dem Radius oder dem Durchmesser entspricht oder zumindest in der Nähe davon liegt.

Höhenminimierung: Nun wenden wir die Höhenminimierungsheuristik auf Gittergraphen an. Hierbei betrachten wir 10 Iterationen. In den Abbildungen 12, 14 und 15 (Unten (rechts)) ist das Höhenspektrum bei Anwendung der Heuristik mit der kantenpermutierenden Breitensuche dargestellt. Außerdem ist auch zu erkennen, dass die permutierenden Breitensuchen deutlich besser im Test abgeschnitten haben als alle anderen Breitensuchen. Die balancierte Breitensuche hat dabei das schlechteste Ergebnis geliefert.

Höhenmaximierung: Die Höhenmaximierungsheuristik lieferte in nur zehn Iterationen immer einen höchsten Breitensuchbaum (Höhe entspricht dem Durchmesser). Der Grund hierfür ist der, dass bei der Wahl eines beliebigen Startknotens, nach der ersten Breitensuche bereits, ein Knoten auf der äußeren Facette der Einbettung mit einem Knotengrad gleich 2 gewählt wird. Es existieren genau vier solche Knoten. Wenn man dann von einem dieser Knoten eine Breitensuche startet, so hat dieser eine Höhe gleich dem Durchmesser und ist somit ein Breitensuchbaum mit einer maximalen Höhe.

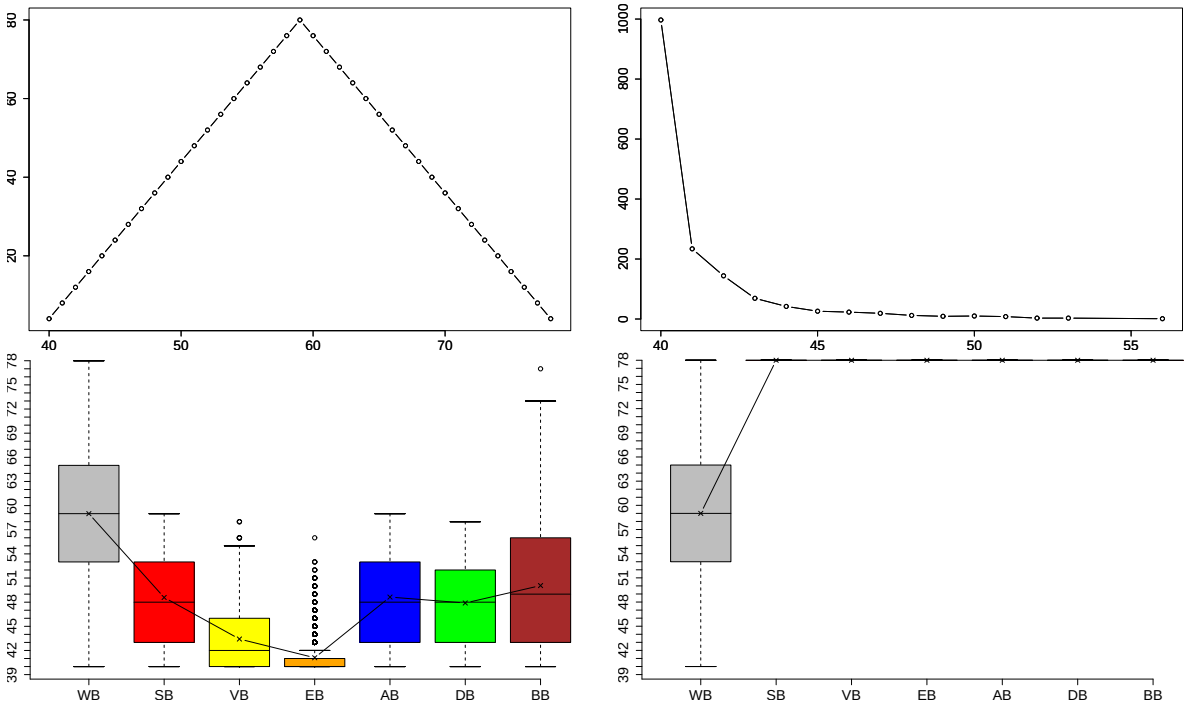


Abbildung 12: Quadratisches Gitter mit 1600 Knoten (40×40). Oben (links): Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit der Baumhöhen ohne Optimierung an. Oben (rechts): Die y-Achse gibt die Häufigkeit der Baumhöhen mit der Höhenminimierungsheuristik und der Breitensuchvariante EB an. Unten: Die x-Achse gibt die Baumhöhe und die y-Achse die angewendete Breitensuchvariante. Links sieht man die Höhenminimierungsheuristik und Rechts die Höhenmaximierungsheuristik.

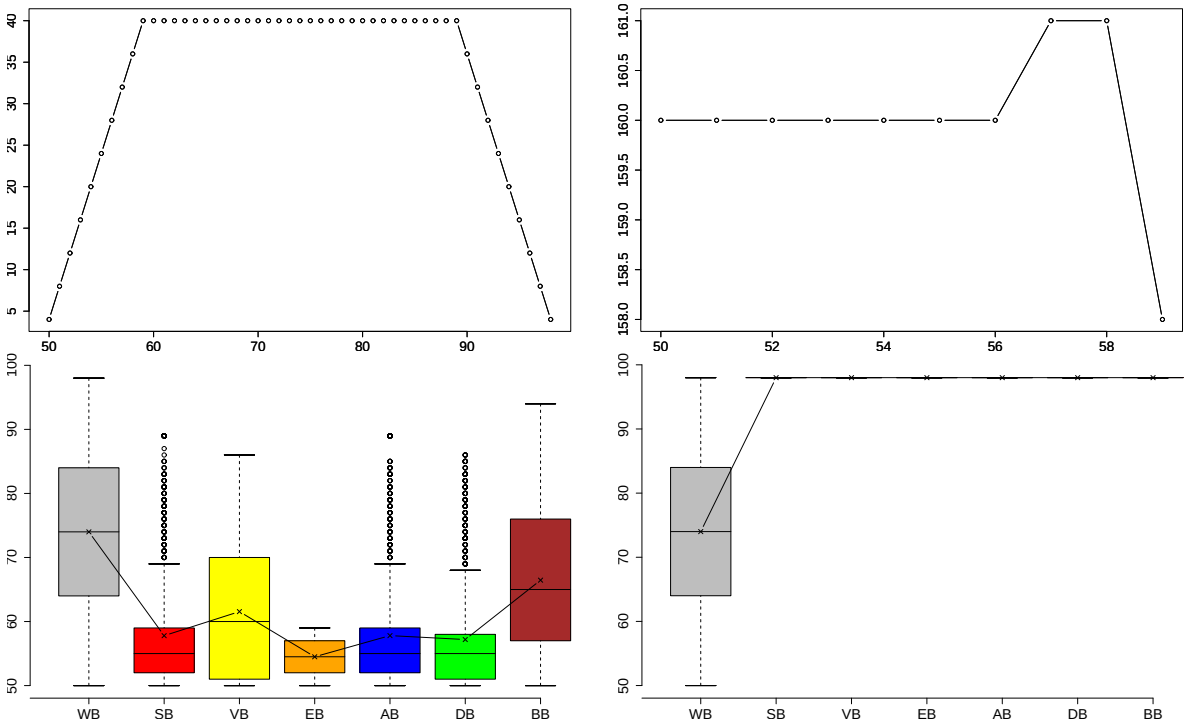


Abbildung 13: Rechteckiges Gitter mit 1600 Knoten (80×20).

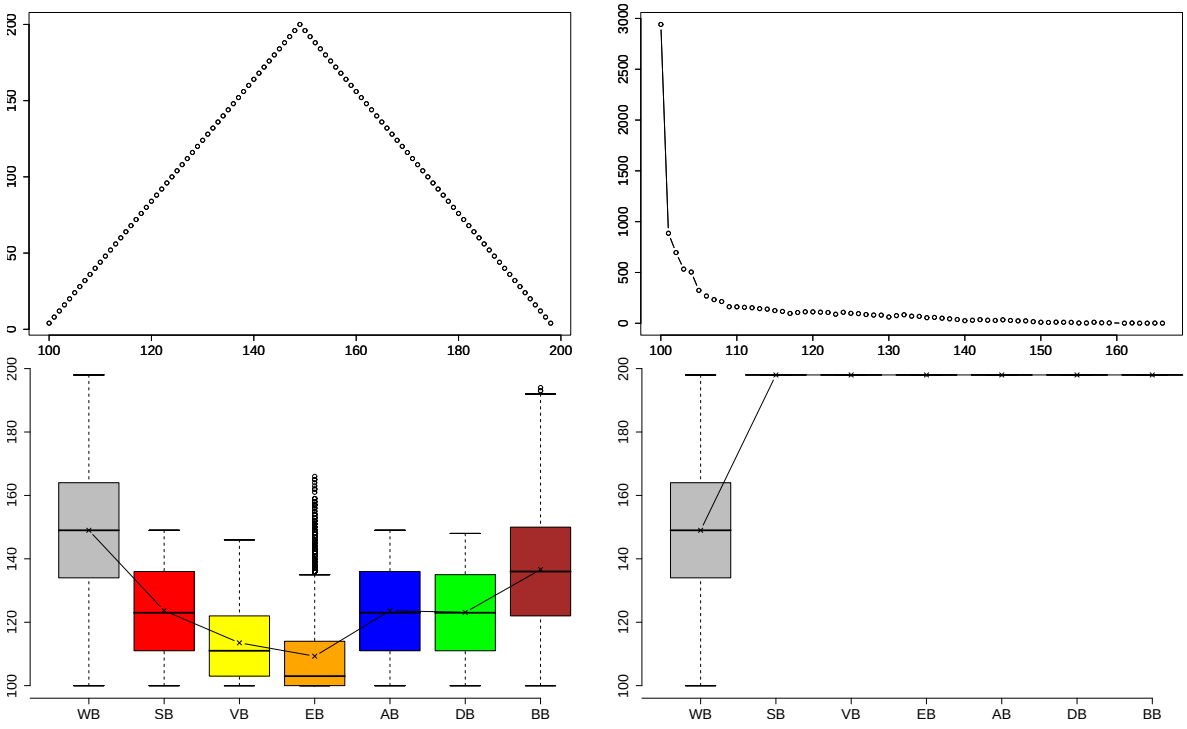


Abbildung 14: Quadratisches Gitter mit 10000 Knoten (100×100).

Diametergraphen Diametergraphen sind planare Graphen, die bereits trianguliert sind und einen möglichst hohen Durchmesser haben. In diesen Graphen existieren also sehr lange kürzeste Pfade zwischen Knoten. Deshalb ist man zuversichtlich, dass die Heuristiken gute Ergebnisse für diese Graphklasse liefern, da bei Graphen, die sehr lange kürzeste Pfade besitzen, meistens ein Mitttelknoten eines solchen Pfades als geeignete Wurzel in Frage kommt und diese von der Heuristik, wegen der Graphstruktur, meistens gefunden werden.

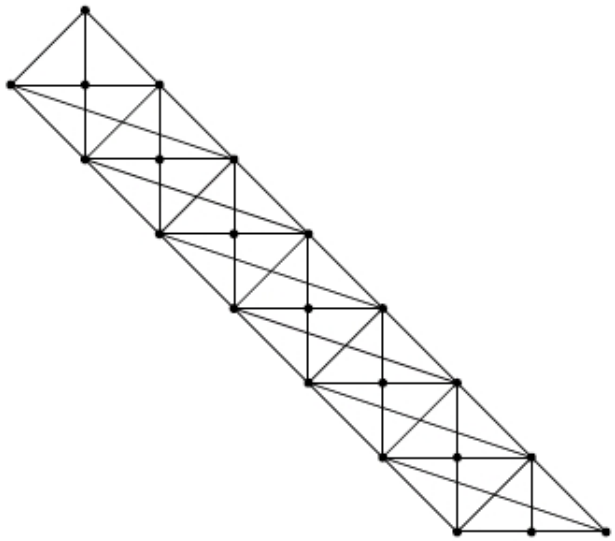


Abbildung 15: Diametergraph mit Durchmesser 7.

Um mal einen Überblick zu verschaffen, welche Höhen insgesamt bei der Konstruktion eines Breitensuchbaums bei der Wahl eines beliebigen Wurzelknotens auftreten, haben wir einen Graphen mit einem Durchmesser gleich 1000 untersucht. Dieser Graph hat 3001 Knoten und 9003 Kanten. Der Radius ist gleich 500 und es existieren wenige Zentrumsknoten, was zur Folge hat, dass der Radius als Breitensuchbaumhöhe selten bei zufälliger Wahl eines Wurzelknotens angenommen wird. Man hat also ein Höhenspektrum zwischen 500 und 1000 für die Breitensuchbaumhöhe.

Allgemein kann man sagen, dass für einen Diametergraphen mit einem Durchmesser gleich d die kleinste Breitensuchhöhe $d/2$ und die größte d beträgt. Je größer also der Durchmesser wird, desto größer wird unser Höhenspektrum. Hierbei wird die Wahrscheinlichkeit natürlich kleiner, einen zufälligen Knoten zu wählen, der eine sehr geringe Breitensuchhöhe liefert.

Höhenminimierung: Ein Diametergraph mit Durchemesser 1000 wurde erzeugt und getestet. Dabei ergaben sich einige Ergebnisse für die Baumhöhe, die in den Abbildungen 16 und 17 zu sehen sind. Es ist deutlich zu erkennen, dass bei der Anwendung der balancierten Breitensuche in der Höhenminimierungsheuristik fast keine Optimierung erzielt wurde. Vor allem ist auch zu erkennen, dass bei allen anderen Varianten der Breitensuche eine deutliche Verbesserung erzielt wurde. Der Erwartungswert für die Breitensuchbaumhöhe beträgt 500, was dem Radius des Graphen entspricht. Das bedeutet, dass immer ein Breitensuchbaum mit der geringsten Höhe gefunden wurde. Weitere Tests haben ergeben, dass man mit viel weniger als 10 Iterationen ebenfalls sehr gute Ergebnisse für diese spezielle Graphklasse erzielen kann, unabhängig davon, wie groß ihr Durchmesser ist.

Höhenmaximierung: Die Tests haben ergeben, dass bereits mit 2 Iterationen in der Höhenmaximierungsheuristik ein höchster Breitensuchbaum gefunden werden kann. Dabei wurde diese Höhe bei allen Breitensuchvarianten unter der Anwendung der Höhenmaximierungsheuristik erreicht.

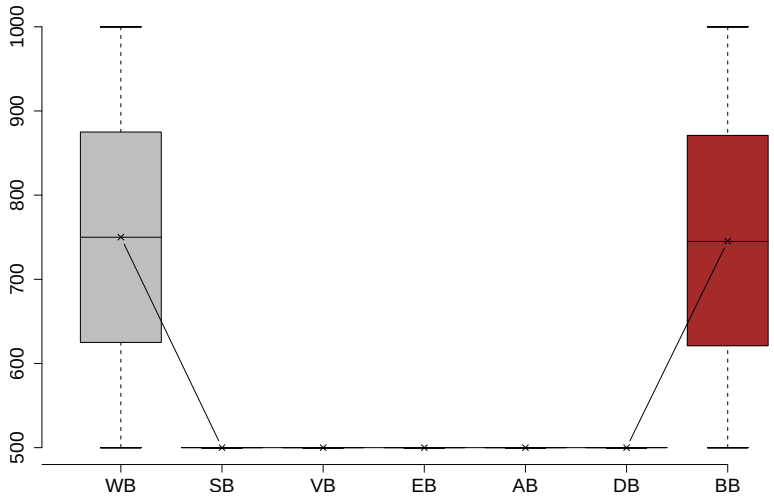


Abbildung 16: Höhenminimierung bei einem Diametergraphen mit dem Durchmesser 1000. Die x-Achse gibt die angewendete Breitensuchvariante und die y-Achse die Breitensuchbaumhöhe an.

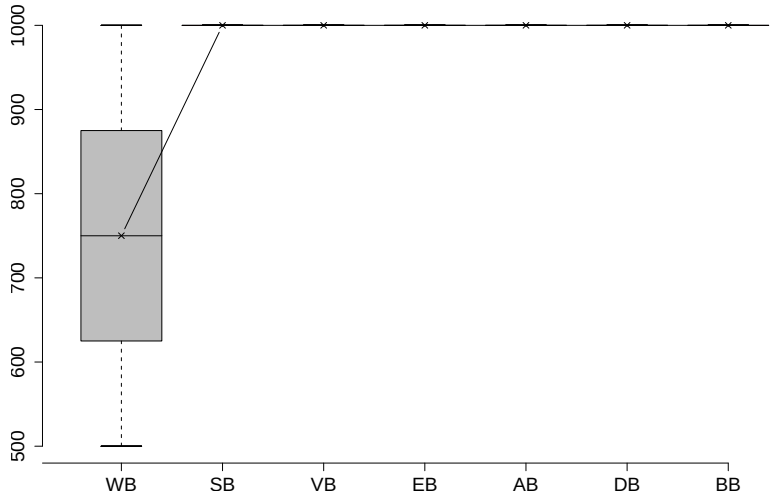


Abbildung 17: Höhenmaximierung bei einem Diametergraphen mit dem Durchmesser 1000. Die x-Achse gibt die angewendete Breitensuchvariante und die y-Achse die Breitensuchbaumhöhe an.

Delaunay-Graphen Delaunay-Graphen sind Graphen, die durch eine Delaunay-Triangulierung entstanden sind [8]. Diese Graphen finden in der industriellen Bildverarbeitung oft Verwendung. Bei der Generierung solcher Graphen werden zufällig eine Anzahl von Knoten beliebig in die Ebene verstreut und das Voronoidiagramm dieser Knoten gebildet. Danach wird das Dualitätsprinzip planarer Graphen verwendet und der Dualgraph des Voronoi-Diagramms bestimmt. In Abbildung 18 sehen wir ein Voronoidiagramm aus dem ein planarer Delaunay-Graph entsteht. Also sind Delaunay Graphen insbesondere planar und alle Facetten außer der äußeren Facette sind trianguliert.

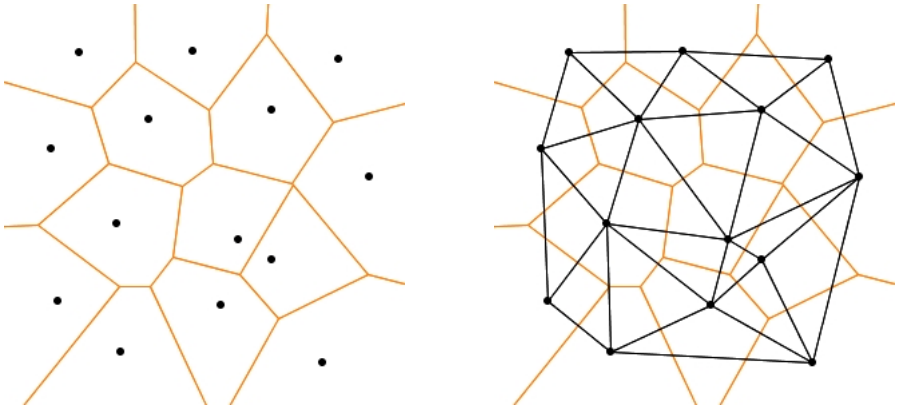


Abbildung 18: Links ist ein Voronoidiagramm abgebildet und Rechts der zugehörige Delaunay-Graph.

Beobachtungen haben gezeigt, dass das Höhenspektrum bei Delaunay-Graphen sehr klein ist und dadurch sich die Wahrscheinlichkeit eine große Verbesserung zu erzielen vermindert. Unsere Vermutung hat sich experimentell bestätigt. Für diese spezielle Graphklasse haben wir auf einigen Graphen, in denen der Mittelwert der Knotengrade $2,5$ beträgt, die Heuristiken angewendet. Alle diese Graphen unterschiedlicher Knotenzahl haben sich als Ähnlich hinsichtlich der Optimierung erwiesen. Wir werden deshalb nur die erstellten Diagramme für einen Delaunay-Graphen mit 10000 Knoten und 25000 Kanten im Folgenden angeben.

Höhenminimierung: Es hat sich herausgestellt, dass die Breitensuchhöhe dieser Graphklasse sich nicht so gut minimieren läßt wie die anderen Graphklassen. Die Abbildungen 19, 20 und 21 verdeutlichen diesen Sachverhalt. Der Radius des Graphen ist 44 und der Durchmesser gleich 54. Sowohl der Radius als auch der Durchmesser werden als Breitensuchöhen selten angenommen. Ein viertel aller Knoten liefern die Baumhöhe 51, welches nahe am Durchmesser liegt. Man sieht, dass die Optimierung die Häufigkeit dieser Höhe nur gering verringern konnte. Aber die Häufigkeit der Höhe 52 konnte ungefähr von 2000 auf 1000 reduziert werden. Der Radius wird ungefähr 250 mal angenommen, was im Vergleich zur nicht optimierten Variante deutlich gestiegen ist. Desweiteren kann man beobachten, dass die Häufigkeit der niedrigen Höhen gestiegen und die der großen Höhen gefallen ist. Die Höhe 50 ist hierbei die Grenze zwischen den niedrigen und den großen Baumhöhen. Die verschiedenen Breitensuchvarianten liefern unterschiedliche Ergebnisse, wobei auch hier die permutierenden Breitensuchen bessere und die balancierte Breitensuche schlechtere Ergebnisse geliefert hat.

Höhenmaximierung: Das Auffinden eines hohen Breitensuchbaums hat sich auch für diese Graphklasse als erfolgreich erwiesen. In fast 100% der Fälle wird durch die Optimierung ein Breitensuchbaum der Höhe größer gleich 52 gefunden. Desweiteren kann man beobachten, dass die Art der Breitensuche, die für die Höhenmaximierung verwendet wird, hier auch keine Rolle spielt.

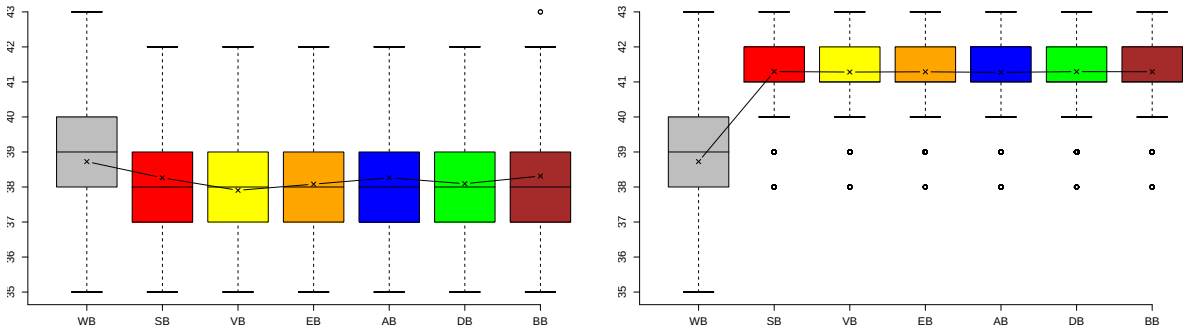


Abbildung 19: Oben: Delaunay-Graph mit 5000 Knoten und 12500 Kanten. Die x-Achse gibt die Breitensuchvariante und die y-Achse die Baumhöhe an. Links: Höhenminimierungsheuristik. Rechts: Höhenmaximierungsheuristik.

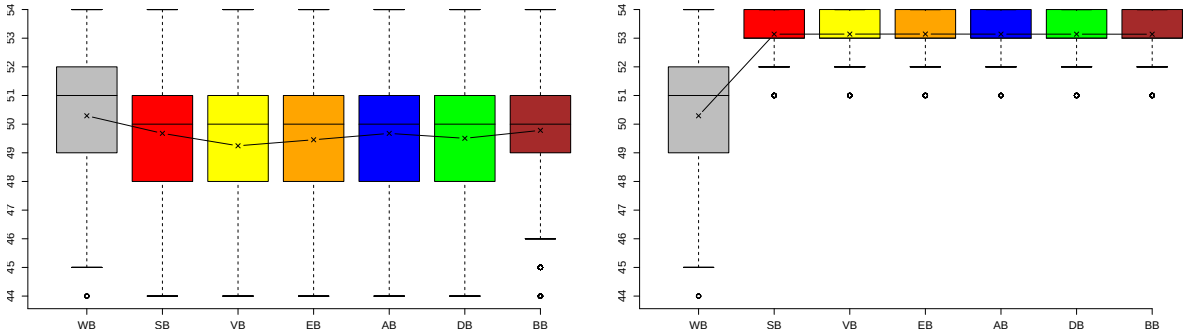


Abbildung 20: Delaunay-Graph mit 10000 Knoten und 25000 Kanten. Die x-Achse gibt die Breitensuchvariante und die y-Achse die Baumhöhe an. Links: Höhenminimierungsheuristik. Rechts: Höhenmaximierungsheuristik.

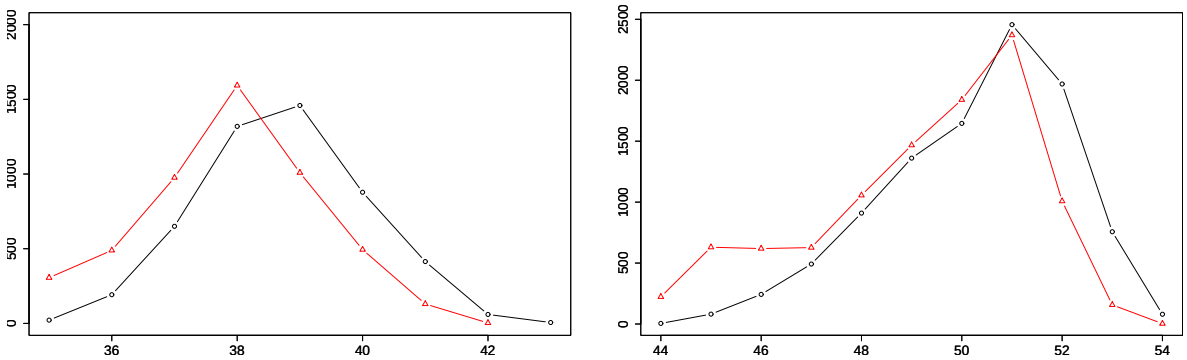


Abbildung 21: Links: Delaunay-Graph mit 5000 Knoten und 12500 Kanten. Rechts: Delaunay Graph mit 10000 Knoten und 25000 Kanten.Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit an. $\circ$ = Ohne Optimierung, $\triangle$ = Mit Optimierung EB.

LEDA-Graphen LEDA-Graphen sind Graphen, die durch eine LEDA-Triangulierung [8] entstanden sind. LEDA-Graphen gehören genau wie die Delaunay-Graphen zur Klasse der planaren Graphen.

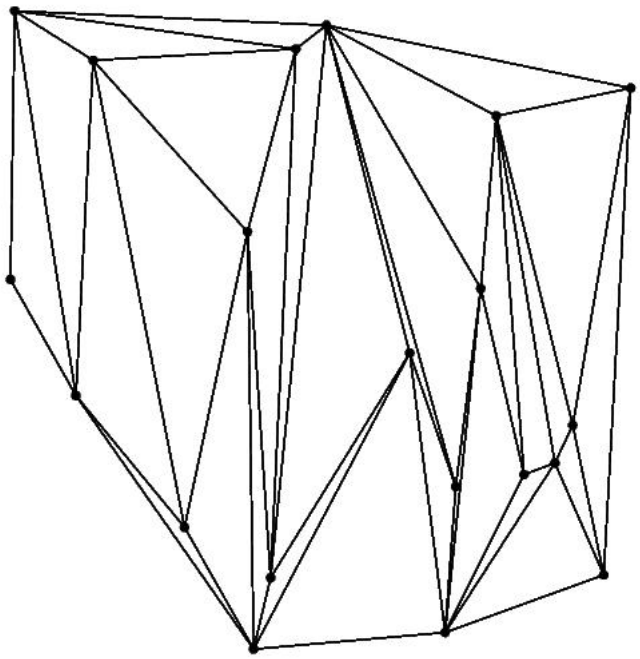


Abbildung 22: LEDA-Graph mit 20 Knoten und 50 Kanten.

Höhenminimierung: In dieser Graphklasse konnte mit der Höhenminierungsheuristik immer eine nennenswerte Verbesserung erzielt werden. Insbesondere wurde dadurch der Radius deutlich öfters angenommen. In der Abbildungen 23, 2.4.5 und 25 sieht man den Unterschied zwischen der nicht optimierten und optimierten Variante.

Der Radius des Graphen mit 10000 Knoten und 25000 Kanten ist 10 und wird als Höhe eines Breitensuchbaums selten angenommen. Wendet man aber die Optimierung PB (genauer VB) an, so sieht man, dass ein Breitensuchbaum mit einer Höhe gleich dem Radius ungefähr 4000 mal gefunden wird. Auch noch zu erkennen ist die Tatsache, dass mit mehr als 80%-er Wahrscheinlichkeit bei der Anwendung dieser Optimierung aus einem beliebigen Startknoten eine Höhe kleiner gleich 13 zu erwarten ist.

Höhenmaximierung: Es wird ein relativ Hoher Baum mit jeder Breitensuchvariante gefunden. Der Durchmesser des Graphen ist 19 und wir finden aus jedem beliebigen Startknoten mit der Heuristik einen Baum, dessen Höhe größer als 17 ist. Anhand der Abbildung 2.4.5 kann man diese Tatsache beobachten. Insbesondere sieht man ganz deutlich anhand der Abbildung 25, dass die 19 als Breitensuchbaumhöhe sehr selten vorkommt.

Insgesamt hat unsere Testreihe mit LEDA-Graphen unterschiedlicher Knotenanzahl n und Kantenanzahl $2.5 \cdot n$ ergeben, dass die Optimierung PB nicht immer, aber im Schnitt betrachtet, bessere Ergebnisse liefert als alle anderen Varianten der Optimierungsheuristik.

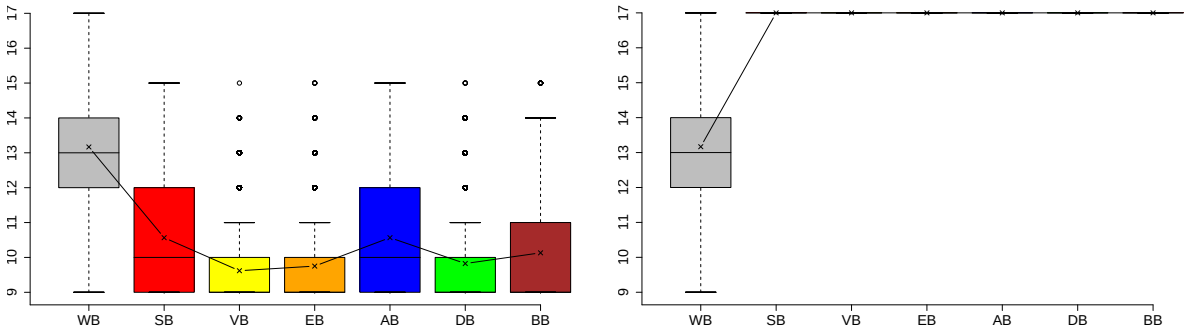


Abbildung 23: Die x-Achse gibt die Breitensuchvariante und die y-Achse die Baumhöhe an. LEDA-Graph mit 5000 Knoten und 12500 Kanten. Links: Anwendung der Höhenminimierungsheuristik. Rechts: Anwendung der Höhenmaximierungsheuristik.

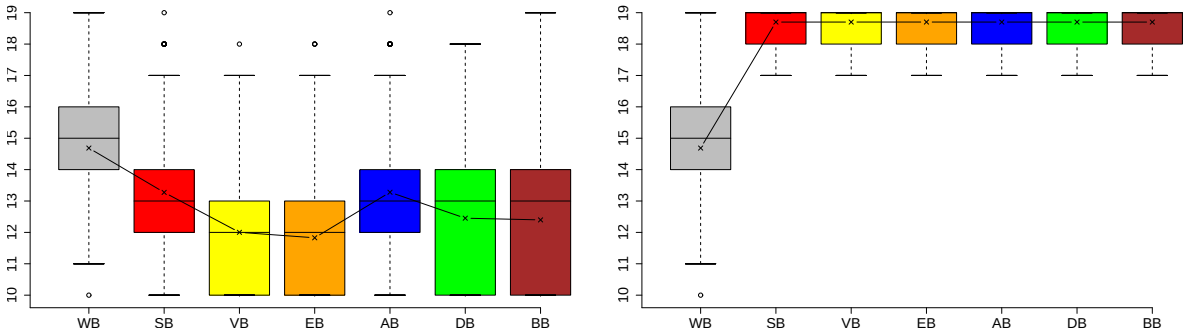


Abbildung 24: Die x-Achse gibt die Breitensuchvariante und die y-Achse die Baumhöhe an. LEDA-Graph mit 10000 Knoten und 25000 Kanten. Links: Anwendung der Höhenminimierungsheuristik. Rechts: Anwendung der Höhenmaximierungsheuristik.

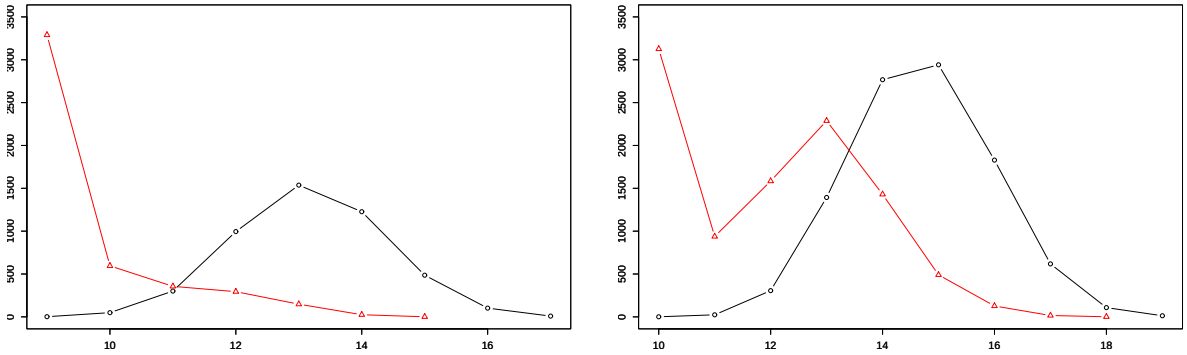


Abbildung 25: Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit an. Links: LEDA-Graph mit 5000 Knoten und 12500 Kanten. Rechts: LEDA-Graph mit 10000 Knoten und 25000 Kanten. $\circ$ =Ohne Optimierung, $\triangle$ mit Optimierung EB.

Straßengraphen Straßengraphen sind Graphen, die aus realen Straßendaten von Städten gebildet werden, in dem Straßen durch Polygonzüge ersetzt werden. In jedem Schnittpunkt von Kanten wird ein neuer Knoten eingefügt, so dass der daraus resultierende Graph planar wird. Die neu eingefügten Knoten führen zu keiner Veränderung der Struktur des Ursprungsgraphen. Wir betrachten in den Tests Straßengraphen von Europa und der USA (Umgebung von San Francisco). Die Straßengraphen der USA unterscheiden sich hinsichtlich ihrer Struktur von den Europagraphen dadurch, dass bei Straßengraphen von US-Städten deutlich mehr Knoten mit einem Knotengrad 2 existieren.

Höhenminimierung: Die größte Verbesserung konnte bei Straßenraphen erzielt werden. In den meisten Fällen kann man beobachten, dass das optimierte Höhenspektrum sich um fast der Hälfte des nicht optimierten Falles erniedrigt hat. Insbesondere wird der Radius des Graphen sehr oft angenommen und sehr große Höhen kommen nicht mehr vor.

Wenn wir den Straßengraph Karlsruhe mit 25539 Knoten und 32615 Kanten betrachten, so sehen wir, dass hier der Radius im optimierten Fall ungefähr über 12000 mal angenommen wird und das Höhenspektrum sich von 300 (310-600) auf 40 (310-350) verkleinert hat. Es ist also sehr wahrscheinlich, dass mit der Höheminimierungsheuristik ein niedriger Baum konstruiert wird. Bei den Straßengraphen von San Francisco konnte das gleiche Ergebnis beobachtet werden. Die unteren Diagramme liefern eine Veranschaulichung des Sachverhalts.

Höhenmaximierung: Die Heuristik zur Höhenmaximierung liefert nach wie vor auch für diese Graphklasse erstaunlich gute Ergebnisse. Es wird fast immer ein höchster Breitensuchbaum gefunden. Dabei scheint für dieses Verfahren auch bei größeren Graphen 10 Durchläufe vollkommen zu genügen. Von diesem Ergebnis kann man sich auch anhand der folgenden Diagramme überzeugen lassen.

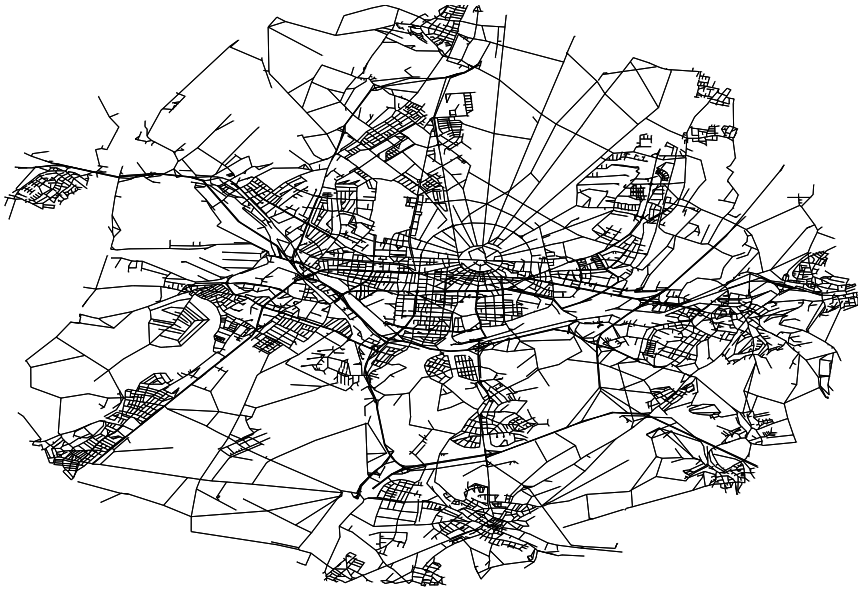


Abbildung 26: Universitätstadt Karlsruhe und Umgebung.

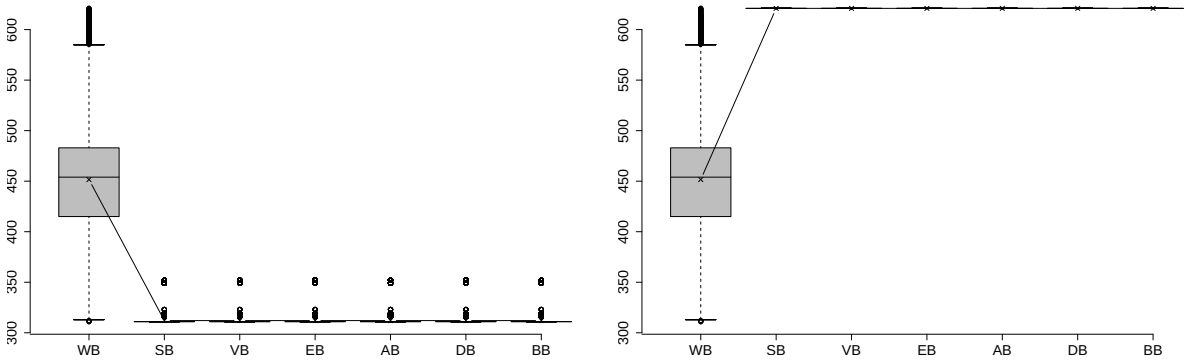


Abbildung 27: Die x-Achse gibt die Breitensuchvariante und die y-Achse die Baumhöhe an. Straßengraph mcity03. Links: Anwendung der Höhenminimierungsheuristik. Rechts: Anwendung der Höhenmaximierungsheuristik.

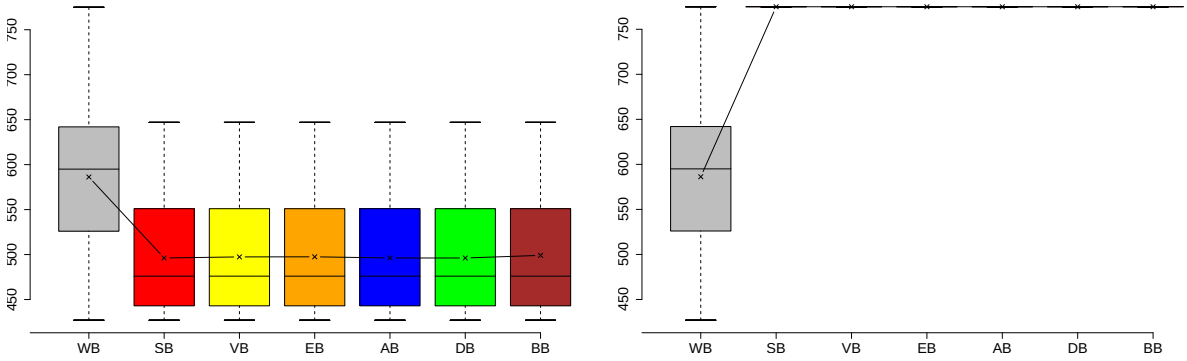


Abbildung 28: Die x-Achse gibt die Breitensuchvariante und die y-Achse die Baumhöhe an. Straßengraph mcity04. Links: Anwendung der Höhenminimierungsheuristik. Rechts: Anwendung der Höhenmaximierungsheuristik.

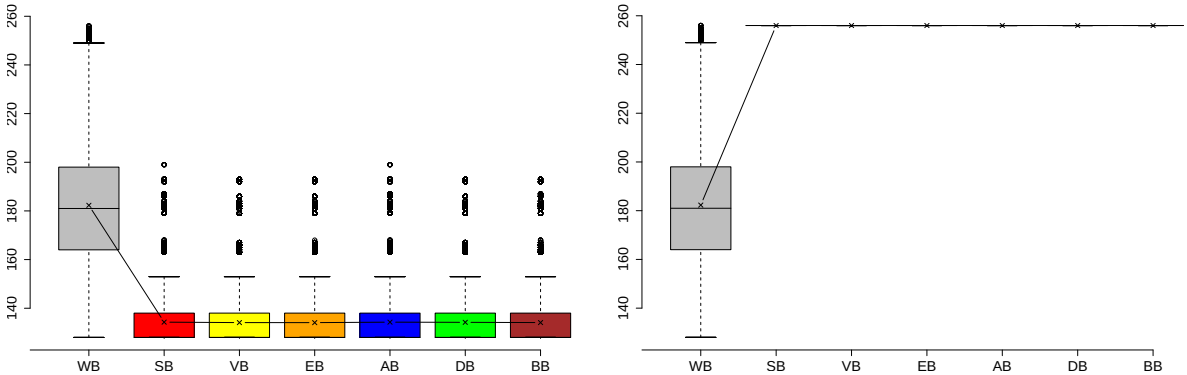


Abbildung 29: Die x-Achse gibt die Breitensuchvariante und die y-Achse die Baumhöhe an. Straßengraph Karlsruhe. Links: Anwendung der Höhenminimierungsheuristik. Rechts: Anwendung der Höhenmaximierungsheuristik.

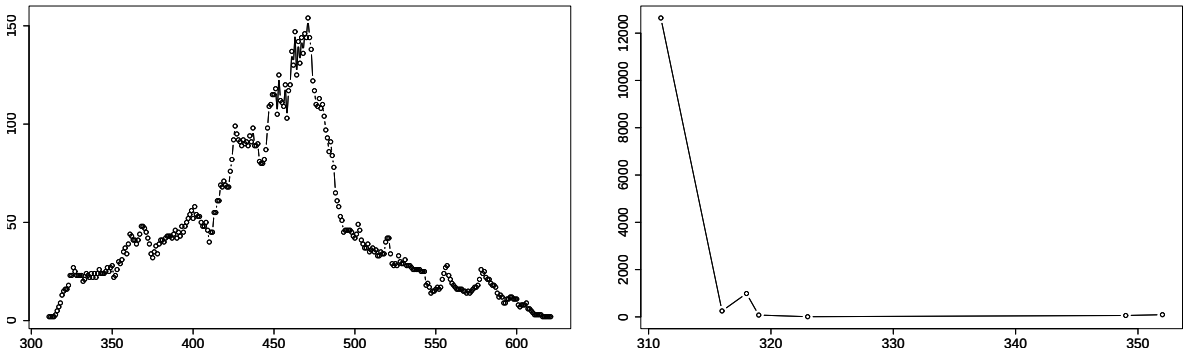


Abbildung 30: Straßengraph aus der Umgebung von San Francisco (Bezeichnung mcity03) mit 14122 Knoten und 14986 Kanten. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit an. Links: Ohne Optimierung. Rechts: Mit der Optimierung EB.

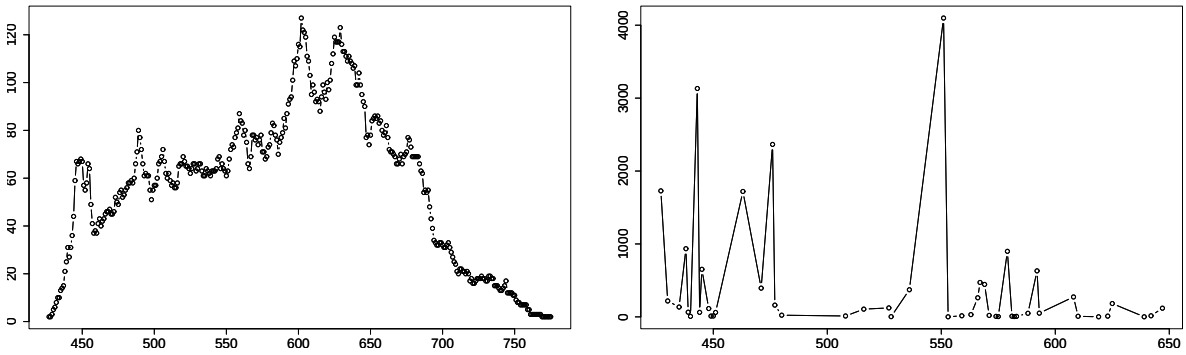


Abbildung 31: Straßengraph aus der Umgebung von San Francisco (Bezeichnung mcity04) mit 20025 Knoten und 20728 Kanten. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit an. Links: Ohne Optimierung. Rechts: Mit der Optimierung EB.

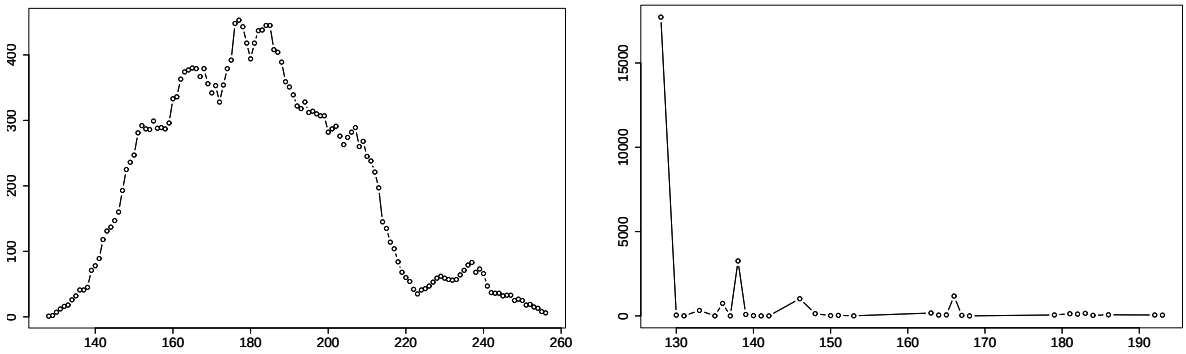


Abbildung 32: Straßengraph aus der Umgebung von Karlsruhe mit 25539 Knoten und 32615 Kanten. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit an. Links: Ohne Optimierung. Rechts: Mit der Optimierung EB.

2.4.6 Zusammenfassung

Höhenminimierung Anhand der Tests konnte man beobachten, dass die permutierenden Breitensuchen bei der Höhenminimierung ein besseres Ergebnis erbracht haben als die anderen Breitensuchvarianten. Die balancierte Breitensuche jedoch, hat im Vergleich am schlechtesten abgeschnitten. Bei Delaunay-Graphen konnte mit keiner Breitensuchvariante eine deutliche Verbesserung erzielt werden, was auf die spezielle Struktur dieser Graphklasse zurückzuführen ist. Für alle anderen Graphklassen, die wir betrachtet haben, konnte eine Verbesserung erzielt werden.

Höhenmaximierung Für die Höhenmaximierung haben alle Breitensuchvarianten fast gleich gute Ergebnisse geliefert, nämlich eine Höhe nahe dem Durchmesser des Graphen. Daher scheint es nicht von Belang zu sein, welche Breitensuche man in dieser Heuristik verwendet. Insbesondere funktioniert die Höhenmaximierung für alle Graphklassen recht gut und man kann in einigen wenigen Iterationen einen hohen Breitensuchbaum finden.

Gesamtergebnisse In den folgenden Tabellen steht EW für den Erwartungswert. Desweiteren sind die Graphklassen der Graphen durch ihren Präfix gekennzeichnet und tragen als ihren Index die Information über ihre Knotenanzahl. Die Anwendung der Höhenminimierungsheuristik kennzeichnen wir mit Min und die Maximierungsheuristik mit Max. Die Art der Breitensuche ist dabei die kantenpermutierende Breitensuche EB.

Graph	Knotenanazahl	Kantenanzahl
Diameter_1000	3001	8997
Rect_80_20	1600	3100
Rect_40_40	1600	3120
Rect_100_100	10000	19800
LEDA_5000	5000	12500
LEDA_10000	10000	25000
Delaunay_5000	5000	12500
Delaunay_10000	10000	25000
mcity03	14122	14986
mcity04	20025	20728
Karlsruhe	25539	32615

Graph	rad	diam	$\max(\text{EB}_{\text{Min}})$	$\min(\text{EB}_{\text{Max}})$	EW(WB)	EW(EB_{Min})	EW(EB_{Max})
Diameter_100	50	100	50	100	75.083	50	100
Rect_80_20	50	98	89	98	74	57.801	98
Rect_40_40	40	78	59	78	59	48.633	78
Rect_100_100	100	198	149	198	149	123.654	198
LEDA_5000	9	17	15	17	13.167	10.566	17
LEDA_10000	10	19	19	17	14.686	13.278	18.702
Delaunay_5000	35	43	42	38	38.725	38.262	41.289
Delaunay_10000	44	54	54	51	50.292	49.677	53.144
mcity03	311	621	352	621	451.595	312.069	621
mcity04	427	775	647	775	586.252	496.213	775
Karlsruhe	128	256	199	256	183.299	134.281	256

3 Triangulierung

In der komplexen Phase vom *Planar-Separator-Theorem* wird ein triangulierter Graph benötigt. Damit die Gesamtlaufzeit des Theorems nicht durch die Laufzeit des Triangulierungsalgorithmus beeinflusst wird, sollte dieser lineare Laufzeit haben. Der bisherige Triangulierungsalgorithmus ist ein topologischer statt ein geometrischer und wurde von Christian Uhrig und Torsten Hagerup vorgeschlagen [8]. Die Laufzeit des Algorithmus beträgt $O(n + m)$, wobei mit n die Knotenanzahl und m die Kantenanzahl gemeint ist. Dabei iteriert man über die Knoten des Graphen und trianguliert für jeden Knoten seine inzidenten Facetten. Welche planare Einbettung hierbei betrachtet wird, hängt von der internen Speicherung der Kanten des Graphen in den jeweiligen Datenstrukturen ab. Die Triangulierung der inzidenten Facetten beruht auf der folgenden Beobachtung.

Gegeben sei ein einfacher ungerichteter zusammenhängender Graph $G = (V, E)$. Desweiteren seien $x_1, \ldots, x_n$ Knoten aus V und $f = (x_1, x_2, x_3, x_4, x_n)$ eine zu x_1 inzidente Facette des Graphen. Falls die Kante $\{x_1, x_3\}$ nicht existiert, so kann man diese Kante hinzufügen und der Graph bleibt planar. Aus der Facette f entstehen somit zwei neue Facetten $f_1 = (x_1, x_2, x_3)$ und $f_2 = (x_1, x_3, \ldots, x_n)$. Die Facette f_1 bildet ein Dreieck und ist damit trianguliert und muss nicht weiter betrachtet werden. Falls die Kante $\{x_1, x_3\}$ aber bereits existiert, so wissen wir, dass die Kante $\{x_2, x_4\}$ nicht existieren kann, da sonst f keine inzidente Facette zu x_1 wäre. Also können wir die Kante $\{x_2, x_4\}$ hinzufügen, ohne die Planarität des Graphen zu verletzen. Dadurch entstehen aus f ebenfalls zwei neue Facetten $f_1 = (x_2, x_3, x_4)$ und $f_2 = (x_1, x_2, x_4, \ldots, x_n)$. Dementsprechend macht man mit der Facette f_2 weiter und erhält letztendlich nur noch eine Facette mit drei umrandenden Kanten. Dieses Verfahren wendet man für die indzidenten Facetten aller Knoten an bis man eine Kantenanzahl von $3n - 6$ erreicht hat. In welcher Reihenfolge die Knoten jedoch abgearbeitet werden ist nicht festgelegt und wir können diese Tatsache, wie wir später noch sehen werden, ausnutzen.

Insgesamt ergibt sich mit dem Satz von Euler über die Kantenanzahl eines planaren Graphen (Siehe Anhang) ein Triangulierungsalgorithmus mit linearer Laufzeit $O(n)$.

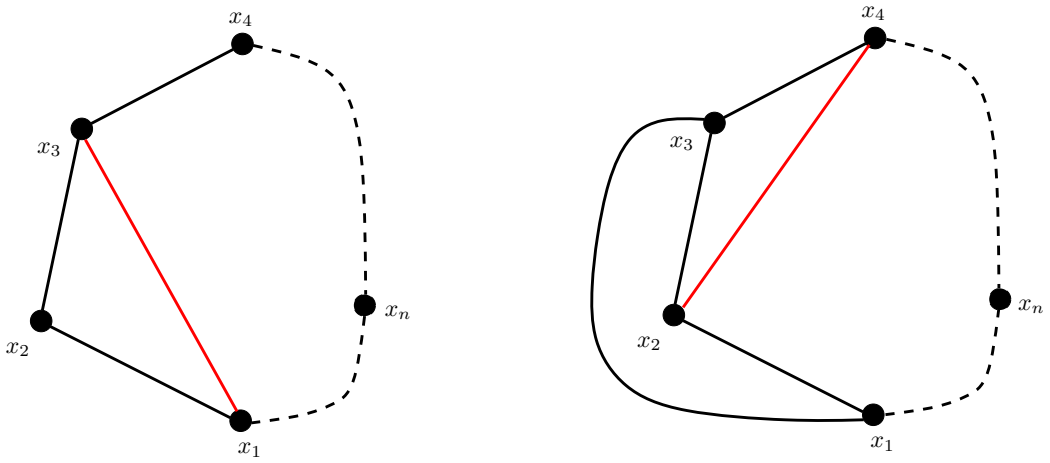


Abbildung 33: Bei der Triangulierung der inzidenten Facette f von x_1 kann man einen der beiden Kanten $\{x_2, x_4\}$ oder $\{x_1, x_3\}$ hinzufügen, ohne die Planarität des Graphen zu verletzen.

3.1 Triangulierende Breitensuche

Wir möchten den vorgestellten Triangulierungsansatz nutzen, um uns einen Breitensuchbaum minimaler Höhe für die komplexe Separationsphase zu konstruieren. Da die Triangulierung einer gegebenen Einbettung eines planaren Graphen direkten Einfluss auf den Radius und Durchmesser des Graphen hat und die Breitensuchhöhe eines Baumes genau zwischen diesen Werten liegt, kann es von Vorteil sein, die Triangulierung bei der Konstruktion von Bäumen mit einzubeziehen. Hierfür benutzen wir den gleichen Ansatz, wie schon in der Einleitung erwähnt, nur dass wir nicht mehr in irgendeiner Reihenfolge die inzidenten Facetten der Knoten des Graphen triangulieren, sondern in einer Reihenfolge, die dem eines Breitensuchbaums entspricht.

Gegeben sei der Graph $G = (V, E)$ mit einer festen Einbettung und ein Knoten x_1 aus V, von dem wir eine triangulierende Breitensuche konstruieren möchten. Desweiteren sei Q eine Warteschlange, die die Abarbeitungsreihenfolge der Knoten bestimmt, deren inzidenten Facetten wir triangulieren. Zuerst fügen wir alle Nachbarknoten von v zu Q hinzu, die noch nicht markiert sind und betrachten danach alle inzidenten Facetten von v. Sei $f = (x_1, x_2, \ldots, x_n)$ einer dieser Facetten, die wir triangulieren möchten. Also gehen wir alle Knoten x_k mit k aus $\{2, \ldots, n-1\}$ durch und fügen, falls die Triangulierung dieser Facette nach dem vorgestellten Ansatz es erlaubt, die Kante $\{x_1, x_k\}$ hinzu und speichern x_k in Q, falls es nicht bereits als besucht markiert ist. Für alle Kanten $\{x_i, x_j\}$ mit i, j aus $\{2, \ldots, n\}$, die bei der Triangulierung dieser Facette hinzugefügt werden könnten, gibt es zwei Möglichkeiten. Entweder wird diese Kante zum Graphen hinzugefügt (Variante TB1) oder wir lassen sie weg (Variante TB2). In beiden Fällen jedoch werden die Knoten x_i und x_j nicht zu Q hinzugefügt. Es könnte aber sein, dass diese bereits in Q eingefügt worden sind, da sie eine inzidente Kante zu x_1 hatten. Nach dem wir alle inzidenten $Facetten$ von x_1 abgearbeitet haben, machen wir mit dem vordersten Knoten von Q weiter. Das Verfahren endet, nach dem alle inzidenten Facetten aller Knoten trianguliert sind.

Es entstehen also zwei Varianten der triangulierenden Breitensuche, die beide einen Baum der gleichen Höhe liefern, aber die daraus resultierenden triangulierten Graphen unterschiedlich sind. Da unsere Heuristiken auf unterschiedlichen Graphen angewendet verschiedene Ergebnisse liefern können, werden hier beide Varianten betrachtet.

Lemma 3.1 *Gegeben sei ein planarer Graph $G = (V, E)$ und ein Knoten v aus V. Weiter sei f eine Facette von G. Bei der triangulierenden Breitensche TB1 vom Wurzelknoten w aus gilt für alle Knoten u, v auf der Facette f die folgende Bedingung:*

$$|\mathit{dist}(w, u) - \mathit{dist}(w, v)| \leq 1.$$

Beweis: Ohne Beschränkung der Allgemeinheit sei $f = (x_1, \ldots, x_n)$ mit x_i aus V für i aus $\{1, \ldots, n\}$ und x_1 der Knoten, der f als inzidente Facette vor allen anderen Knoten abarbeitet und trianguliert. Das heißt, x_1 steht an vorderster Stelle in Q unter allen Knoten, die f als inzidente Facette haben. Für alle Knoten t aus $\{x_2, \ldots, x_n\}$ gilt:

$$\text{dist}(w, t) \geq \text{dist}(w, x_1).$$

Denn würde es nicht so sein, so würde ein Knoten wegen der Breitensucheigenschaft vor x_1 in Q existieren, der auf der Facette f liegt und aus dem die Facette f bereits trianguliert worden wäre. Andererseits kann wegen der Breitensucheigenschaften kein markierter Knoten z aus $\{x_2, \ldots, x_n\}$

einen Abstand zur Wurzel dist(w, z) größer als dist$(w, x_1) + 1$ haben, da sonst die inzidenten Facetten zu x_1 schon bearbeitet worden wären. Also gilt für alle markierten Knoten z auf der Facette f: $|\text{dist}(w, z) - \text{dist}(w, x_1)| \leq 1$. Nach der Triangulierung von f existiert zu jedem Knoten h aus f eine Kante $\{x_1, h\}$. Insbesondere für die noch nicht markierten Knoten h bedeutet das:

$$\text{dist}(w, h) = \text{dist}(w, x_1) + 1.$$

Somit ist die Behauptung für alle Knoten auf der Facette f bewiesen. □

Lemma 3.2 *Die triangulierende Breitensuche in der Variante TB1 liefert einen genau so hohen Baum wie die Variante TB2.*

Beweis: Seien G, x_1 und f wie im vorherigen Satz gewählt. Mit G_1 bzw. G_2 bezeichnen wir jeweils die Graphen, die aus G nach der Abarbeitung von x_1 mit der Variante TB1 bzw. TB2 entstanden sind. Es gilt offensichtlich $G_1 \subseteq G_2$ (Siehe Konstruktion). Ohne Einschränkung sei bei der Abarbeitung von x_1 die erste Kante $e = \{u, v\}$ in G_2 hinzugekommen, die nicht in G_1 enthalten ist. Die Kante e verbindet zwei Knoten auf der Facette f. Nach dem Satz 3.1 gilt damit:

$$|\text{dist}(w, u) - \text{dist}(w, v)| \leq 1.$$

Somit wurde diese Kante, im schlimmstem Fall, zwischen zwei aufeinander folgenden Level des Breitensuchbaums gezogen und ändert daher nichts an der Höhe des Breitensuchbaums. □

Aus dem Lemma 3.1 läßt sich noch eine weitere interessante Feststellung machen, die den Einsatz des Verfahrens in der komplexen Phase des *Planar-Separator-Theorems* noch stärker motiviert. Man kann nämlich zeigen, dass man einen planaren zusammenhängenden Graphen mit einer gegebenen Einbettung und einem fest gewählten Knoten w in einer Art und Weise triangulieren kann, dass der Breitensuchbaum mit w als Wurzel, eine minimale Höhe als Baum unter allen Möglichen Triangulierungen hat, die für den Graphen existieren. Wir werden diese Feststellung als Satz formulieren und beweisen. Im Folgenden bezeichnen wir mit B_{w_T} einen Breitensuchbaum mit der Wurzel w in der Triangulierung T des Graphen G.

Definition 3.1 *Eine Triangulierung T eines einfachen zusammenhängenden Graphen $G = (V, E)$ heißt optimal für einen Knoten $w \in V$, falls für alle anderen Triangulierungen $\acute{T}$ des Graphen gilt:*

$$\textit{Höhe}(B_{w_T}) \leq \textit{Höhe}(B_{w_{\acute{T}}}).$$

Satz 3.1 *Gegeben sei ein einfacher zusammenhängender ungerichteter planarer Graph $G = (V, E)$ mit einer Einbettung und ein Knoten w aus V. Unter allen Breitensuchbäumen, die man aus w in einer beliebigen Triangulierung dieser Einbettung erzeugen kann, liefert die triangulierende Breitensuche die Triangulierung, für die ein Breitensuchbaum aus w eine minimale Höhe hat.*

Beweis: Man nehme an, wir hätten eine optimale Triangulierung zu einem Knoten w und einen dazugehörigen Breitensuchbaum T_{opt} mit w als Wurzel, der eine niedrigere Baumhöhe als der Baum T_{tb} hat, der durch eine triangulierende Breitensuche aus w entstanden ist. Sei $e = \{u, v\}$ eine Kante, die durch die optimale Triangulierung hinzugekommen ist und für die gilt:

$$\text{dist}_{T_{\text{opt}}}(w, u) = \text{dist}_{T_{\text{tb}}}(w, u) \quad (4)$$

$$\text{dist}_{T_{\text{opt}}}(w, v) < \text{dist}_{T_{\text{tb}}}(w, v) \quad (5)$$

$$\text{dist}_{T_{\text{opt}}}(w, v) = \text{dist}_{T_{\text{opt}}}(w, u) + 1 \quad (6)$$

Eine solche Kante muss existieren, da sonst die Breitensuchbäume T_{opt} und T_{tb} die gleiche Höhe hätten. Diese Kante verbindet zwei Knoten einer Facette f von G und ist nicht in E enthalten. Denn wäre sie in E enthalten, so würde $\text{dist}_{T_{\text{opt}}}(w, v)$ gleich $\text{dist}_{T_{\text{tb}}}(w, v)$ sein, was der Gleichung (5) widersprechen würde. Damit ergibt sich insgesamt die folgende Berechnung:

$$\text{dist}_{T_{\text{tb}}}(w, v) > \text{dist}_{T_{\text{opt}}}(w, v) = \text{dist}_{T_{\text{opt}}}(w, u) + 1 = \text{dist}_{T_{\text{tb}}}(w, u) + 1.$$

Das heißt, die Knoten $u,\ v$ aus V liegen auf einer Facette f und dennoch gilt für sie:

$$\text{dist}_{T_{\text{tb}}}(w, v) - \text{dist}_{T_{\text{tb}}}(w, u) \geq 2.$$

Dies ist jedoch ein Widerspruch zum Lemma 3.1. Also ist die Annahme falsch und T_{tb} ist ein optimaler Breitensuchbaum. □

Im Folgenden werden wir anhand par Beispielen sehen, welche Auswirkungen die Triangulierung auf Distanzen zwischen Knoten hat, die die Höhen von Breitensuchbäumen beeinflussen können. In der Abbildung 34 betrachten wir die Distanz zwischen den Knoten x_1 und x_2 bei einer Triangulierung des Graphen nach einer zufällig gewählten Abarbeitungsreihenfolge der Knoten. Diese Distanz beträgt 3. Wenn wir aber eine triangulierende Breitensuche beginnend aus dem Knoten x_1 starten würden, wie es in der Abbildung 35 zu sehen ist, so verringert sich die Distanz zwischen den beiden Knoten auf 2. Diese Distanz ist gleichzeitig auch der kleinste annehmbare Wert bei der Betrachtung aller Triangulierungen dieser festen Einbettung des Graphen. Die Triangulierung der äußeren Facette brauchen wir für dieses Beispiel nicht zu betrachten, da die Triangulierung der äußeren Facette hier keinen Einfluss auf die Distanz der Knoten x_1 und x_2 hat. Man muß aber immer daran denken, dass alle diese Behauptungen für eine feste Einbettung gelten. Bei der Triangulierung einer anderen Einbettung des Graphen könnte sich die Distanz der Knoten wieder um einiges verringern oder erhöhen. Man hat also auch die freie Wahl bei der Betrachtung der Einbettungen. Mit der Wahl einer geschickten Einbettung werden wir uns in dieser Arbeit aber nicht näher befassen, da dies eine Studie für sich ist.

Ein anderer Aspekt den wir motivieren möchten, ist die Minimierung des Radiuses eines planaren Graphen. Wir wissen aus den vorausgegangenen Kapiteln, dass in einem Graphen ein Breitensuchbaum minimaler Höhe, eine Höhe gleich dem Radius hat. Es stellt sich nun die Frage, wie man eine gegebene Einbettung eines planaren Graphen triangulieren muss, damit der daraus resultierende Graph den kleinstmöglichen Radius unter allen möglichen Triangulierungen hat.

Lemma 3.3 *Wenn man von jedem Knoten eines planaren zusammenhängenden Graphen $G = (V, E)$ eine triangulierende Breitensuche bildet, so liefert diese Breitensuche für den Knoten aus dem ein Breitensuchbaum minimaler Höhe zustande gekommen ist, eine optimale Triangulierung der Einbettung hinsichtlich der Minimalität des Radiuses.*

Beweis: Der Beweis dieses Lemmas folgt direkt aus dem Lemma 3.1. Dieser sagt aus, dass man durch die triangulierende Breitensuche aus einem Knoten eine Triangulierung der Einbettung bekommt, in der die Breitensuche minimale Höhe hat. Außerdem wissen wir, dass in jedem Graphen ein Knoten existiert, dessen Höhe gleich dem Radius ist. Wir betrachten eine beliebige Triangulierung des Graphen $G = (V, E)$, die einen minimalen Radius verursacht. Den triangulierten Graphen nennen wir $G' = (V, E')$. In G' existiert ein Knoten w aus dem ein Breitensuchbaum als Höhe den Radius annimmt. Wenn wir nun aus v eine triangulierende Breitensuche in G konstruieren, so hat der daraus resultierende Breitensuchbaum eine Höhe gleich dem minimalen Radius. □

Das Lemma 3.3 motiviert eine Heuristik, in der wir versuchen eine geeignete Triangulierung und einen geeigneten Baum gleichzeitig zu finden. Diese Heuristik wird im nächsten Abschnitt vorgestellt.

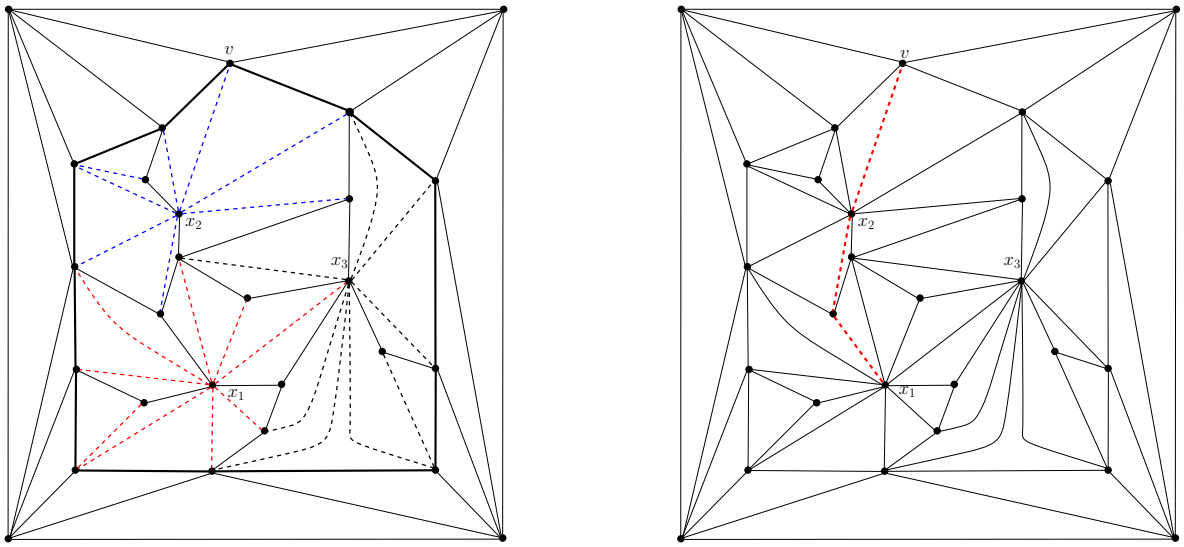


Abbildung 34: Links: Triangulierung des Graphen nach der zufällig gewählten Knotenreihenfolge (x_1, x_2, x_3). Rechts: Der Abstand zwischen x_1 und v ist 3.

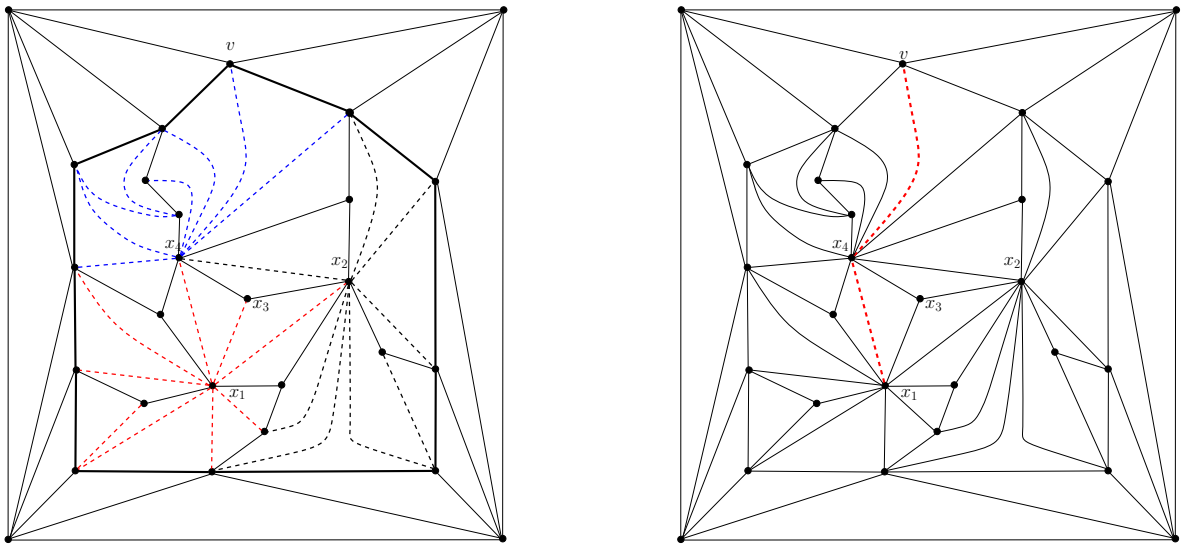


Abbildung 35: Links: Triangulierende Breitensuche aus dem Knoten x_1. Die Abarbeitungsreihenfolge der Knoten ist (x_1, x_2, x_3, x_4). Rechts: Der Abstand zwischen x_1 und v beträgt 2.

3.2 Heuristik

Nach dem wir festgestellt haben, dass die Triangulierung des Graphen einen direkten Einfluss auf das Höhenspektrum der Breitensuchbäume hat, möchten wir nun die Triangulierung bei der Suche eines Breitensuchbaums mit minimaler Höhe mit einbeziehen. Wir kombinieren die Höhenminimierungsheuristik mit der Triangulierung, in dem wir nicht mehr eine gewöhnliche Breitensuche machen, sondern eine triangulierende.

Algorithmus 7 : Höhenminimierungsheuristik mit der triangulierenden Breitensuche.

Eingabe : Ein feste planare Einbettung eines einfachen zusammenhängenden Graphen $G = (V, E)$ mit einem Startknoten w aus V und Parameter i, der die Anzahl der Durchläufe in der Heuristik bestimmt.

Ausgabe : Knoten $z \in V$ aus dem ein Breitensuchbaum mit einer geringen Höhe konstruiert werden kann.

1 Starte eine triangulierende Breitensuche T mit w als Wurzel.
2 Speichere alle Kanten, die durch die Triangulation hinzugekommen sind in eine Liste L. Bestimme die Menge der Zentrumsknoten Z von T und wähle einen Knoten z aus Z.
3 Lösche alle Kanten aus dem Graphen, die in L enthalten sind und erniedrige i.
4 Falls i größer als Null ist, so setze w auf z und springe zu Schritt 1. Ansonsten gib z zurück.

Jetzt müssen wir nur noch zeigen, dass nach Anwendung der Heuristik, ein Breitensuchbaum aus dem Knoten z eine niedrigere oder höchstens gleiche Höhe wie ein Breitensuchbaum aus dem Startknoten w hat. Diese Behauptung wird im folgenden Lemma bewiesen.

Lemma 3.4 *Sei T_w der Breitensuchbaum, der aus dem Startknoten w und T_z ein Breitensuchbaum, der aus dem Ausgabeknoten z gebildet wurde. Es gilt dann die folgende Ungleichung für die Baumhöhen:*

$$\textit{Höhe}(T_z) \leq \textit{Höhe}(T_w).$$

Beweis: Sei G_w der Graph der aus G bei der Konstruktion von T_w entstanden ist und z_1 ein Knoten aus $Z(T_w)$. Ein Breitensuchbaum B_{z_1} mit z_1 als Wurzel hat in G_w bereits eine Höhe kleiner gleich der Höhe von T_w. Wenn wir nun eine triangulierende Breitensuche mit z_1 als Wurzel konstruieren, so ist dieser, nach dem Lemma 3.1, höchstens gleich Höhe$(B_{z_1}) \leq$ Höhe(T_w). Nun können wir diese Argumentation induktiv mit z_1 als neues w fortsetzen und erhalten somit die Behauptung. □

3.2.1 Experimente

Um zu untersuchen, wie gut die aufgestellte Heuristik mit der triangulierende Breitensuche funktioniert, wurde sie mit der Höhenminimierungsheuristik und dem nicht optimierten Fall verglichen. Damit dieser Vergleich auch fair bleibt, hat man auf einer planaren festen Einbettung eines Graphen die Experimente mit einigen Graphklassen durchgeführt.

Freiheitsgrade Bei der Verwendung dieser Heuristik steht man vor dem Problem, den Parameter i für die Anzahl der Durchläufe geeignet zu wählen. Desweiteren muss noch untersucht werden, ob die Anwendung der Varianten TB_1 und TB_2 unterschiedliche Ergebnisse liefern oder ob es nicht darauf ankommt, welche Variante man verwendet.

Testreihe Zuerst hat man eine zufällige Triangulierung des Graphen betrachtet und danach die Höhenminimierungsheuristik und den ursprünglichen nicht optimierten Ansatz für diesen Graphen getestet. Anschließend hat man für den gleichen Graphen, der noch nicht trianguliert ist, mit der selben Einbettung von jedem Knoten aus mit der Triangulierungsheuristik einen geeigneten Knoten gefunden und dessen Baumhöhe gemessen.

Für alle Verfahren bis auf die nicht optimierte Variante (WB) wurde der Parameter für die Anzahl der Durchläufe der Heuristik auf 10 gesetzt. Außerdem wurde jeder Knoten des Graphen genau einmal als Startknoten der Heuristik ausgewählt. Dann wurde anhand dieser Daten der Erwartungswert und der Median der Baumhöhen für das jeweilige Verfahren bestimmt.

Testgraphen Für die Experimente hat man alle Graphklassen außer den Diametergraphen auch in diesen Experimenten verwendet. Die Diametergraphen haben wir ausgelassen, da diese bereits trianguliert sind und die triangulierende Breitensuche somit die gleiche Wirkung wie eine normale Breitensuche hätte .

Diagramme Es sind einige interessante Ergebnisse hinsichtlich der Breitensuchbaumhöhen für verschiedene Klassen von Graphen entstanden, die wir im Folgenden hauptsächlich, wie bereits bei den vorherigen Untersuchungen, anhand von Boxplots präsentieren. Die Erwartungswerte der Baumhöhen der verschiedenen Verfahren haben wir als Kreuze eingezeichnet und mit einem Linienzug miteinander verbunden. Es sind auch einige Diagramme dabei, in denen Treppenfunktionen zu sehen sind, die die akkumulierten Wahrscheinlichkeiten angeben, eine Baumhöhe kleiner gleich einem bestimmten Wert zu haben. Die akkumulierte Wahrscheinlichkeit für den Wert x bezeichnet dabei die Wahrscheinlichkeit, bei der Wahl eines zufälligen Wurzelknotens unter allen Knoten des Graphen, einen Knoten zu wählen, aus dem sich ein Breitensuchbaum mit einer Höhe kleiner oder gleich x konstruieren läßt.

Gittergraphen Es wurden die gleichen Gittergraphen wie schon zuvor untersucht. Es ist recht interessant zu sehen, dass durch die triangulierende Breitensuche tatsächlich eine Verbesserung erzielt werden konnte.

Ein anderes Ergebnis welches man beobachten kann, ist die Tatsache, dass die Höhenminimierungsheuristik u.U. ein noch besseres Ergebnis geliefert hat, falls die Triangulierung der Einbettung optimal hinsichtlich der Minimalität des Radiuses gewesen ist.

Weiter hat man die akkumulierte Wahrscheinlichkeit, eine kleinere Höhe als einen bestimmten Wert zu haben, bei einem quadratischen Gittergraphen mit 10000 Knoten mit der akkumulierten Wahrscheinlichkeiten der Optimierungsansätze verglichen. Hierfür siehe Abbildung 37.

Ergebnis Desweiteren sieht man aus dem Diagramm 37 auch, dass bei Anwendung der Triangulierungsheuristik mit 100 %-er Wahrscheinlichkeit eine Höhe kleiner gleich 51 zu Stande gekommen ist,

egal welcher Knoten als Wurzel des Baumes gewählt wurde. Somit kann man mit der triangulierenden Breitensuche für Gittergraphen ein deutlich besseres Ergebnis im Vergleich zur Höhenminimierungsheuristik erzielen.

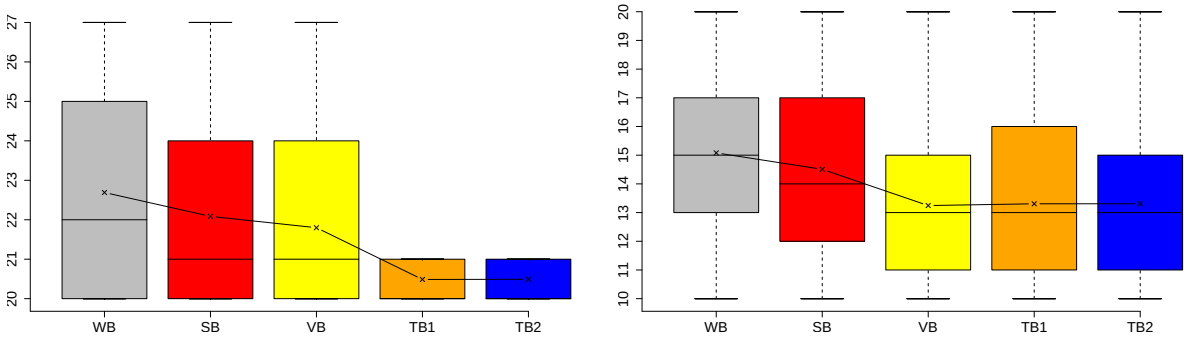


Abbildung 36: Die x-Achse gibt das angewendete Optimierungsverfahren und die y-Achse die Baumhöhe. Links: Quadratisches Gitter mit 1600 Knoten(40×40). Rechts: Rechteckiges Gitter mit 1600 Knoten(80×20).

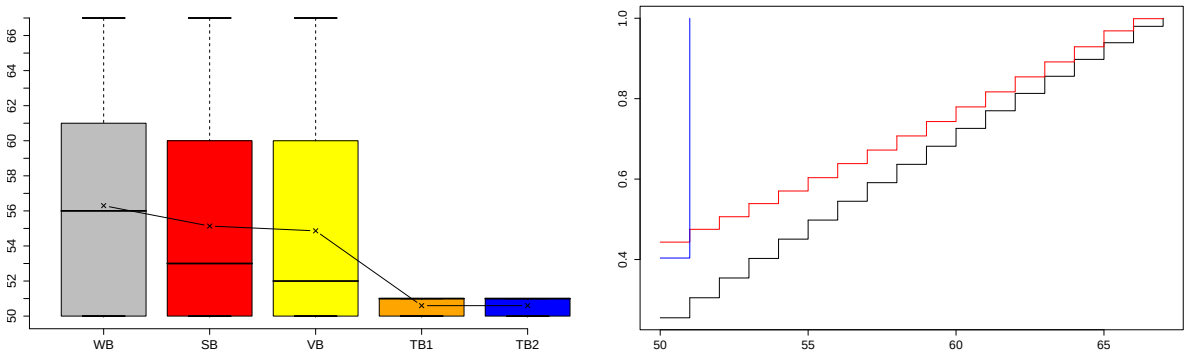


Abbildung 37: Links: Die x-Achse gibt das angewendete Optimierungsverfahren und die y-Achse die Baumhöhe. Quadratisches Gitter mit 10000 Knoten (100×100). Rechts: Vergleich der akkumulierten Wahrscheinlichkeiten. Schwarz: Ohne Optimierung. Rot: Höhenminimierungsheuristik 10 Iterationen. Blau: Triangulierungsheuristik 10 Iterationen.

Delaunay-Graphen Man sieht recht deutlich an den unteren Diagrammen, die man aus Experimentergebnissen mit Delaunay-Graphen unterschiedlicher Knotenzahl estellt hat, dass sich der Radius bei verschiedener Triangulierung der Einbettung eines planaren Graphen ändert. Die Delaunay-Graphen wurden so erzeugt, dass deren Kantenanzahl gleich dem 2.5-fachen der Knotenanzahl entspricht. Daher kommen garantiert neue Kanten durch die Triangulierung hinzu.

Ergebnis Für alle Graphen, die wir betrachtet haben, wurde durch die triangulierende Breitensuche eine Verbesserung erzielt, worauf die Erwartungswerte der Baumhöhen der verschiedenen Verfahren andeuten. Die Varianten TB1 und TB2 liefern aber das gleiche Ergebnis.

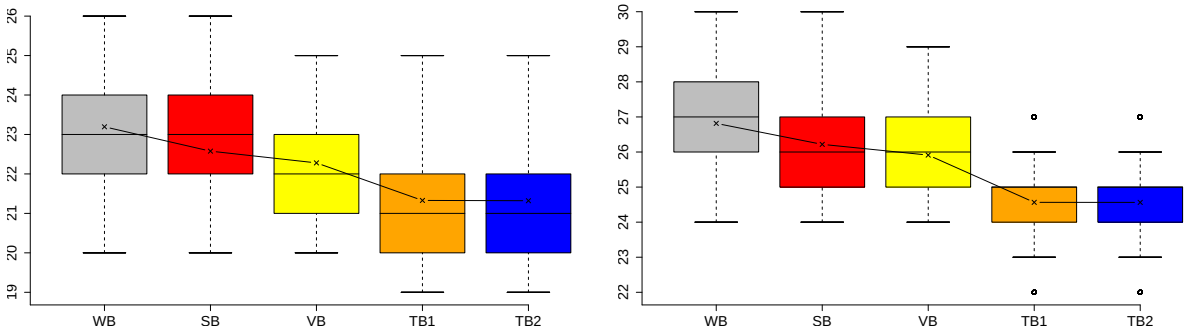


Abbildung 38: Links: Graph mit 3000 Knoten. Rechts: Graph mit 4000 Knoten. Die x-Achse gibt das angewendete Optimierungsverfahren und die y-Achse die Baumhöhe an.

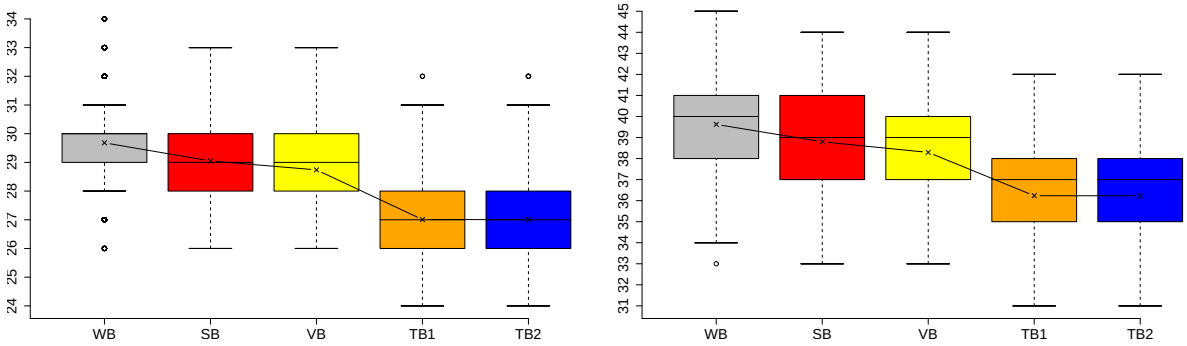


Abbildung 39: Links: Graph mit 5000 Knoten. Rechts: Graph mit 10000 Knoten. Die x-Achse gibt das angewendete Optimierungsverfahren und die y-Achse die Baumhöhe an.

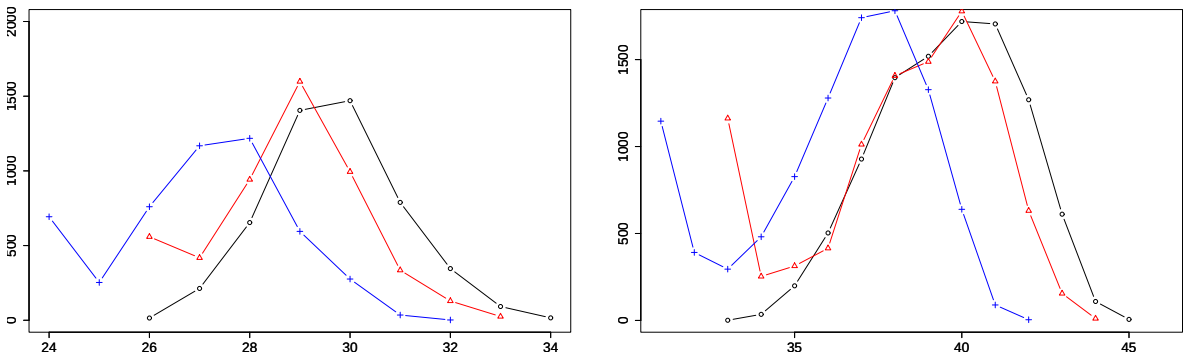


Abbildung 40: Links: Graph mit 5000 Knoten und 12500 Kanten. Rechts: Graph mit 10000 Knoten. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit der Baumhöhe an.($\circ$=WB, $\triangle$=VB, $+$=TB1).

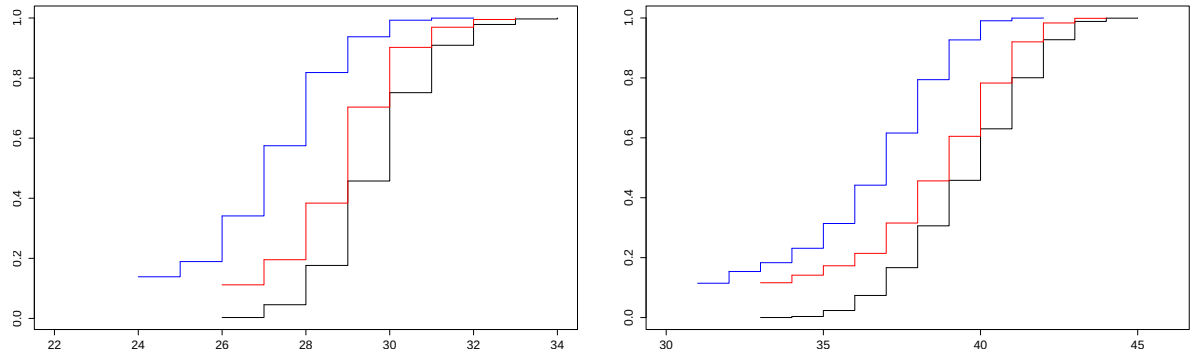


Abbildung 41: Vergleich der akkumulierten Wahrscheinlichkeiten. Schwarz: Ohne Optimierung. Rot: Höhenminimierungsheuristik 10 Iterationen. Blau: Triangulierungsheuristik 10 Iterationen. Links: Delaunay-Graph mit 5000 Knoten. Rechts: Delaunay-Graph mit 10000 Knoten.

LEDA-Graphen Die Höhenminimierungsheuristik liefert bei LEDA-Graphen kein viel schlechteres Ergebnis als bei der Triangulierungsheuristik. Besser gesagt, kann ein schlechteres Ergebnis nur dann anhand der unteren Diagramme beobachtet werden, wenn die Triangulierung in der ursprünglichen Variante (ohne Berücksichtigung der Reihenfolge der Knoten) nicht optimal gewesen ist. Mit der optimalen Triangulierung ist eine Triangulierung gemeint, die für eine feste Einbettung den kleinsten Radius für den Graphen hervorruft.

Für die untersuchten Graphen gilt das Kanten und Knotenverhältnis von 2.5, also genau wie bei den Delaunay-Graphen. Es existiert also auch hier ein kleiner Freiraum für neue Kanten die durch die Triangulierung hinzukommen würden. Eine Vermutung ist, dass bei einer triangulierenden Breitensuche alle neuen Kanten zu Gunsten des Wurzelknotens hinzukommen und deshalb dieser auch mit größerer Wahrscheinlichkeit im Zentrum des Baumes liegt. Diese Vermutung könnte eine Erklärung dafür sein, dass die Höhenminimierungsheuristik fast genau so gute Ergebnisse wie die Triangulierungsheuristik geliefert hat.

Aus der Abbildung 43 (Unten (rechts)) ist insbesondere zu erkennen, dass der kleinste mögliche Radius, der nach einer Triangulierung der Einbettung des Graphen zu Stande kommen kann, fast zu 100%-er Wahrscheinlichkeit mit der Triangulierungsheuristik erreicht wurde, welches ein faszinierendes Ergebnis ist.

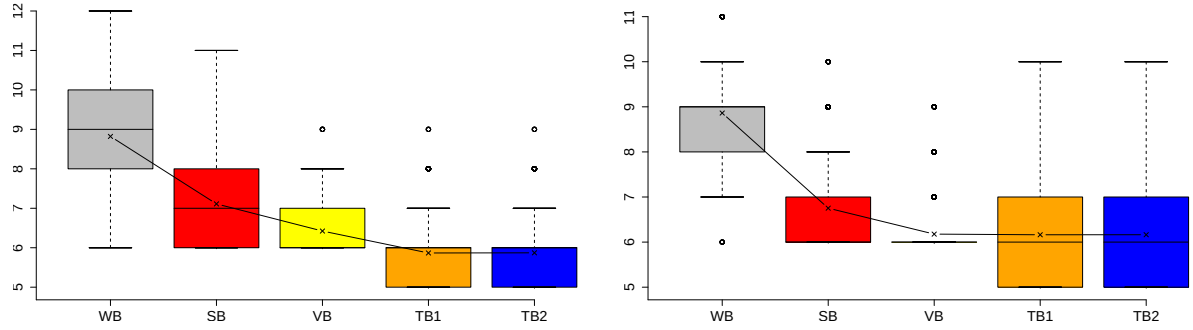


Abbildung 42: Links: Graph mit 1000 Knoten. Rechts: Graph mit 2000 Knoten. Die x-Achse gibt das angewendete Optimierungsverfahren und die y-Achse die Baumhöhe.

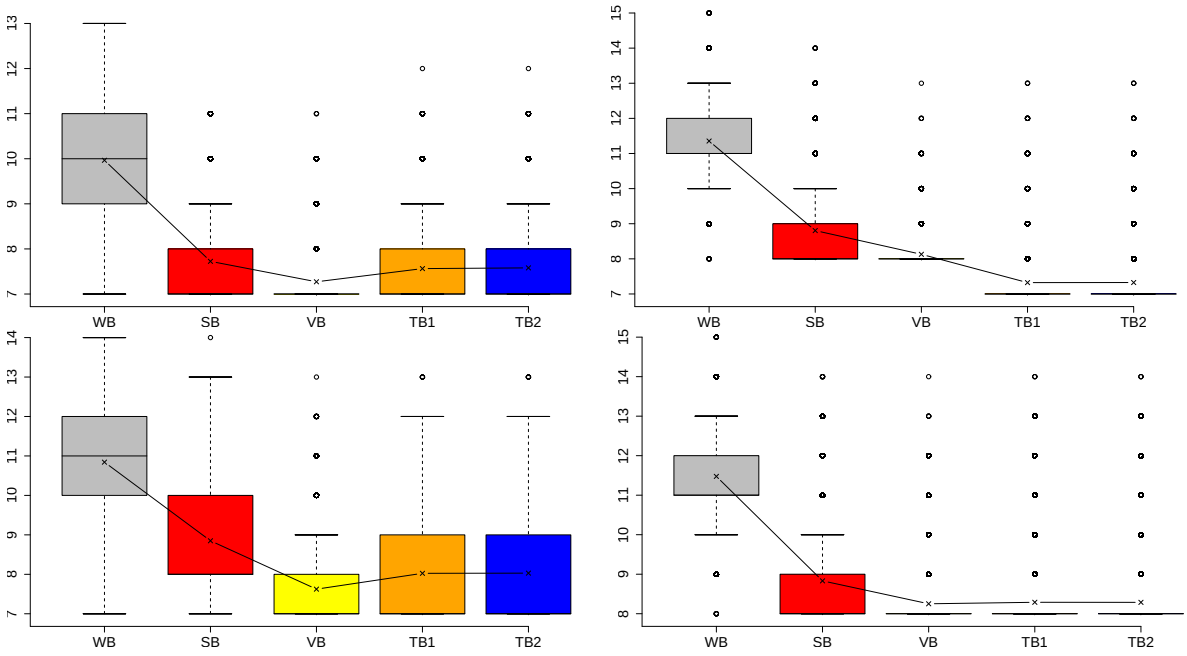


Abbildung 43: Oben (links): Graph mit 3000 Knoten. Oben (rechts): Graph mit 4000 Knoten. Unten (links): Graph mit 5000 Knoten. Unten (rechts): Graph mit 10000 Knoten. Die x-Achse gibt das angewendete Optimierungsverfahren und die y-Achse die Baumhöhe an.

Straßengraphen Bei Straßengraphen hat sich die triangulierende Breitensuche als sehr effektiv erwiesen. Das liegt daran, dass diese Graphen, was die Kantenanzahl betrifft, nicht sehr dicht sind und man deshalb meistens einen größeren Freiraum bei der Triangulierung solcher Graphen ausnutzen kann. Außerdem funktioniert bei dieser Klasse von Graphen bereits die Höhenminimierungsheuristik ziemlich gut, daher würde durch eine Kombination dieser Heuristik mit der Breitensuche ein noch besseres Ergebnis erzielt werden.

Testreihe Wir haben drei Experimente mit Straßengraphen durchgeführt. Zwei davon auf Straßengraphen von US-Städten und den dritten auf den Straßengraphen von Karlsruhe, welches auch Entstehungsort dieser Arbeit ist. Die Experimete haben unsere Vermutung, eine Verbesserung zu erzielen, wie man sich anhand der unteren Diagramme selbst überzeugen kann, bestätigt.

Ein sehr verblüffendes Ergebnis kann man bei der Abbildung 45 sehen. Hier wurde bei einem Straßengraphen mit ungefähr 25000 Knoten fast 15000 Mal der optimale Radius der Einbettung erreicht. Man sieht auch, dass für diese Graphklasse die Triangulierungsheuristik im Test besser abschneidet als die Höhenminimierungsheuristik.

Ergebnis Hier sollte man nochmal erwähnen, dass allgemein die Heuristiken für diese spezielle Graphklasse besonders gute Ergebnisse erbracht haben. Diese Graphklasse findet in der Praxis häufig Verwendung z.B. bei der Berrechnung kürzester Wege für ganze Landstriche und Länder und sonstiges.

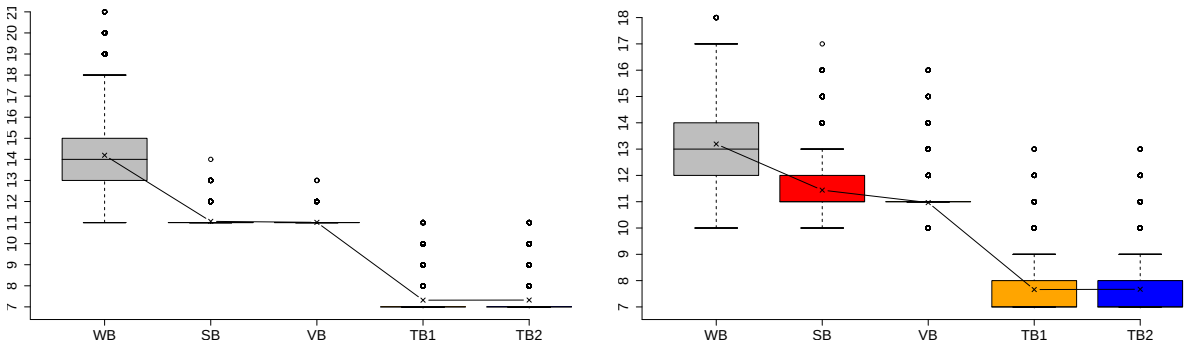


Abbildung 44: Links: Straßengraph aus der Gegend San Francisco mit 14122 Knoten. Rechts: Straßengraph aus der Gegend San Francisco mit 20025 Knoten. Die x-Achse gibt das angewendete Optimierungsverfahren und die y-Achse die Baumhöhe an.

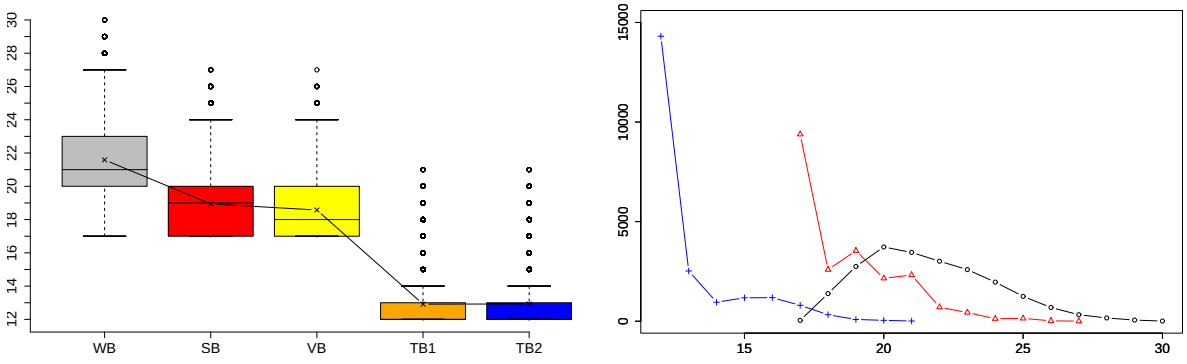


Abbildung 45: Straßengraph von Karlsruhe und Umgebung mit 25539 Knoten. Links: Die x-Achse gibt das angewendete Optimierungsverfahren und die y-Achse die Baumhöhe. Rechts: Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit der Baumhöhe an. ($\circ$=WB, $\triangle$=VB, $+$=TB1)

Zusammenfassung Insgesamt konnte anhand den Experimenten beobachtet werden, dass mit der Triangulierungsheuristik eine deutliche Verbesserung bzgl der erwarteten Baumhöhe bei zufälliger Wahl eines Wurzelknotens erzielt wurde und diese im Vergleich zur Höhenminimierungsheuristik eine besseres Resultat hinsichtlich dem Erwartungswert der Baumhöhen liefert. Dabei hängt das Ergebnis der Triangulierungsheuristik im wesentlichen davon ab, wie viele Kanten der Eingabegraph hat. Je weniger Kanten im Graphen existieren, desto mehr neue Kanten können durch die Triangulierung hinzukommen, die Distanzen zwischen Knoten verringern. Insbesondere wenn der Graph bereits trianguliert ist, stimmt die Triangulierungsheuristik mit der Höhenminimierungsheuristik überein.

In den unten angegebenen Tabellen sind die Testgraphen und die Ergebnisse der durchgeführten Experimente noch einmal vollständig aufgelistet.

Graph	Knotenanzahl	Kantenanzahl
Rect_80_20	1600	3100
Rect_40_40	1600	3120
Rect_100_100	10000	19800
LEDA_5000	5000	12500
LEDA_10000	10000	25000
Delaunay_5000	5000	12500
Delaunay_10000	10000	25000
mcity03	14122	14986
mcity04	20025	20728
Karlsruhe	25539	32615

Graph	rad	diam	$\min$(TB1)	$\max$(TB1)	EW(WB)	EW(TB1)
Rect_80_20	10	20	10	20	15.081	13.306
Rect_40_40	20	27	20	21	22.690	20.486
Rect_100_100	50	67	50	51	56.298	50.596
LEDA_5000	7	14	7	13	10.842	8.025
LEDA_10000	8	15	7	14	11.475	8.289
Delaunay_5000	26	34	24	32	29.681	27.008
Delaunay_10000	33	45	31	42	39.622	36.234
mcity03	11	21	7	11	14.194	7.318
mcity04	10	18	7	13	13.193	7.660
Karlsruhe	17	30	12	21	21.586	12.918

4 Planar-Separator-Theorem

Ein wichtiges Hilfsmittel für die Separation von planaren Graphen ist das *Planar-Separator-Theorem* (PST). Das Schöne an diesem Theorem ist die Aussage über die lineare Laufzeit des Verfahrens, welches die Verwendung dieses Theorems in der Praxis motiviert. Wie schon in der Einleitung erwähnt wurde, ist eine hinsichtlich der Separatorgröße und Balance akzeptable Separation von allgemeinen Graphen ein NP-schweres Problem. Wobei jedoch, für die Klasse der planaren Graphen eine akzeptable Separation in linearer Zeit möglich ist. Im Folgenden fassen wir die wichtigen Lemmata und Sätze zusammen, die bei der Separation von planaren Graphen verwendet werden.

Satz 4.1 ***Planar-Separator-Theorem*** *Die Knotenmenge eines zusammenhängenden planaren Graphen $G = (V, E)$, $n = |V| \geq 5$ kann so in linearer Zeit $O(n)$ in drei Mengen $V_1,\ V_2,\ S \subseteq V$ partitioniert werden, dass*

1. *S Separator, der V_1 und V_2 trennt,*
2. *$|V_1|,\ |V_2| \leq \frac{2}{3} \cdot n$,*
3. *$|S| \leq 4 \cdot \sqrt{n}$.*

Ein anderes wichtiges Lemma, welches insbesondere im Satz 4.1 Verwendung findet, ist das *FundamentalkreisLemma* (FCL) mit dem man auch einen Graphen separieren kann. Die Anzahl der Separationsknoten ist hierbei durch die Höhe eines aufspannenden Baumes nach oben abgeschätzt.

Lemma 4.1 *Fundamentalkreis-Lemma Sei $G = (V, E)$ ein planarer zusammenhängender Graph mit $|V| = n \geq 5$ und $T = (V, E(T))$ ein aufspannender Baum von G mit Wurzel w und Höhe h. Die Knotenmenge von G kann so in linearer Zeit $O(n)$ in drei Mengen $V_1,\ V_2,\ S \subseteq V$ partitioniert werden, dass*

1. *S Separator, der V_1 und V_2 trennt,*
2. *$|V_1|, |V_2| \leq \frac{2}{3} \cdot n$,*
3. *$S \leq 2 \cdot h + 1$.*

Die Suche nach einem aufspannenden Baum minimaler Höhe wurde hauptsächlich durch dieses Lemma motiviert. Wenn es möglich wäre, aufspannende Bäume mit geringerer Höhe zu konstruieren, so könnten wir damit kleinere obere Schranken für die Separatorgröße angeben und damit würde sich auch eventuell die Separatorgröße verringern. Daher haben wir in den vorherigen Kapiteln Heuristiken und Verfahren untersucht, mit Hilfe denen wir Bäume geringere Höhe finden und konstruieren können, ohne dabei die Linearität der Laufzeit des Gesamtverfahrens zu verletzen. In diesem Kapitel wird nun das Gesamtverfahren des PST in verschiedenen Phasen eingeteilt und vollständig erläutert. Danach wird eine Testreihe aufgestellt, um das PST mit den zusätzlichen Heuristiken und der nicht optimierten Variante des PST vergleichen zu können. Wir werden sehen, dass in der Tat eine deutliche Verbesserung bei den untersuchten Graphklassen erzielt werden konnte.

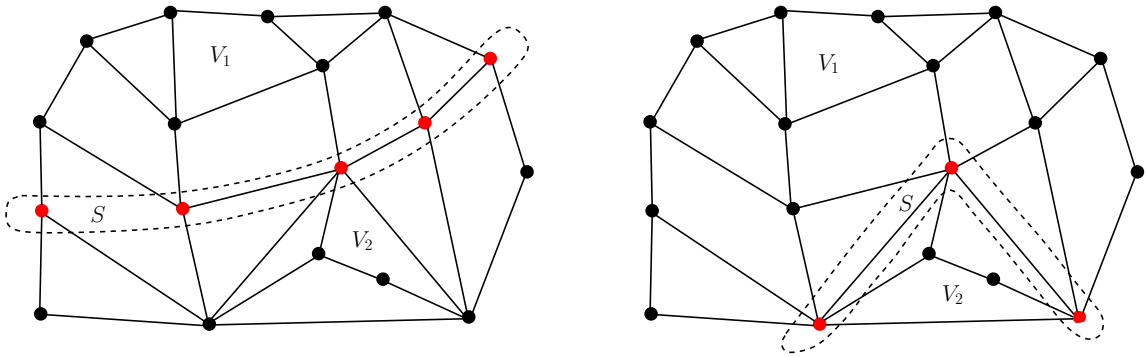


Abbildung 46: Ein Graph mit 18 Knoten. Links: Ein Separator, der die Bedingungen des PST erfüllt $|V_1| = 7,\ |V_2| = 6,\ |S| = 5$. Rechts: Ein Separator, der die Bedingungen des PST nicht erfüllt $|V_1| = 13,\ |V_2| = 2,\ |S| = 3$.

4.1 Phasen des PST

Das PST kann man in zwei Phasen einteilen. Eine Phase der einfachen Separation und eine Phase der komplexen Separation. In der Phase der einfachen Separation wird von einem beliebigen Knoten w im gegebenen planaren Graphen $G = (V, E)$ aus ein Breitensuchbaum gebildet, der eine beliebige Höhe h hat. Die Knoten des Graphen werden hierbei wie in Abschnitt 2.3.1 in Levels eingeteilt. Dann werden drei Levels Level(m), Level(μ) und Level(M) so ausgesucht, dass folgendes gilt:

1. $\mu = \min\{i \mid \sum_{k=0}^{i} \text{Level}(K) \geq \frac{n}{2}\}$,
2. $m = \max_{i \leq \mu}\{i \mid \text{Level}(i) < \sqrt{n}\}$,
3. $M = \min_{i \geq \mu}(\{i \mid \text{Level}(i) < \sqrt{n}\} \cup \{h+1\})$.

Anhand dieser Level wird dann versucht, wenn möglich, einen Separator anzugeben. Wir nennen eine Separation gültig, wenn sie die Garantien des PST erfüllt. Ansonsten bezeichnen wir sie als ungültig. Falls $|\text{Level}(\mu)| \leq 4 \cdot n$ und $\mu < h$, so liefert die folgende Belegung der Mengen $V_1,\ V_2$ und S wegen der Leveleinteilungseigenschaft von Breitensuchbäumen einen gültigen Separator.

1. $S := Level(\mu)$,
2. $V_1 := \bigcup_{i=0}^{\mu-1} \text{Level}(i)$,
3. $V_2 := \bigcup_{\mu+1}^{h} \text{Level}(i)$.

Ansonsten betrachtet man die Menge $A_2 := \bigcup_{i=m+1}^{M-1} \text{Level}(i)$. Falls $|A_2| \leq 2/3 \cdot n$ ist, so kann man ebenfalls einen Separator mit einer anderen geschickteren Wahl von S, V_1 und V_2 angeben. Hierfür betrachtet man zusätzlich die Knotenmengen $A_1 := \bigcup_{i=0}^{m-1} \text{Level}(i)$ und $A_3 := \bigcup_{i=M+1}^{h} \text{Level}(i)$. Dann ergibt die folgende Wahl der drei Mengen $V_1,\ V_2$ und S eine gültige Separation.

1. $V_1 := A_k\ ; k \in \{1,2,3\} \wedge |A_k| = \max\{A_1, A_2, A_3\}$,

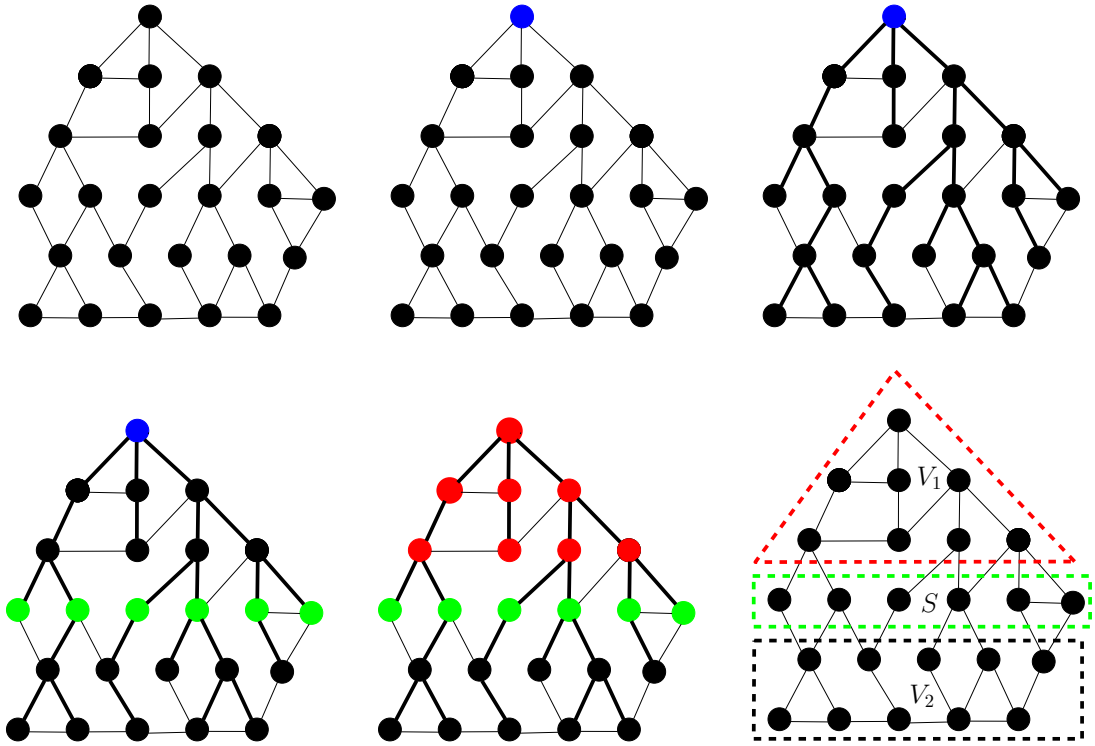


Abbildung 47: Veranschaulichung der einfachen Separationsphase.

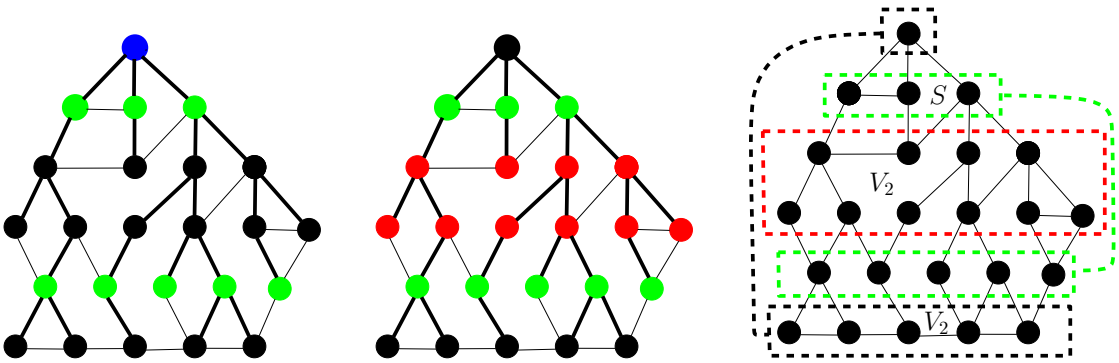


Abbildung 48: Veranschaulichung der einfachen Separationsphase.

2. $S := \text{Level}(m) \cup \text{Level}(M)$,

3. $V_2 := V \setminus (V_1 \cup V_2)$.

Falls aber $|A_2| \geq 2/3 \cdot n$ ist, so gehen wir in die komplexe Separation über. In dieser Phase wird aus den Graphen G ein neuer Graph G' erzeugt, in dem man die Knoten $\bigcup_{i=0}^{m} \text{Level}(i)$ durch sukzessives Zusammenziehen von Kanten zu einem einzigen Knoten verschmilzt und alle Knoten aus $\bigcup_{i=M}^{h} \text{Level}(i)$ entfernt. Auf G' wird dann das Lemma 4.1 angewendet und ein Separator in G' gefunden, den man leicht zu einem Separator von G, der die Bedingungen des PST erfüllt, umbauen kann.

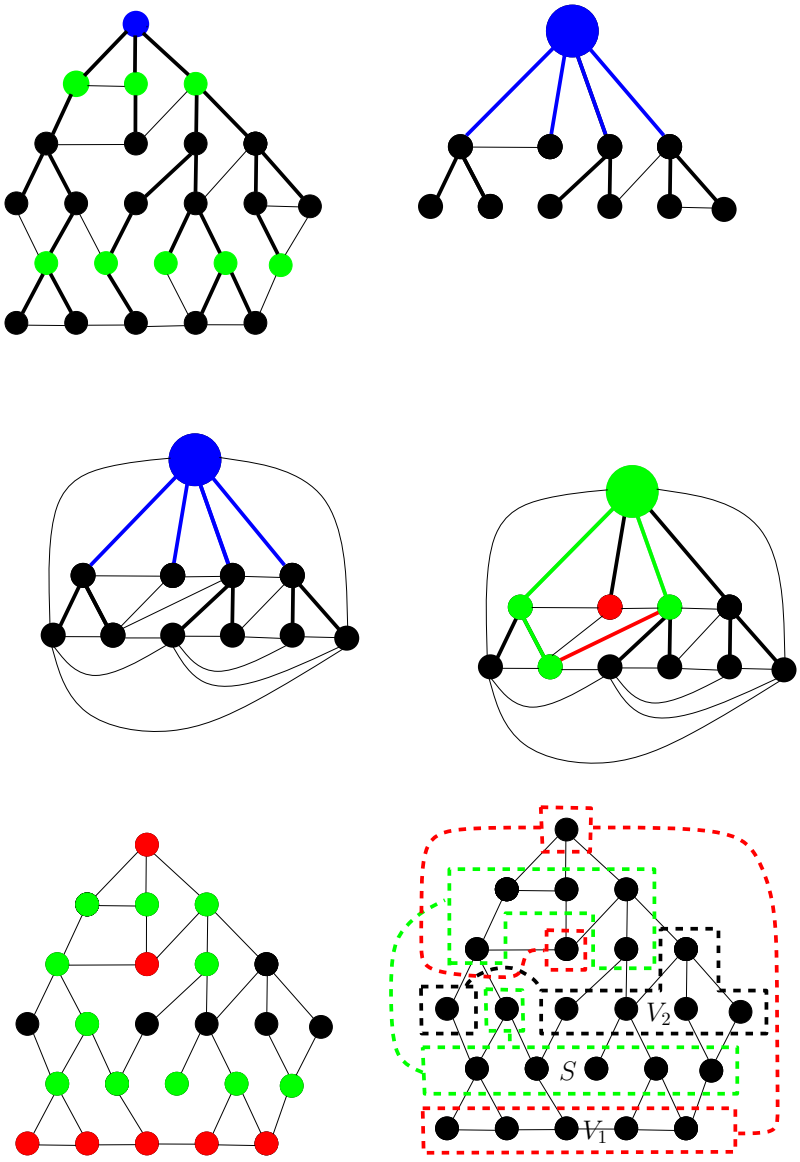


Abbildung 49: Veranschaulichung der komplexen Separationsphase.

4.2 Optimierungskriterien

Bei praktischen Anwendungen von Algorithmen, die einen Separator benötigen, gibt es eine Vielzahl anwendungsspezifischer Optimierungsgrößen. Drei dieser Optimierungsgrößen, die oft betrachtet werden sind die *Balance*, die *Separatorgröße* und die *Ratio*. Sei oBdA $|V_1| < |V_2|$, so ist die Balance definiert als $|V_1|/|V_2|$. Je näher dieser Wert an 1 ist, desto geringer wird die Rekursionstiefe bei *Divide-and-Conquer* Ansätzen sein. Die Separatorgröße beeinflusst entscheidend die Laufzeit, wenn Teillösungen zu Gesamtlösungen zusammengesetzt werden. Aus diesen beiden Tatsachen ergibt sich der Kompromisswert Ratio, der als $|S|/|V_1|$ definiert wird. Bei Optimierungen, in denen man sowohl die Separatorgröße als auch die Balance verbessern möchte, sollte man die Ratio als Optimierungskriterium verwenden. Je kleiner die Ratio letztendlich ist, desto besser ist die angewandte Optimierung.

4.3 Freiheitsgrade

Wenn wir uns die Phase der einfachen Separation im Abschnitt 4.1 anschauen, stellen wir fest, dass wir in der Konstruktion des Breitensuchbaums dieser Phase frei sind. Die Frage ist nun, welchen Einfluss der Breitensuchbaum auf das PST haben kann. In dieser Arbeit wird untersucht, ob man durch eine geschicktere Konstruktion dieses Breitensuchbaums, es schaffen kann, die Performanz des Verfahrens zu verbessern. Wir vermuten, dass ein hoher Breitensuchbaum eine bessere Verteilung der Knoten verursacht. Außerdem wird es vielleicht nicht mehr so oft nötig sein, in die komplexe Phase des Theorems einzusteigen, da eine bessere Verteilung der Knoten die Wahrscheinlichkeit erhöht, bereits bei der einfachen Separationsphase einen gültigen Separator zu finden. Anders als in der ersten Phase wird in der zweiten Phase nicht unbedingt ein Breitensuchbaum benötigt, sondern nur ein aufspannender Baum. Dieser sollte eine möglichst geringe Höhe haben, da die Höhe des Baumes, wie man anhand des FCL sehen kann, die Schranke der Separatorgröße bestimmt. Bei aufspannenden Bäumen sind einige Baumparameter entscheidend von der Wurzelwahl abhängig. Daher kann man die Wahl einer geeigneten Wurzel ebenfalls als ein Freiheitsgrad bezeichnen. Hinzu kommt noch die Triangulierung, die direkten Einfluss auf den Radius und Durchmesser des Graphen haben kann. Die Wahl der Nichtbaumkante wird in dieser Arbeit nicht näher untersucht, da sie Teil vorausgegangener Arbeiten ist, und ausreichend untersucht wurde. Die Wahl der Einbettung wird ebenfalls nicht betrachtet. Insgesamt kann man die genannten Freiheitsgrade wie folgt zusammenfassen.

- Konstruktion eines Breitensuchbaums für die einfache Separationsphase.
- Konstruktion eines aufspannenden Baumes für die komplexe Separationsphase.
- Triangulierung des Graphen G' in der komplexen Separationsphase.
- Wahl einer Wurzel.
- Wahl einer Nichtbaumkante.
- Wahl der planaren Einbettung des Graphen.

4.4 Vorausgegangene Arbeiten

Wie bereits erwähnt, wurde das *Planar-Separator-Theorem* von Lipton und Tarjan eingeführt [1]. Sie haben einen linearen Algorithmus angegeben, der einen Separator kleiner als $2\sqrt{2n}$ in einem gegebenen planaren Graphen findet, der den Graphen in zwei Komponenten kleiner als $2n/3$ aufteilt. Djidjev verbesserte die Schranke der Separatorgröße auf $6\sqrt{n}$, und lieferte einen Beweis für eine untere Schranke von $1.55\sqrt{n}$ [2]. Beide Algorithmen basieren auf das *Fundamental-Cycle-Lemma*. Seither entstanden eine ganze Reihe von Verfeinerungen. Schließlich wurde die derzeit beste untere Schranke $1.97\sqrt{n}$. von Djidjev und Venkatesan bewiesen [3].

Ein weiteres Paper über das *Planar-Separator-Theorem* (PST) mit einigen interessanten Ergebnissen wurde von Holzer et al. in [10] veröffentlicht. Bei der experimentellen Phase dieser Studie, kann man beobachten, dass das Auslassen der Leveleinteilung in der ersten Phase und die Betrachtung von Fundamentalzyklen als Separatoren ein deutlich besseres Ergebnis liefert. Daher kann man, was praktische Anwendungen betrifft, direkt auf den Eingabegraphen das *Fundamental-Cycle-Lemma* (FCL) anwenden. Diese Art der Separation wird auch *Fundamental-Cycle- Separation* (FCS) genannt. Was die Garantien dieses Verfahrens betrifft, kann man lediglich eine Separatorgröße kleiner als $2d + 1$ garantieren, wobei mit d der Durchmesser des Eingabegraphen gemeint ist. Desweiteren wurden in diesem Paper eine ganze Reihe von Optimierungsansätzen experimentell untersucht. Dabei wurden die optimierten Algorithmen auf eine Vielzahl von Graphklassen angewendet, und statistisch bewertet. Es hat sich herausgestellt, dass man durch die Berücksichtigung einiger weiterer Freiheitsgrade ein nochmal besseres Ergebnis in der Praxis erzielen kann. Mit diesen experimentellen Ergebnissen werden wir uns nun näher befassen.

4.5 Experimente

In diesem Abschnitte geht es darum, die verschiedenen Verfahren und Optimierungsansätze, die man im Laufe dieser Arbeit über aufspannende Bäume und der Triangulierung von planaren Graphen entwickelt hat im PST einzubauen und deren Auswirkung auf die Balance und Separatorgröße zu testen. Mit den derzeitigen Erkenntnissen aus den vorausgegangenen Kapiteln existieren eine Vielzahl von Experimenten, die man durchführen und bewerten kann. Es wird versucht, anhand von einigen Experimenten auf die wesentlichen Ergebnisse die entstanden sind anzudeuten. Für die Tests hat man viele dieser Verfahren miteinander kombiniert und eine Reihe von Testläufen zusammengestellt, die im Folgenden vorgestellt werden.

- **A:** Es wird nur die FCS ohne Optimierung angewendet.
- **B:** Es wird die die FCS angewendet, wobei statt der ursprünglichen Triangulierung die triangulierende Breitensuche benutzt wird.
- **C:** Es wird die modifizierte Höhenabschätzung mit 10 Durchläufen angewendet, in der versucht wird, Hohe Breitensuchbäume zu konstruieren, um die Breitensuchhöhe eines Knotens besser abschätzen zu können. Danach werden 5 Iterationen der Höhenminimierungsheuristik und anschließende 5 für die Triangulierungsheuristik spendiert.
- **D:** Es wird die Höhenminimierungsheuristik mit 10 Durchläufen angewendet und danach aus dem Ausgabeknoten eine triangulierende Breitensuchbaum gebildet.

- **E:** Es wird die Höhenminimierungsheuristik mit 15 Durchläufen angewendet und danach der Ausgabeknoten als Startknoten der Triangulierungsheuristik mit 5 Durchläufen verwendet.
- **F:** Es wird das klassische PST angewendet, bei der nicht gleich in die Komplexe phase eingestiegen wird, sondern erst dann, wenn die einfache Separationsphase fehlgeschlagen hat.
- **G:** Es wird die Höhenmaximierungsheuristik mit 10 Durchläufen im klassischen PST angewendet.

4.5.1 Gittergraphen

Wie auch zuvor werden wir jeden Knoten des Graphen einmal als Startknoten wählen und die erzielten Ergebnisse abspeichern. Danach werden die Ergebnisse vom optimierten und nicht optimierten Fall miteinander verglichen. Als Testgraphen haben wir ein quadratisches Gitter mit 10000 Knoten gewählt. Es konnte eine Verbesserung der Separatorgröße erzielt werden. Um zu sehen, wie die Separation mit der bei der Separation verwendeten Bäume zusammenhängt, haben wir die Baumhöhen der Separation auch abgespeichert und verglichen. Es hat sich in der Tat herausgestellt, dass die Separatorgröße mit der Baumhöhe korrespondiert. Außerdem sieht man deutlich, das große Separatoren nicht mehr vorkommen und der Erwartungswert der Separatorgröße sich durch die Optimierung verändert hat. Man möge sich selbst anhand der folgenden Diagramme ein Bild machen.

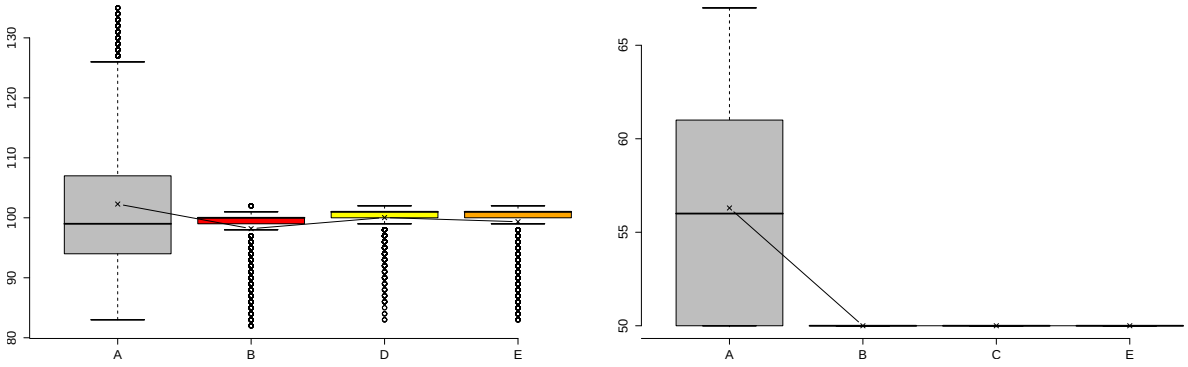


Abbildung 50: Quadratisches Gitter mit 10000 Knoten (100×100). Links: Die x-Achse gibt das angewendete Verfahren an und die y-Achse die Separatorgröße. Rechts: Die entsprechenden Baumhöhen der Separation. Die x-Achse gibt das angewendete Verfahren und die y-Achse die Baumhöhe an.

4.5.2 Diametergraphen

Für die Tests haben wir einen Diamtergraphen mit dem Durchmesser 10000 ausgesucht. Da Diametergraphen bereits trianguliert sind, kann man hier nur mit den Optimierungsverfahren der Höhenminimierung oder Höhenabschätzung ansetzen. In Diametergraphen besteht ein optimaler Separator genau aus drei Knoten. Bei Diametergraphen mit anderem Durchmesser haben wir genau die gleichen Ergebnisse erzielt und werden deshalb nicht alle Experimente als Diagramme präsentieren. Die unteren Diagramme zeigen welche Wirkung die Optimierung auf die Separatorgröße hat.

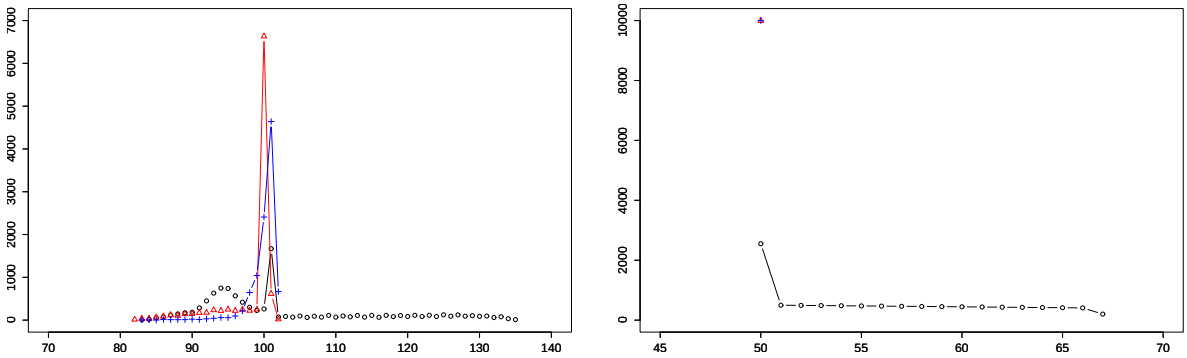


Abbildung 51: Quadratisches Gitter mit 10000 Knoten (100×100). Links: Vergleich der Separation. Die x-Achse gibt die Separatorgröße und die y-Achse die Häufigkeit der Separatorgröße an. Rechts: Vergleich der Baumhöhen bei der Separation. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit der Baumhöhe an. ($\circ$ = Ohne Optimierung, $\triangle$ = Optimierung B, $+$ = Optimierung C).

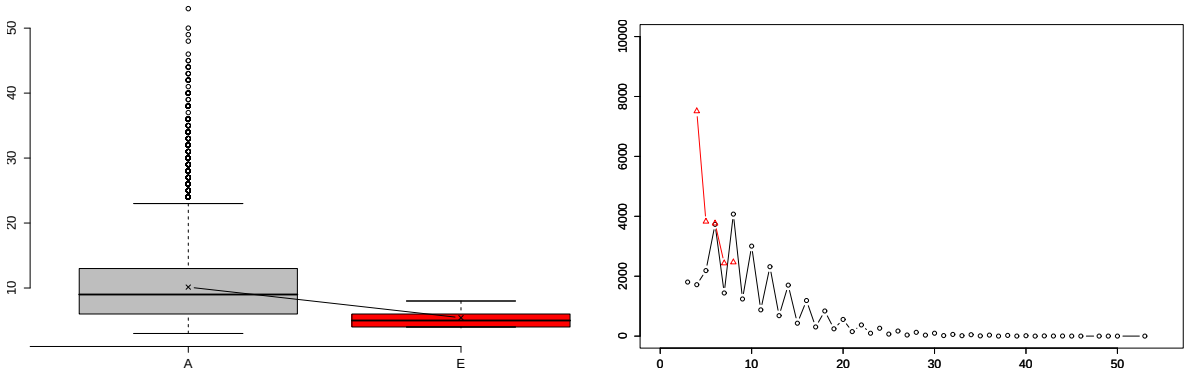


Abbildung 52: Diametergraph mit Durchemsser 10000. Links: Die x-Achse gibt die Separatorgröße an und die y-Achse das angewendete Verfahren. Rechts: Die x-Achse gibt die Häufigkeit der Separatorgröße an und die y-Achse die Separatorgröße. ($\circ$=Ohne Optimierung, $\triangle$=Experiment E).

4.5.3 Delaunay-Graphen

Beobachtungen aus den Experimenten haben gezeigt, dass Delaunay-Graphen relativ große Separatoren haben und sich nicht so leicht separieren lassen. Was die Baumhöhe angeht, hat die Höhenminimierungsheuristik auch keine große Verbesserung erbracht. Dieses Ergebnis deutet darauf hin, dass die Struktur dieser Graphen sehr speziell ist und vielleicht andere Verfahren angepasst an diese Graphklasse entwickelt und untersucht werden müssen.

Insgesamt konnte aber mit einem kombinierenden Verfahren bestehend aus einer anfänglicher Höhenabschätzung mit anschließender Höhenminimierung und zu letzt Anwendung der Triangulierungsheuristik doch noch eine Verbesserung der Separatorgröße erzielt werden. Die folgenden Diagramme verdeutlichen diese Ergebnise.

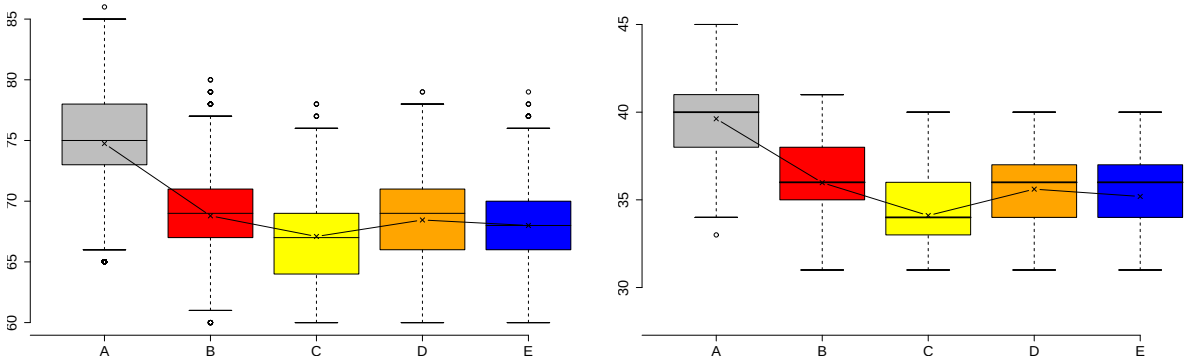


Abbildung 53: Delaunay-Graph mit 10000 Knoten. Links: Die x-Achse gibt das angewendete Verfahren und die y-Achse die Separatorgröße an. Rechts: Die entsprechenden Baumhöhen bei der Separation. Die x-Achse gibt das angewendete Verfahren und die y-Achse die Baumhöhe an.

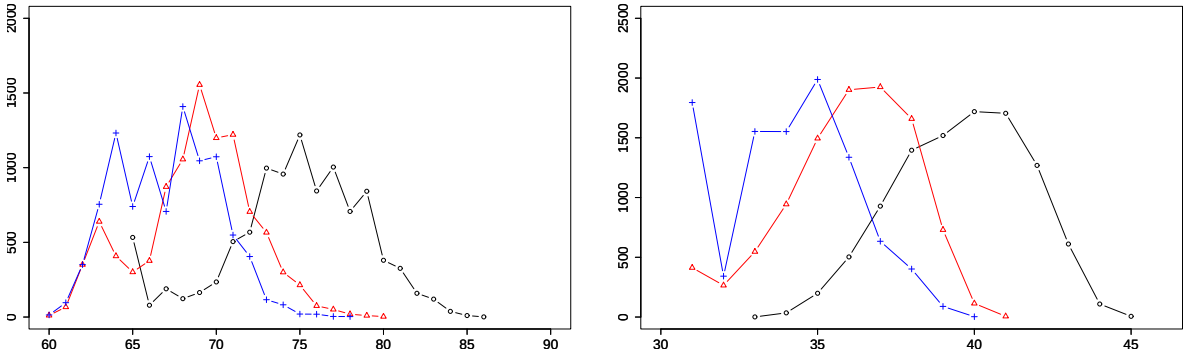


Abbildung 54: Links: Vergleich der Separation. Die x-Achse gibt die Separatorgröße und die y-Achse die Häufigkeit der Separatorgröße an. Rechts: Vergleich der Baumhöhen bei der Separation. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit der Baumhöhe an. ($\circ$ = Ohne Optimierung, $\triangle$ = Optimierung B, $+$ = Optimierung C).

4.5.4 LEDA-Graphen

LEDA-Graphen lassen sich besonders leicht separieren und haben im Vergleich zu anderen Graphen auch meistens sehr kleine Separatoren. Weil bereits der Separator, der im nicht optimierten Fall ge-

funden wird sehr klein ist, kann die Optimierung nicht sehr groß sein, da einfach kleinere Separatoren nicht existieren können. Aber dennoch konnte mit dem Einsatz der aufgestellten Heuristiken auch eine Verbesserung bei dieser Graphklasse erzielt werden.

Erstaunlich ist die Tatsache, dass die Optimierungsvariante B, bei der lediglich die triangulierende Breitensuche angewendet wird, hier und auch in den vorhergehenden untersuchten Graphklassen eine deutliche Verbesserung erbringt und das bei geringerer Laufzeit, da ja eine Triangulierung während der Breitensuche stattfindet.

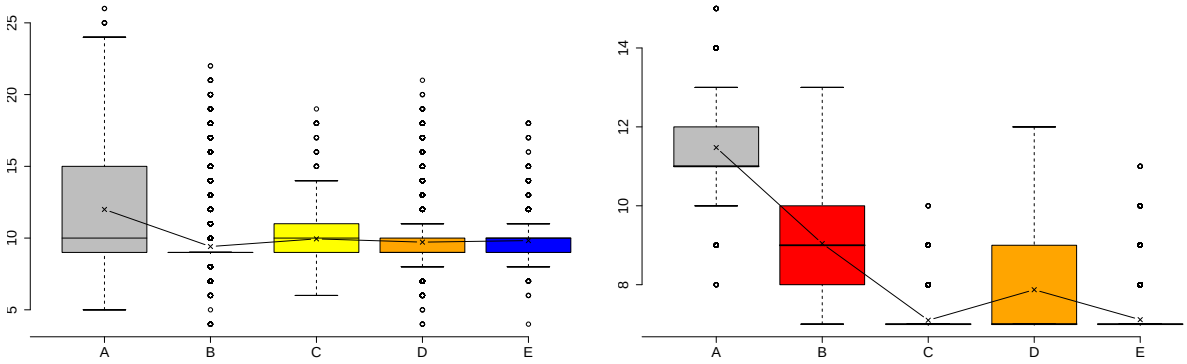


Abbildung 55: LEDA-Graph mit 10000 Knoten. Rechts: Die x-Achse gibt das angewendete Verfahren und die y-Achse die Separatorgröße an. Links: Die entsprechenden Baumhöhen bei der Separation. Die x-Achse gibt das angewendete Verfahren und die y-Achse die Baumhöhe an.

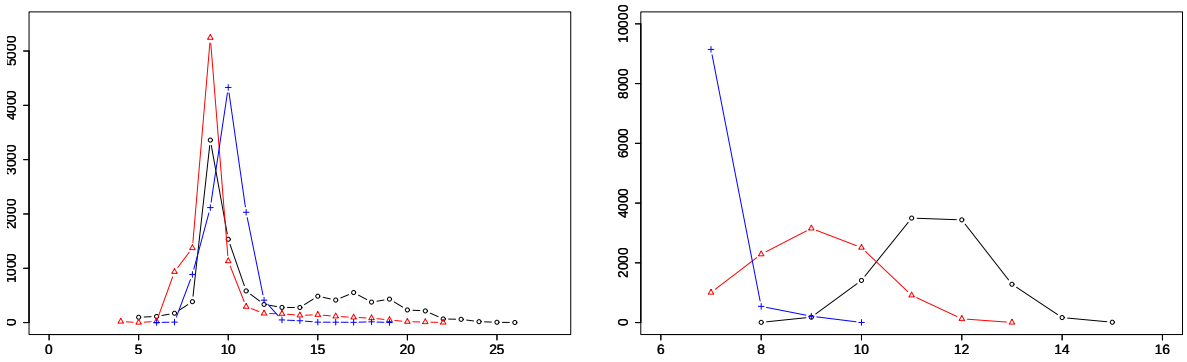


Abbildung 56: Links: Vergleich der Separation: Die x-Achse gibt die Separatorgröße und die y-Achse die Häufigkeit der Separatorgröße an. Rechts: Vergleich der Baumhöhen bei der Separation. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit der Baumhöhe an. ($\circ$ = Ohne Optimierung, $\triangle$ = Optimierung B, $+$ = Optimierung C).

4.5.5 Straßengraphen

Die Straßengraphen haben wir ausgiebiger getestet als alle anderen Graphklassen, da Straßengraphen die realen Straßennetze modellieren und damit als Testgraphen oder Eingabegraphen für Algorithmen in praktischen Anwendungen sehr oft Verwendung finden. Oft möchte man von Divide-and-Conquer Ansätzen bei sehr großen Eingabegraphen Gebrauch machen, da diese eine Parallelisierung und Nebenläufigkeit ermöglichen. Um parallele Algorithmen auf große Eingabegraphen anwenden zu können, ist das Separieren des Graphen eine Möglichkeit . Deshalb ist es umso wichtiger kleine Separatoren für diesen Ansatz finden zu können.

Wir fangen an mit dem Straßengraphen von Karlsruhe. Es ist eine erstaunlich deutliche Verbesserung der Separatorgröße zu erkennen. Vor allem sieht man hier ganz deutlich, dass die Kurve der Separatorgrößen im Diagramm, welches in der Abbildung 58 links zu sehen ist, ein ganzes Stück nach Links verschoben ist. Weiter sieht man auf dem Diagramm auf der rechten Seite in der Abbildung 58 die Baumhöhen der Separation und sieht ganz deutlich, dass die durch die Optimierung gelieferten Baumhöhen keine Schnittmenge mit dem Höhenspektrum des nicht optimierten Falles haben. Wenn wir die Laufzeit, die wir insgesamt für unsere Heuristiken verbrauchen, für das zufällige Wählen von Wurzelknoten und Konstruktion von Breitensuchbäumen investieren würden, hätten wir ein viel schlechteres Ergebnis zu erwarten.

Die weiteren Straßengraphen, die wir untersucht haben, waren Straßengraphen der Städte Hamburg, Berlin und San Francisco. Das Ergebnis welches für Karlsruhe erzielt wurde, wurde auch bei den Straßengraphen aller anderen getesteten Städte erzielt. Somit können wir laut den Beobachtungen aus den durchgeführten Experimenten behaupten, dass für Straßengraphen die Optimierung einen besonders hohen Performanzgewinn zur Folge hatte.

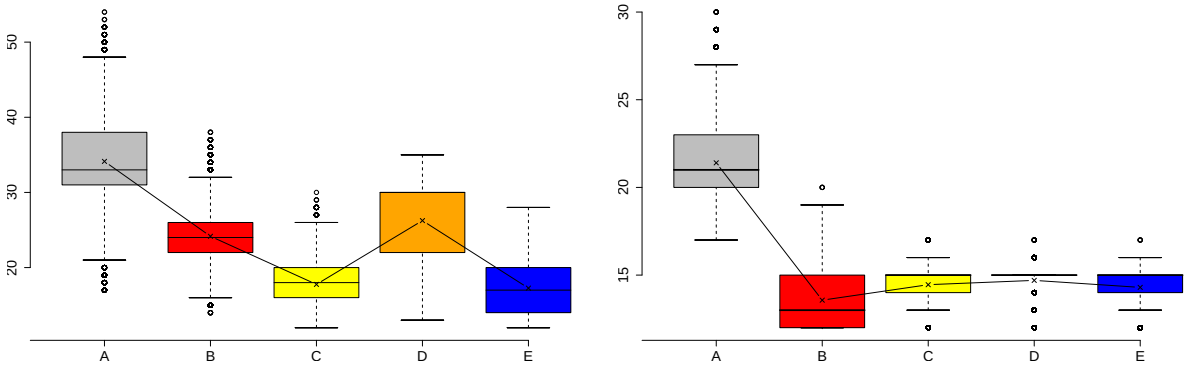


Abbildung 57: Straßengraph von Karlsruhe mit 25000 Knoten. Oben (rechts): Die x-Achse gibt das angewendete Verfahren und die y-Achse die Separatorgröße an. Oben (links): Die entsprechenden Baumhöhen bei der Separation. Die x-Achse gibt das angewendete Verfahren und die y-Achse die Baumhöhe an.

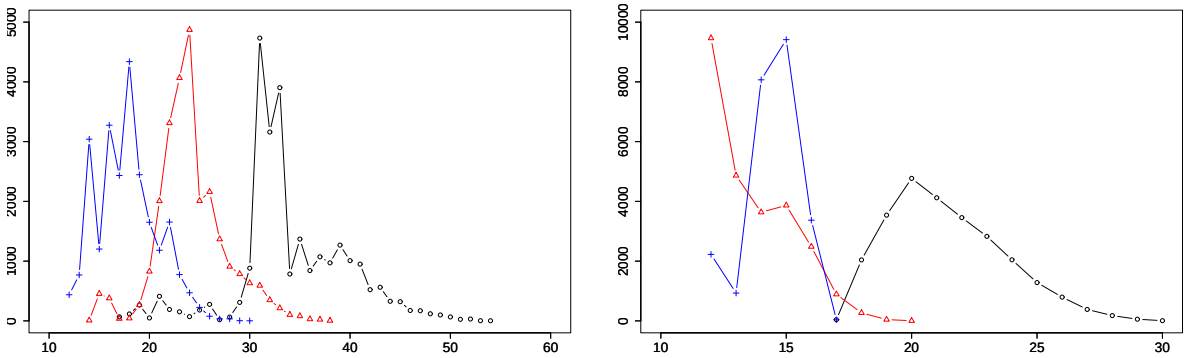


Abbildung 58: Straßengraph von Karlsruhe mit 25000 Knoten. Links: Vergleich der Separation. Die x-Achse gibt die Separatorgröße und die y-Achse die Häufigkeit der Separatorgröße an. Rechts: Vergleich der Baumhöhen bei der Separation. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit der Baumhöhe an. ($\circ$ = Ohne Optimierung, $\triangle$ = Optimierung B, $+$ = Optimierung C).

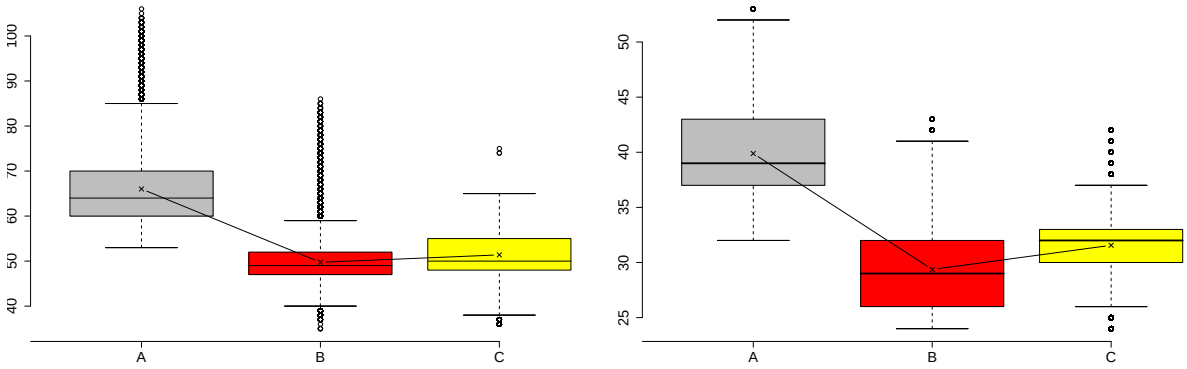


Abbildung 59: Straßengraph von Berlin mit 63336 Knoten und 177458 Kanten. Oben (rechts): Die x-Achse gibt das angewendete Verfahren und die y-Achse die Separatorgröße an. Oben (links): Die entsprechenden Baumhöhen bei der Separation. Die x-Achse gibt das angewendete Verfahren und die y-Achse die Baumhöhe an.

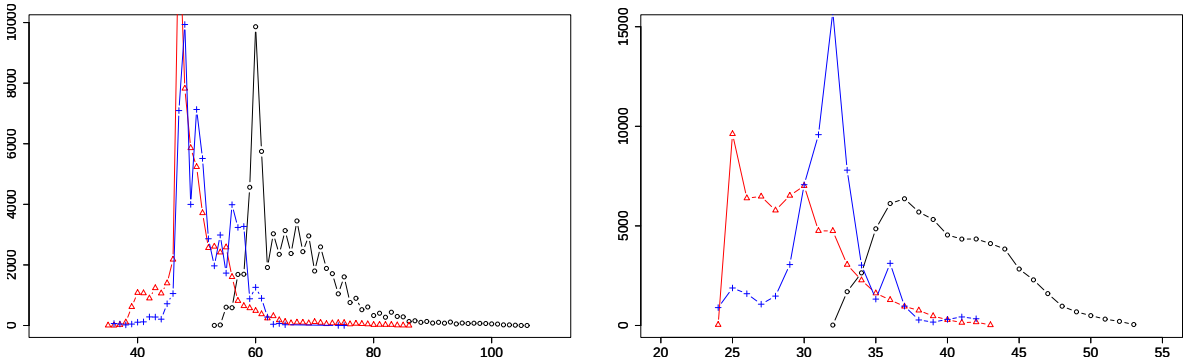


Abbildung 60: Straßengraph von Berlin mit 63336 Knoten und 177458 Kanten. Links: Vergleich der Separation. Die x-Achse gibt die Separatorgröße und die y-Achse die Häufigkeit der Separatorgröße an. Rechts: Vergleich der Baumhöhen bei der Separation. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit der Baumhöhe an. ($\circ$ = Ohne Optimierung, $\triangle$ = Optimierung B, $+$ = Optimierung C).

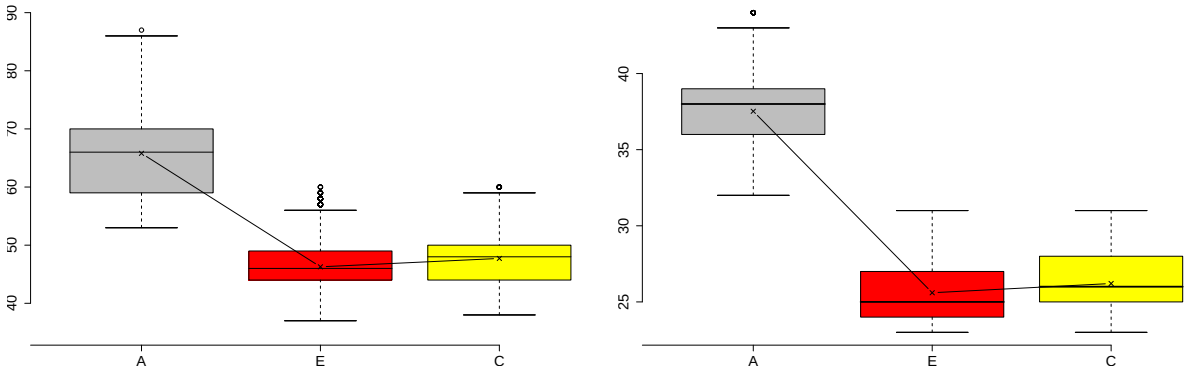


Abbildung 61: Straßengraph von Hamburg mit 42424 Knoten und 111982 Kanten. Links: Die x-Achse gibt das angewendete Verfahren und die y-Achse die Separatorgröße an. Rechts: Die entsprechenden Baumhöhen bei der Separation. Die x-Achse gibt das angewendete Verfahren und die y-Achse die Baumhöhe an.

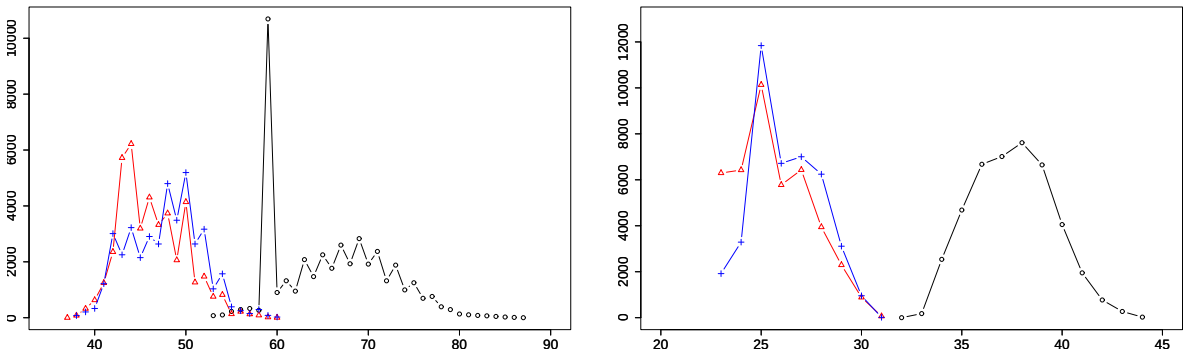


Abbildung 62: Straßengraph von Hamburg mit 42424 Knoten und 111982 Kanten. Links: Vergleich der Separation. Die x-Achse gibt die Separatorgröße und die y-Achse die Häufigkeit der Separatorgröße an. Rechts: Vergleich der Baumhöhen bei der Separation. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit der Baumhöhe an. ($\circ$ = Ohne Optimierung, $\triangle$ = Optimierung B, $+$ = Optimierung C).

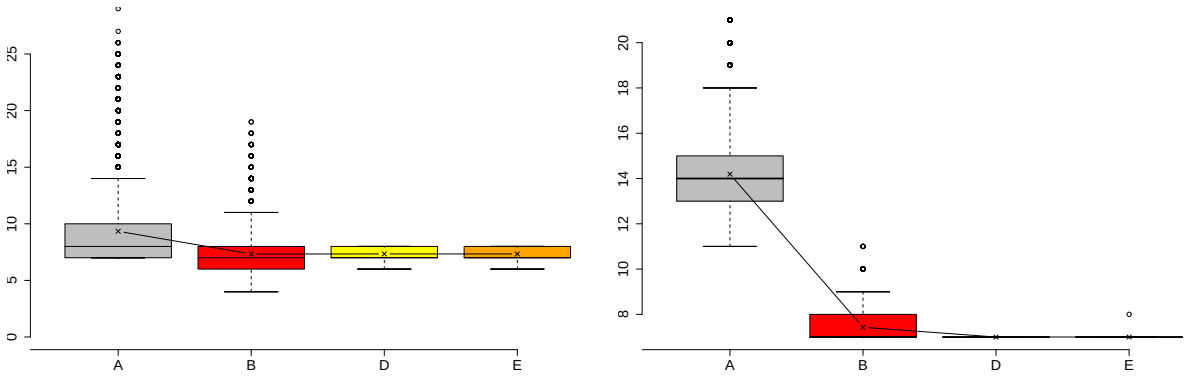


Abbildung 63: Straßengraph von SanFrancisco mit 14122 Knoten und 29972 Kanten. Links: Die x-Achse gibt das angewendete Verfahren und die y-Achse die Separatorgröße an. Rechts: Die entsprechenden Baumhöhen bei der Separation. Die x-Achse gibt das angewendete Verfahren und die y-Achse die Baumhöhe an.

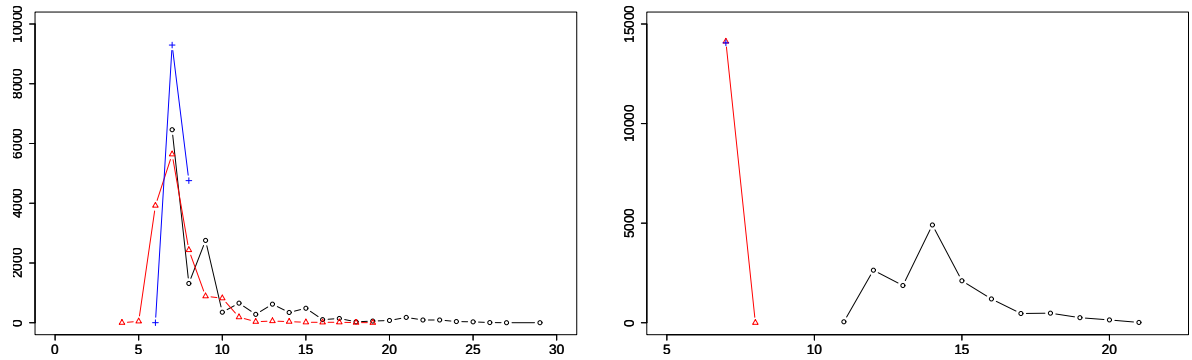


Abbildung 64: Straßengraph von San Francisco mit 14222 Knoten und 29972 Kanten. Links: Vergleich der Separation. Die x-Achse gibt die Separatorgröße und die y-Achse die Häufigkeit der Separatorgröße an. Rechts: Vergleich der Baumhöhen bei der Separation. Die x-Achse gibt die Baumhöhe und die y-Achse die Häufigkeit der Baumhöhe an. ($\circ$ = Ohne Optimierung, $\triangle$ = Optimierung B, $+$ = Optimierung C).

4.5.6 Balance

Ein anderes Ziel außer der Verbesserung der Separatorgröße, das wir anfänglich auch verfolgt hatten, war die Verbesserung der Balance. Nach unserer Intuition hatten wir vermutet, dass ein hoher Breitensuchbaum zu einer besseren Verteilung der Knoten in den Levels eines Breitensuchbaums führen würde und somit die Levels spärlicher besetzt wären. Eine spärlichere Besetzung der Levels würde dann dazu führen, dass man seltener in die komplexe Separationsphase einsteigt. Leider konnte aber bei der Balance keine deutliche Verbesserung erzielt werden. Dabei haben wir die höchsten Breitensuchbäume anhand der Höhenmaximierungsheuristik mit 10 Durchläufen erzeugt und mit dem klassischen nicht optimierten PST verglichen. Es waren keine deutliche Veränderungen zu erkennen. Die Balance hängt also nicht von der Baumhöhe ab, sondern von anderen Strukturen, die näher untersucht werden müssen. Für den Vergleich der Balance vom optimierten Fall und nicht optimierten Fall haben wir die untere Tabelle aufgestellt. Man sieht, dass außer bei den Gittergraphen die Balance sich nur um einige Tausendstel verbessert hat.

Graph	Balance (ohne Optimierung)	Balance (mit Optimierung)
diameter_1000	0.9999000	0.9998667
rect_100_100	0.9879280	1.0000000
leda_10000	0.5364803	0.5491240
delaunay_10000	0.9352510	0.9498685
mcity03	0.9927714	0.9943335
berlin	0.9922583	0.9954651
Karlsruhe	0.9858630	0.9821042

5 Zusammenfassung

In diesem Abschnitt werden wir die wesentlichen Ergebnisse zusammenfassen, die hinsichtlich des *Planar-Separator-Theorem* zu Stande gekommen sind und dem Leser einen Gesamtüberblick über unser Vorhaben und Ergebnisse in dieser Arbeit verschaffen.

Ziele Ziel dieser Arbeit war es, das PST für praktische Anwendungen zu optimieren und experimentell zu analysieren. Dafür haben wir einige Freiheitsgrade des Theorems, wie die Konstruktion aufspannender Bäume und die Triangulierung detailliert untersucht. Aus diesen Untersuchungen haben wir dann einige Heuristiken und Verfahren entwickelt, mit denen wir das PST optimieren wollten. Die wichtigsten Heuristiken waren die Höhenminimierungsheuristik und die Triangulierungsheuristik, in der eine triangulierende Breitensuche verwendet wird, die aus der Kombination einer Breitensuche und einer Triangulierung des Graphen entstanden ist.

Experimente Die Experimente bestanden darin, die Heuristiken im PST einzubauen und mit dem nicht optimierten Fall zu vergleichen. Es gab auch einige Freiheitsgrade in den aufgestellten Heuristiken, wie die Wahl einer Breitensuchvariante und Anzahl der Durchläufe. Schließlich konnten wir anhand der durchgeführten Experimente in dieser Arbeit beobachten, dass sowohl die aufspannenden Bäume als auch die Triangulierung einen direkten Einfluss auf die Seaparatorgröße haben. Im Beispiel der Straßengraphen hat sich zum Beispiel das Intervall der angenommenen Separatorgrößen für den Straßengraphen Hamburg von $[58, 87]$ (nicht optimierter Fall) zu $[37, 60]$ (optimierter Fall) verändert, welches ein beachtliches Ergebnis ist, da das Separatorgrößenspektrum fast vollständig verschoben wurde.

Außerdem hat sich die Fesstellung aus den vorausgegangenen Arbeiten nochmal bestätigt, dass die Wahl der Wurzel eine unmittelbare Wirkung auf die Qualität des Separators hat. Diese Tatsache liegt in erster Linie daran, dass die Struktur der aufspannenden Bäume, die im Verlauf des Algorithmus verwendet werden, stark von der Wurzelwahl abhängig ist und genau deswegen auch einen Einfluss auf den Separator haben. Desweiteren konnten wir auch beobachten, dass durch eine geschickte Triangulierung des Graphen, ebenfalls kleinere Separatoren gefunden wurden.

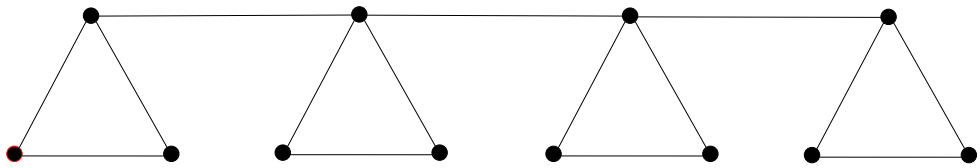
Gesamtergebnisse Insgesamt können wir behaupten, dass wir mit Hilfe der aufgestellten Heuristiken zum Zwecke der Wurzelwahl und Triangulierung des Graphen für die meisten getesteten Graphklassen eine deutliche Verbesserung hinsichtlich der Separatorgröße erzielt haben. Die Balance scheint aber von anderen strukturellen Eigenschaften des Graphen abzuhängen als die Baumhöhe. Denn mit der Höhenmaximierungsheuristik haben wir sehr hohe Breitensuchbäume konstruiert und dennoch die Balance dadurch nicht verbessern können. Im Großen und Ganzen sind jedoch die Garantien des *PST* für die Größen der zerlegten Teilgraphen weiterhin erfüllt und wir haben kleinere Separatoren gefunden.

6 Ausblick

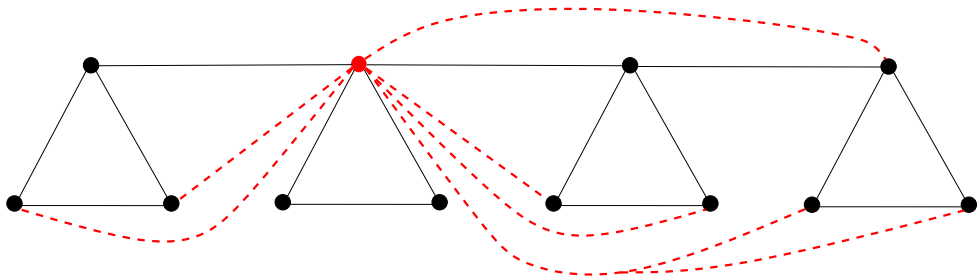
Mit den hier in dieser Arbeit betrachteten Freiheitsgraden hat man noch lang nicht alle weiteren Möglichkeiten einer Verbesserung des *PST* ausgeschöpft. Es sind noch etlich viele Optimierungsansätze

vorhanden, die es Wert sind näher untersucht zu werden und die wir im Folgenden zusammenfassen werden.

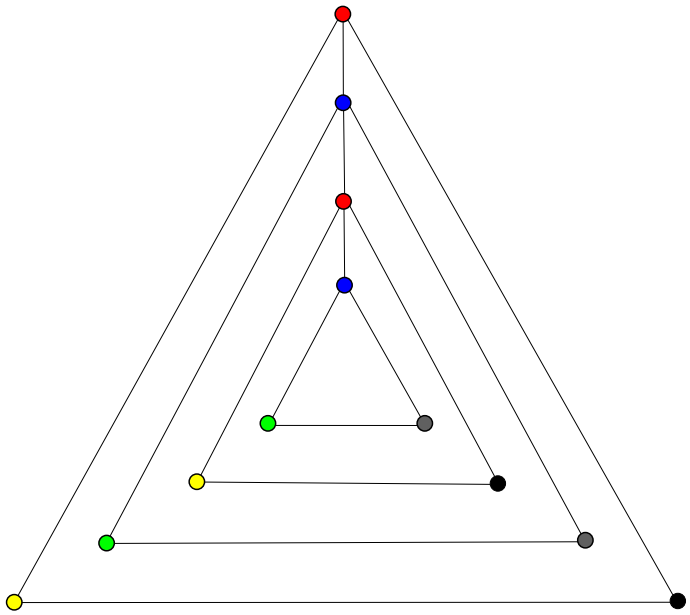
- Welche Einbettung eines planaren Graphen ist am besten für eine Triangulierung geeignet. Durch eine geeignete Einbettung und anschließender Triangulierung kann der Radius des Graphen noch kleiner werden und somit auch die Konstruktion von niedrigeren Bäumen möglich sein. Hierfür siehe man die Abbildung 65, die zwei Einbettungen des gleichen Graphen darstellt, in denen durch eine Triangulierung verschiedene Radien entstehen .
- Welche Ergebnisse würde man erhalten, wenn man die triangulierende Breitensuche mit anderen Breitensuchvarianten kombiniert. Hierfür wären z.B. die permutierenden Breitensuche besonders interessant zu untersuchen, da sie im Vergleich zu anderen Breitensuchen bessere Ergebnisse erbracht haben.
- Gibt es besondere Eigenschaften und Freiheitsgrade, mit denen wir die Balance verbessern können oder ist die Balance von der Struktur der Graphen abhängig.



(a) Einbettung eines Graphen G.



(b) Durch eine Triangulierung der Einbettung in (a) des Graphen G ist Radius 1 möglich.



(c) Andere Einbettung des Graphen G, in der Radius 1 nicht möglich ist. Die Knotenpaare mit der gleichen Farbe haben für beliebige Triangulierungen dieser Einbettung einen Mindestabstand gleich 2.

Abbildung 65: Einfluss der Einbettung eines planaren Graphen auf die durch eine Triangulierung entstehenden Radius.

Literatur

[1] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM* Journal on Applied Mathematics, 36(2):177-189, 1979.

[2] H. N. Djidjev. On the problem of partitioning planar graphs. *SIAM* Journal on Algebraic and Discrete Methods, 3(2):229-240, 1982.

[3] H. N. Djidjev and S. M. Venkatesan. Reduced constants for simple cycle graph separation. Acta Informatica, 34:231-243, 1997.

[4] L. Aleksandrov, H. N. Djidjev, H. Guo, and A. Maheshwari. Partitioning planar graphs with costs and weights. In ALENEX 2002, volume 2409 of *LNCS*, pages 98-110. Springer, 2002.

[5] D. Wagner. Algorithmen auf planaren Graphen. http://i11www.informatik.uni-karlsruhe.de/algo/teaching/scripts/index.php.

[6] D. Kozen. The Design and Analysis of Algorithms. Springer, 1992.

[7] K. Melhorn. Data Structures and Algorithms 1, 2, and 3. Springer, 1984.

[8] S.Näher and K.Mehlhorn. The LEDA Platform of Combinatorial and Geometric Computing. Cambridge University Press, 1999. http://www.algorithmic-solutions.com.

[9] Bay Area Regional Database. http://bard.wr.usgs.gov.

[10] Holzer et al. Engineering Planar Separator Algorithms. In: Proceedings of the 13th European Symposium on Algorithms (ESA), 2005. Springer-Verlag, October 2005.

Anhang

Im Folgenden werden einige Definitionen von Begriffen aus der Graphentheorie angegeben, die das Verständnis der Arbeit erleichtern.

1 ***Einfacher ungerichteter Graph:*** *Ein* einfacher ungerichteter Graph $G = (V, E)$ *besteht aus einer endlichen Menge V von Knoten und einer endlichen Menge E von Kanten, sowie einer Vorschrift, die jeder Kante $e \in E$ genau zwei Knoten $v,\ w \in V$ mit $v \neq w$ zuordnet und umgekehrt. Die Knoten $v,\ w$ werden Endknoten der Kante e genannt.*

2 ***Darstellung von Graphen:*** *Ein Graph $G = (V, E)$ kann dargestellt werden, indem man die Knoten aus V auf Punkte in der Ebene abbildet, und die Kanten aus E als Jordam-Kurven (stetige sich selbst nicht kreuzende Kurven) zwischen den Endpunkten.*

3 ***Teilgraph:*** *Ein* Teilgraph $T = (V(T), E(T))$ *eines Graphen $G = (V, E)$ ist wieder ein Graph, so dass $V(T) \subseteq V$ und $E(T) \subseteq E$.*

4 ***Weg:*** *Ein* Weg W *ist eine Folge $v_0, \ldots, v_k$ von nicht notwendig verschiedenen Knoten $v_i \in V$mit $i = 0, \ldots, k$, wobei entweder $k = 0$ oder so dass für $i = 0, \ldots, k$ gilt: $\{v_i, v_{i+1}\} \in E$. Seine Länge ist die Anzahl k der Kanten, die er enthält. Ein Weg ohne Knotenwiederholung heißt einfach.*

5 ***Kreise und Zyklen:*** *Ein Weg für den Anfangs- und Endknoten identisch sind heißt Kreis oder Zykel. Ein Kreis, an dem außer Anfangs- und Endknoten alle Knoten verschieden sind, heißt einfach.*

6 ***Zusammenhängender Graph:*** *Ein Graph $G = (V, E)$ heißt* zusammenhängend*, wenn es zwischen je zwei Knoten aus V einen Weg in G gibt. Ansonsten ist G unzusammenhängend.*

7 ***Aufspannender Baum:*** *Ein Teilgraph $T = (V(T), E(T)$ eines Graphen $G = (V, E)$ heißt* aufspannender Baum *von G, falls T ein Baum ist und $V(T) = V$. Ein beliebiger Knoten in T kann als Wurzel w ausgezeichnet sein. Dann wird T auch Wurzelbaum genannt. Das Level oder die Höhe eines Knotens v definiert sich als die Länge des eindeutigen Weges vom Knoten v zur Wurzel.*

8 ***Planarer Graph:*** *Ein Graph $G = (V, E)$ heißt planar, wenn es eine Darstellung von G gibt, in der sich die Kanten nicht kreuzen, also nur in Knoten treffen. Eine solche Darstellung nennen wir dann auch planare Einbettung. Eine planare Einbettung eines Graphen zerlegt die Ebene in Facetten (Gebiete, Flächen).*

9 ***Bemerkung:*** *Zu jedem Knoten v bzw. Kante e eines planaren Graphen existiert eine planare Einbettung des Graphen, bei der v bzw. e auf dem Rand der äußeren Facette liegt.*

10 ***Satz von Euler:*** *In einem einfachen zusammenhängenden planaren Graph $G = (V, E)$, mit $|V| = n$, $|E| = m$ und f Anzahl der Facetten gilt für jeder seiner planaren Einbettungen*

$$n - m + f = 2.$$

11 ***Bemerkung:*** *Ein zusammenhängender planarer Graph ohne Kreise (also ein Baum) mit n Knoten besitzt $n-1$ Kanten.*

12 ***Bemerkung:*** *Ein planarer einfacher Graph mit $n \geq 3$ Knoten hat höchstens $3n-6$ Kanten. Ein planarer Graph mit n Knoten ist Kantenmaximal falls er $3n-6$ Kanten besitzt.*

13 ***Triangulierung:*** *Sei $G = (V, E)$ ein planarer eingebetteter Graph. Ein Graph $G' = (V, E')$ heißt* Triangulierung *von G, falls G' ein kantenmaximaler planarer Graph ist, der G als Subgraph enthält. In einer Einbettung von G' sind alle Facetten Dreiecke.*

14 ***Separator:*** *Ein Separator S im Graphen $G = (V, E)$ ist eine Teilmenge von V, so dass durch das entfernen von S aus G, der Graph G in mindestens zwei Teilgraphen, die nicht miteinander verbunden sind, zerfällt.*