Universität Karlsruhe (TH)

Manhattan-Netzwerke und Stufenpolygone

Studienarbeit am Institut für
Logik, Komplexität und Deduktionssysteme
Prof. Dr.rer.nat. D. Wagner
Fakultät für Informatik
Universität Karlsruhe (TH)

von

cand. inform.
Florian Widmann

Betreuer:
Dipl.-Math. Marc Benkert
Dr.rer.nat. Alexander Wolff

Tag der Abgabe: 31. Januar 2005

Inhaltsverzeichnis

1 Einführung **1**

2 Bisherige Arbeiten **3**

3 Grundlegende Begriffe und Eigenschaften **5**

3.1 Definitionen grundlegender Begriffe 5

3.2 Beispiele und nützliche Eigenschaften 8

3.3 Über die Approximation minimaler Manhattan-Netzwerke 14

4 Ein Approximationsalgorithmus für Stufenpolygone **15**

4.1 Die Aufgabe 15

4.2 Ein Faktor-2-Approximationsalgorithmus 18

4.2.1 Die Idee 18

4.2.2 Der Algorithmus 18

4.2.3 Der Beweis der Faktor-2-Approximation 21

4.2.4 Eine $O(n \log n)$-Implementierung 25

4.2.5 Eine $O(n)$-Implementierung 26

5 Approximative Manhattan-Netzwerke für einen Spezialfall **29**

5.1 Hüllenmengen 29

5.2 Änderungen am Algorithmus *ApproxMMN* 35

5.3 Der Beweis der Faktor-2-Approximation 39

6 Zusammenfassung und Ausblick **47**

Literaturverzeichnis **50**

Index **51**

Abbildungsverzeichnis

3.1 (Gegen-)Beispiele für Manhattan-Wege. 7
3.2 Induzierte Gitterpunkte und induziertes Gitter. 9
3.3 Induziertes Gitter vs. minimales Manhattan-Netzwerk. 9
3.4 Illustration zur Definition 3.9. 11
3.5 Illustration zur Definition 3.10. 12

4.1 Beispiel für ein Manhattan-Polygon. 16
4.2 Beispiel für ein Stufenpolygon. 17
4.3 Beispiel für eine Zerlegung. 17
4.4 Grundprinzip des Algorithmus. 19
4.5 Bezeichnungen im Stufenpolygon. 19
4.6 Illustration zum Algorithmus *Zerlegung*. 21
4.7 Illustration zur Definition 4.6. 22
4.8 Zum Beweis von Lemma 4.7. 23

5.1 Beispiele für (nicht) orthokonvexe Mengen. 30
5.2 Beispiele für Pareto-Hüllen. 31
5.3 (Gegen-)Beispiele für Hüllenmengen. 31
5.4 Zum Beweis von Satz 5.4. 33
5.5 Zum Beweis von Lemma 5.5. 34
5.6 Illustration zur *Änderung 1* des Algorithmus. 36
5.7 Illustration zur *Änderung 2* des Algorithmus. 37
5.8 Illustration zur *Änderung 3* des Algorithmus. 38
5.9 Zum Beweis von Lemma 5.10. 41
5.10 Zum Beweis von Lemma 5.13. 43
5.11 Worst-Case-Szenarien im Beweis von Lemma 5.7. 45
5.12 Zum Beweis von Lemma 5.7. 46

1. Einführung

Einer der vielen Bedürfnisse der Menschheit war schon seit je her, Knoten jeglicher Art möglichst „günstig“ miteinander zu verbinden. Dabei hängen die Möglichkeiten der Verbindungen und die Definition von „günstig“ natürlich von der Art der Knoten und den gegebenen Umständen ab. Ein Klasse von Beispielen sind Transportnetze für Autos (also Straßen), Pakete, Strom, Daten, etc. Wir befassen uns in dieser Arbeit mit einer gegebenen Menge S von Punkten in der Ebene, die als Knoten dienen, und suchen Strecken, sodass es zwischen jeweils zwei Punkten aus S eine Verbindung gibt. Wählt man als Kosten die Gesamtlänge der Strecken bezüglich der euklidischen Norm und verlangt, dass die Verbindung zwischen jeweils zwei Punkten nicht länger als der euklidische Abstand der Punkte ist, so gibt es nur eine („sinnvolle“) Lösung, nämlich den vollständigen Graphen. Allerdings ist seine Gesamtlänge – d. h. die aufsummierte Länge seiner Kanten – sehr groß. Ist man an Graphen interessiert, die alle Punkte aus S verbinden, aber eine geringere Gesamtlänge als der vollständige Graph haben, so muss man folglich in Kauf nehmen, dass die Länge der Verbindung zwischen zwei Punkten nicht mehr minimal ist. In diesem Bereich gibt es bereits eine Vielzahl von Untersuchungen und Resultaten, als Beispiel seien die *t-Spanner* erwähnt. Bei ihnen wird gefordert, dass es zwischen je zwei Punkten aus S eine Verbindung gibt, die höchstens t Mal so lang ist wie der Abstand beider Punkte.

Verwendet man statt der euklidischen Norm die l_1-Norm (auch Manhattan-Norm genannt) und erlaubt nur horizontale und vertikale Strecken, so ändert sich die Lage. Jetzt gibt es im Allgemeinen unendlich viele Mengen von Strecken, die alle Punkte aus S paarweise miteinander verbinden, wobei die Länge der Verbindungen gerade der Manhattan-Abstand der jeweiligen Punkte ist. Diese Mengen heißen *Manhattan-Netzwerke* (für eine exakte Definition siehe Kapitel 3). Manhattan-Netzwerke sind also „1-Spanner“, wenn man die l_1-Norm verwendet und sich auf horizontale und vertikale Strecken beschränkt. Eine Verbindung zwischen zwei Punkten, deren Länge gerade der Manhattan-Abstand der beiden Punkte ist, heißt *Manhattan-Weg*. Unter allen Manhattan-Netzwerken bezüglich einer festen Menge S interessiert man sich natürlich besonders für diejenigen, deren Gesamtlänge minimal ist. Die Gesamtlänge stellt nämlich oft ein Maß für die Kosten dar. Manhattan-Netzwerke, die diese Eigenschaft erfüllen, heißen *minimale Manhattan-Netzwerke*.

Interessant sind minimale Manhattan-Netzwerke in der praktischen Anwendung immer dann, wenn nur horizontale und vertikale Strecken als Verbindung zwischen Punkten möglich sind. Dabei können selbstverständlich noch weitere Nebenbedingungen gelten, die zusätzlich beachtet werden müssen. Das Problem wird – wie fast alle theoretischen Probleme – im Allgemeinen nicht in seiner „reinen“ Form auftreten. In diesem Fall muss das Problem geeignet transformiert oder modifiziert werden. Ideal ist natürlich, wenn man es direkt auf das Problem in „Reinform“ reduzieren kann. Die Artikel, die sich bis jetzt mit Manhattan-Netzwerken beschäftigt haben, führen als mögliches Einsatzgebiet meistens recht vage den Entwurf von Schaltungen (Stichwort VLSI) an. Eine weitere potentiell mögliche Anwendung wäre, wenn auf einer bestehenden Infrastruktur (z. B. den Straßen von Manhattan) gewisse Punkte jeweils auf einem der kürzesten Wege miteinander verbunden werden sollen (z. B. durch Leitsysteme, Beleuchtung, etc.), wobei die Gesamtlänge der Wege möglichst klein sein soll. Hier würde wahrscheinlich der Broadway eine Nebenbedingung darstellen. Eine Anwendung aus der Biologie liefert [LAP03]. Grob gesagt geht es darum, zwei Gensequenzen durch Einfügen oder Auslassen von Basen ineinander zu überführen (Stichwort minimale Editierdistanz). Dabei werden minimale Manhattan-Netzwerke approximiert, um die „realistischen“ Editierpfade einzugrenzen. Die Punkte entsprechen dabei Paaren von sehr ähnlichen Teilstücken der beiden Sequenzen, die mit einer anderen Methode gefunden wurden. Allerdings müssen nicht alle Paare von Punkten miteinander verbunden werden. Auch hier ist das Problem also modifiziert. Für genauere Informationen sei auf den angegebenen Artikel verwiesen.

Im zweiten Kapitel geben wir einen kurzen Überblick darüber, welche Arbeiten sich bereits mit (minimalen) Manhattan-Netzwerken beschäftigt haben und welche Ergebnisse dabei erzielt wurden.

Im dritten Kapitel definieren wir (minimale) Manhattan-Netzwerke. Zu diesem Zweck benötigen wir eine ganze Reihe von weiteren Begriffen, die wir ebenfalls einführen. Danach machen und beweisen wir einige grundsätzliche Aussagen über (minimale) Manhattan-Netzwerke. Diese Aussagen sind zwar allgemein bekannt, allerdings gibt es in den bisherigen Veröffentlichungen keinen mathematisch sauberen Beweis. Am Ende des Kapitels wird noch kurz wiedergegeben, was man über die Komplexität des Problems „finde ein minimales Manhattan-Netzwerk“ weiß.

Das vierte Kapitel beschäftigt sich mit einem Teilproblem, welches in dem Algorithmus aus [BWW04a] auftritt. Dieser approximiert minimale Manhattan-Netzwerke mit einem Approximationsfaktor von 3, wobei seine Laufzeit in $O(n \log n)$ liegt. Wir beschreiben das Teilproblem und geben einen Algorithmus an, der es löst. Den Großteil des Kapitels nimmt der Beweis ein, dass der Algorithmus eine korrekte Lösung liefert. Danach entwickeln wir – aufeinander aufbauend – zwei Implementierungen unseres Algorithmus. Die Laufzeit der ersten Implementierung liegt in $O(n \log n)$, die Laufzeit der zweiten sogar in $O(n)$.

Im fünften Kapitel zeigen wir, dass der Algorithmus aus [BWW04a], mit dem sich schon das vorherige Kapitel beschäftigt hat, sogar eine Faktor-2-Approximation liefert, wenn die Eingabe eine *Hüllenmenge* ist (siehe Definition in Kapitel 3). Zum Verständnis dieses Kapitels ist eine sehr gute Vertrautheit mit dem Artikel [BWW04a] unbedingt notwendig, da wir große Teile der dort gemachten Definitionen und Aussagen direkt übernehmen.

2. Bisherige Arbeiten

In diesem Kapitel stellen wir die Arbeiten vor, die sich bisher mit (minimalen) Manhattan-Netzwerken beschäftigt haben. In allen Arbeiten werden Approximationsalgorithmen für minimale Manhattan-Netzwerke entwickelt. Wir beleuchten kurz die prinzipielle Herangehensweise für jeden dieser Algorithmen und geben den Approximationsfaktor und die asymptotische Laufzeit an. Im Folgenden sei P mit $n := |P|$ die Menge von Punkten, bezüglich der ein minimales Manhattan-Netzwerk approximiert werden soll.

In [GLN01] wird ein Algorithmus vorgestellt, der abhängig von P eine Menge von Stufenpolygonen bestimmt. Diese Stufenpolygone sind ähnlich wie unsere Stufenpolygone aus Kapitel 4 definiert. Für jedes der Stufenpolygone wird eine Rechteckzerlegung berechnet. Die Vereinigung der Strecken aller Zerlegungen ergibt (im Wesentlichen) ein Manhattan-Netzwerk. Wird für jedes Stufenpolygon eine minimale Rechteckzerlegung berechnet, so beträgt der Approximationsfaktor 4 und die Laufzeit liegt in $O(n^3)$. Der relativ hohe Zeitaufwand kommt daher, dass bereits die Laufzeit für die Berechnung einer minimalen Rechteckzerlegung – bei der dynamisches Programmieren verwendet wird – in $O(k^3)$ liegt. Dabei bezeichne k die Menge der Punkte des Stufenpolygons, für das die minimale Rechteckzerlegung bestimmt werden soll. Alternativ können die minimalen Rechteckzerlegungen auch approximiert werden. Es existiert ein einfacher Faktor-2-Approximationsalgorithmus, dessen Laufzeit in $O(k \log k)$ liegt. Werden die minimalen Rechteckzerlegungen lediglich 2-approximiert, so verdoppelt sich der Approximationsfaktor des eigentlichen Algorithmus auf 8. Dafür liegt seine Laufzeit in diesem Fall in $O(n \log n)$.

In [KIA02] wird ein Faktor-2-Approximationsalgorithmus präsentiert, leider ist sein Korrektheitsbeweis unvollständig. Aus diesem Grund erwähnen wir lediglich, dass in dem Artikel ein Schema für die Bildung einer *generierenden Menge* bezüglich P angegeben wird, welches auch in den beiden nächsten Arbeiten verwendet wird. Eine generierende Menge ist eine Menge von Punktepaaren aus P, die die folgende Eigenschaft besitzt: Falls in einem Netzwerk für jedes Punktepaar aus der generierenden Menge ein Manhattan-Weg existiert, so ist das Netzwerk bereits ein Manhattan-Netzwerk. Eine triviale generierende Menge ist $\binom{P}{2}$, deren Kardinalität allerdings

in $\Theta(n^2)$ liegt. Die Größe der in [KIA02] angegebenen generierenden Menge liegt dagegen in $\Theta(n)$.

In [BWW04b, BWW04a] wird ein Faktor-3-Approximationsalgorithmus präsentiert, der eine Laufzeit in $O(n \log n)$ besitzt. Der Algorithmus sorgt dafür, dass für jedes Punktepaar aus der – in [KIA02] definierten – generierenden Menge ein Manhattan-Weg eingefügt wird. Dabei werden drei Typen von Punktepaaren unterschieden und die generierende Menge wird dementsprechend in drei Teilmengen partitioniert. Die Punktepaare in einer Teilmenge werden zusammen bearbeitet, allerdings wird jede der drei Teilmengen unterschiedlich behandelt. Für den Beweis des Approximationsfaktors wird die Ebene in zwei disjunkte Flächen aufgeteilt. Das Manhattan-Netzwerk, welches der Algorithmus erstellt hat, wird durch den Schnitt mit den beiden Fläche in zwei Teile zerlegt. Jeder dieser Teile wird dann gegen den entsprechenden Teil eines minimalen Manhattan-Netzwerkes abgeschätzt. Ein Teil des Algorithmus ist Gegenstand dieser Studienarbeit und wird in Kapitel 4 entwickelt. In Kapitel 5 wird gezeigt, dass der Algorithmus sogar eine Faktor-2-Approximation liefert, wenn P eine Hüllenmenge ist.

In [CNV04] wird ein schöner Faktor-2-Approximationsalgorithmus beschrieben. Dieser stellt ein ganzzahliges lineares Programm (GLP) auf, dessen Belegung der Problemvariablen ein minimales Manhattan-Netzwerk liefert. Da ein GLP im Allgemeinen nicht in polynomieller Zeit gelöst werden kann, werden die ganzzahligen Variablen in der Problembeschreibung relaxiert und es wird das entsprechende lineare Programm (LP) gelöst. Dabei kann sich der Wert der Zielfunktion, welcher der Länge eines minimalen Manhattan-Netzwerkes entspricht, höchstens verringern. Allerdings ist die gefundene (minimale) Lösung des angegebenen Problems im Allgemeinen nicht als Manhattan-Netzwerk interpretierbar. Wenn z. B. $X = 1$ „Strecke X einfügen“ und $X = 0$ „Strecke X nicht einfügen“ bedeutet, macht die Lösung $X = 1/2$ keinen Sinn. Der Trick besteht nun darin, bestimmte Variablen auf den Wert 1 aufzurunden, sodass die korrespondierenden Strecken ein Manhattan-Netzwerk bilden. Dabei wird sichergestellt, dass der Wert der Zielfunktion der (minimalen) Lösung des LPs maximal verdoppelt wird. Dies geschieht folgendermaßen: Für jede Variable, die aufgerundet wird, wird eine Menge von Variablen – eine sogenannte *Zeugenmenge* – angegeben. Diese besitzt die Eigenschaft, dass die Summe aller Variablen aus der Zeugenmenge einen Wert von mindestens 1/2 ergeben. Dabei sind die Zeugenmengen von zwei unterschiedlichen Variablen, die aufgerundet werden, natürlich disjunkt. Insgesamt wird somit garantiert, dass die Länge des resultierenden Manhattan-Netzwerkes maximal doppelt so groß ist wie die Länge eines minimalen Manhattan-Netzwerkes. Der Zeitaufwand für das Lösen dieses LP liegt allerdings in $O(n^8)$, da die Anzahl der Variablen in $\Theta(n^2)$ liegt.

3. Grundlegende Begriffe und Eigenschaften

In diesem Kapitel sollen als erstes die Begriffe, die wir im weiteren Verlauf der Studienarbeit benötigen, erklärt und formal definiert werden. Dies beginnt mit rudimentären Begriffen wie „Strecke“ und beinhaltet nicht zuletzt die zentrale Definition des *(minimalen) Manhattan-Netzwerkes*. Im zweiten Teil des Kapitels werden wir einige Eigenschaften und Sätze über Manhattan-Netzwerke aufführen und beweisen. Diese Beobachtungen sollen dabei helfen, ein Gefühl für die vorgestellten Begriffe zu gewinnen, sie sind aber auch im Bezug auf die Erkenntnisse in den weiteren Kapiteln von Interesse. Im letzten Teil wird kurz zusammengefasst, was über die Komplexität des Problems „finde ein minimales Manhattan-Netzwerk“ bekannt ist.

3.1 Definitionen grundlegender Begriffe

Der elementarste Begriff ist der *Punkt*. Im gesamten Text werden nur Punkte in der Ebene (d. h. im $\mathbb{R}^2$) betrachtet. Für einen Punkt p bezeichnen p^x und p^y seine x- bzw. y-Koordinate. Wir vergleichen Punkte koordinatenweise. Die Relation „$\geq$“ ist folgendermaßen definiert:

$$p \geq q \quad :\Longleftrightarrow \quad p^x \geq q^x \ \wedge \ p^y \geq q^y.$$

Die übrigen Relationen (z. B. „$\leq$“) sind analog definiert. Mit $d(p,q) := \|p-q\|_{l_1} := |p^x - q^x| + |p^y - q^y|$ bezeichnen wir den Abstand zweier Punkte p und q bezüglich der l_1-Norm. Für eine Menge oder eine Sequenz von Punkten führen wir zwei abkürzende Bezeichnungen ein, die selbsterklärend sein dürften.

Definition 3.1 *Es sei S eine Menge oder eine Sequenz von Punkten. Die Menge der x- bzw. y-*Werte *von S ist definiert als*

$$S^x := \Big\{ p^x \mid p \in S \Big\} \quad \textit{bzw.} \quad S^y := \Big\{ p^y \mid p \in S \Big\}.$$

Ein weiterer grundlegender Begriff ist die *Strecke* zwischen zwei Punkten. Da in dieser Arbeit nur horizontale und vertikale Strecken von Interesse sind, soll dies in der Definition gleich berücksichtigt werden.

Definition 3.2 (Manhattan-Strecke) *Es seien p und q zwei Punkte mit $p^x = q^x$ oder $p^y = q^y$. Die* Manhattan-Strecke $s_{p,q}$ *zwischen p und q ist die Menge*

$$s_{p,q} := \left\{ t \in \mathbb{R}^2 \mid p \leq t \leq q \ \vee \ q \leq t \leq p \right\} \subseteq \mathbb{R}^2.$$

Die Länge *einer Manhattan-Strecke $s_{p,q}$ ist definiert als*

$$\|s_{p,q}\| := d(p,q).$$

Es seien $s_{\tilde{p},\tilde{q}}$ und $s_{p,q}$ zwei Manhattan-Strecken und $Q := \{\tilde{p}, \tilde{q}, p, q\}$. Wir definieren weiterhin:

- $s_{\tilde{p},\tilde{q}}$ *ist eine* Teilstrecke *von* $s_{p,q} :\Longleftrightarrow s_{\tilde{p},\tilde{q}} \subseteq s_{p,q}$
- $s_{\tilde{p},\tilde{q}}$ *und* $s_{p,q}$ überlappen *sich* $:\Longleftrightarrow |s_{\tilde{p},\tilde{q}} \cap s_{p,q}| > 1$
- $s_{\tilde{p},\tilde{q}}$ *und* $s_{p,q}$ kreuzen *sich* $:\Longleftrightarrow \exists r \in \mathbb{R}^2 \backslash Q : s_{\tilde{p},\tilde{q}} \cap s_{p,q} = \{r\}$
- $s_{\tilde{p},\tilde{q}}$ *und* $s_{p,q}$ berühren *sich* $:\Longleftrightarrow \exists r \in Q : s_{\tilde{p},\tilde{q}} \cap s_{p,q} = \{r\}$.

Wir lassen den Zusatz „Manhattan“ im Weiteren aus Bequemlichkeit meistens weg, unter einer Strecke ist also immer eine Manhattan-Strecke zu verstehen. Die Menge $\{p, q\}$ beschreibt die (Manhattan-)Strecke $s_{p,q}$ eineindeutig. Deshalb ist die Länge einer Strecke wohldefiniert. Man beachte, dass $s_{p,q}$ und $s_{q,p}$ die gleiche Strecke charakterisieren; die Reihenfolge der Punkte in der Notation spielt keine Rolle. Eine in dieser Hinsicht bessere – aber umständlichere – Schreibweise wäre $s_{\{q,p\}}$ gewesen. Um Sonderfälle zu vermeiden, ist auch die Strecke $s_{p,p}$ zugelassen, sie hat nach Definition die Länge 0. Bei den Beziehungen zwischen zwei Strecken mache man sich klar: Zwei Strecken, die sich überlappen, müssen die gleiche Ausrichtung – vertikal oder horizontal – haben. Zwei Strecken, die sich kreuzen, müssen hingegen unterschiedlich ausgerichtet sein und der Schnittpunkt muss im *Inneren* beider Strecken liegen. Unter den *inneren Punkten* einer Strecke $s_{p,q}$ verstehen wir alle Punkte in $s_{p,q}$ außer den Endpunkten p und q. Für Mengen von Manhattan-Strecken – also insbesondere für Manhattan-Wege und Manhattan-Netzwerke – führen wir folgende Bezeichnungen ein.

Definition 3.3 *Es sei M eine endliche Menge von Manhattan-Strecken. Die* Länge *von M ist definiert als*

$$\|M\| := \sum_{s \in M} \|s\|.$$

Die Menge der vertikalen *bzw.* horizontalen Strecken *von M ist definiert als*

- $M^{\text{ver}} := \left\{ s_{p,q} \in M \mid p^x = q^x \right\}$ *bzw.*

- $M^{\text{hor}} := \left\{ s_{p,q} \in M \mid p^y = q^y \right\}$.

Die Menge der Breiten *bzw.* Höhen *von M ist definiert als*

- $B(M) := \left\{ x \in \mathbb{R} \mid \exists s_{p,q} \in M^{\text{ver}} : x = p^x (= q^x) \right\}$ *bzw.*
- $H(M) := \left\{ y \in \mathbb{R} \mid \exists s_{p,q} \in M^{\text{hor}} : y = p^y (= q^y) \right\}$.

Es gilt $M = M^{\text{hor}} \cup M^{\text{ver}}$. Die Höhen von M sind genau die y-Koordinaten, auf deren Höhe eine horizontale Strecke aus M liegt. Analoges gilt für die Breiten.

Eine Bedingung für Manhattan-Netzwerke ist, dass die Punkte paarweise durch Streckenzüge verbunden sind, die gewissen Anforderungen genügen müssen. So dürfen nur horizontale oder vertikale Strecken verwendet werden und die Länge des Streckenzuges muss genauso groß wie der Abstand der entsprechenden Punkte in der l_1-Norm sein. Wir definieren *Manhattan-Weg* formal (siehe auch Abbildung 3.1).

Definition 3.4 (Manhattan-Weg) *Es seien p und q zwei Punkte. Ein* Manhattan-Weg *(MW) $W_{p,q}$ zwischen p und q ist eine endliche, nicht leere Menge von Manhattan-Strecken $\{s_{p_1,q_1}, \ldots, s_{p_n,q_n}\}$, für die gilt:*

(i) $p_j^x = q_j^x \ \vee \ p_j^y = q_j^y \qquad \forall j \in \{1, \ldots, n\}$,

(ii) $p_1 = p \ \wedge \ q_n = q$,

(iii) $q_j = p_{j+1} \qquad \forall j \in \{1, \ldots, n-1\}$,

(iv) $\|W_{p,q}\| \ = \ d(p, q)$.

Wie schon bei der Manhattan-Strecke spielt die Reihenfolge von p und q in der Notation keine Rolle. Falls p und q nicht auf derselben horizontalen oder vertikalen Gerade liegen, gibt es unendlich viele Manhattan-Wege zwischen zwei Punkten p und q.

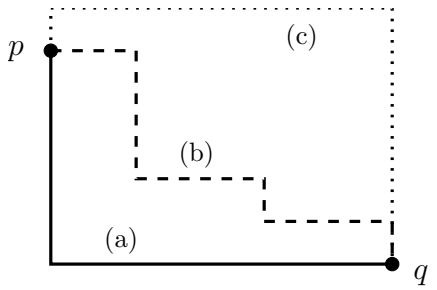


Abbildung 3.1: (a) und (b) zeigen zwei mögliche MWe zwischen p und q. (c) ist *kein* MW, da die Bedingung (iv) verletzt ist.

Damit kommen wir zum zentralen Begriff des *Manhattan-Netzwerkes*. Die Aufgabe besteht darin, eine gegebene Menge S von Punkten so mit Strecken zu verbinden,

dass zwischen jedem Paar von Punkten ein Manhattan-Weg existiert. Zusätzlich fordern wir, dass die Strecken sich weder überlappen noch kreuzen oder außerhalb ihrer Endpunkte berühren dürfen. Außerdem dürfen sie in ihrem Inneren keine Punkte aus S enthalten. Von besonderem Interesse sind natürlich die Manhattan-Netzwerke, bei denen die Gesamtlänge aller Strecken minimal ist, sie werden als *minimale Manhattan-Netzwerke* bezeichnet. Die formale Definition lautet:

Definition 3.5 (Manhattan-Netzwerk) *Es sei S eine endliche Menge von Punkten. Ein* Manhattan-Netzwerk *(MN) M_S bezüglich S ist eine endliche Menge von Manhattan-Strecken, für die gilt:*

(i) $\forall p, q \in S, p \neq q : \exists MW\ W_{p,q} : \forall s \in W_{p,q} : \exists \tilde{s} \in M_S : s \subseteq \tilde{s}$,

(ii) $\forall s_{p,q} \in M_S : \forall \tilde{s} \in M_S, \tilde{s} \neq s_{p,q} : s_{p,q} \cap \tilde{s} \subseteq \{p, q\}$,

(iii) $\forall s_{p,q} \in M_S : s_{p,q} \cap S \subseteq \{p, q\}$.

Ein Manhattan-Netzwerk M_S ist ein minimales Manhattan-Netzwerk*, wenn gilt:*

$$\forall MN\ \tilde{M}_S : \|M_S\| \leq \|\tilde{M}_S\|.$$

Man beachte, dass die Manhattan-Wege auch aus Teilstrecken der Strecken in M_S bestehen dürfen. Bedingung (i) ist die entscheidende Forderung. Man macht sich leicht klar, dass sich jede Menge von Strecken durch geeignetes Aufspalten der Strecken so transformieren lässt, dass die Bedingungen (ii) und (iii) erfüllt sind. Wir nehmen im Weiteren stillschweigend an, dass diese Transformation immer durchgeführt wird, wenn sie nötig ist. Folglich können wir uns beim Nachweis der Manhattan-Eigenschaft auf die Bedingung (i) beschränken. Manhattan-Netzwerke können auch über Graphen in der Ebene definiert werden. Dabei werden nur horizontale und vertikale Kanten erlaubt und das „Gewicht“ einer Kante ist ihre euklidische Länge. Ein Manhattan-Weg zwischen zwei Kanten (Punkten) entspricht dann einem Pfad im graphentheoretischen Sinne. Für eine genauere Ausführung siehe z. B. [GLN01].

Mit diesem formellen Rüstzeug im Rücken können wir uns jetzt einigen Aussagen über Manhattan-Netzwerke widmen, die das Verständnis für Manhattan-Netzwerke vertiefen und für die weiteren Kapitel von Nutzen sind.

3.2 Beispiele und nützliche Eigenschaften

Eine erste, leicht einzusehende Beobachtung ist, dass es keine Schwierigkeit darstellt, ein Manhattan-Netzwerk zu einer gegebenen (endlichen) Menge S von Punkten zu finden. Eine Möglichkeit besteht z. B. darin, das von S *induzierte Gitter* zu verwenden. Dazu werden als erstes die *induzierten Gitterpunkte* einer Menge von Punkten eingeführt.

Definition 3.6 (induzierte Gitterpunkte) *Es sei S eine endliche Menge von Punkten. Die* induzierten Gitterpunkte *bezüglich S sind die Elemente der (endlichen) Menge*

$$V_S := \left\{ q \in \mathbb{R}^2 \mid q^x \in S^x \ \wedge\ q^y \in S^y \right\}.$$

Das induzierte Gitter besteht nun aus allen Strecken zwischen zwei *benachbarten* Punkten aus der Menge der induzierten Gitterpunkte (siehe Abbildung 3.2).

Definition 3.7 (induziertes Gitter) *Es sei S eine endliche Menge von Punkten. Das* induzierte Gitter *bezüglich S ist die (endliche) Menge von Strecken*

$$E_S := \Big\{ s_{p,q} \mid p, q \in V_S,\ p \neq q,\ p^x = q^x \vee p^y = q^y,\ s_{p,q} \cap V_S = \{p, q\} \Big\}.$$

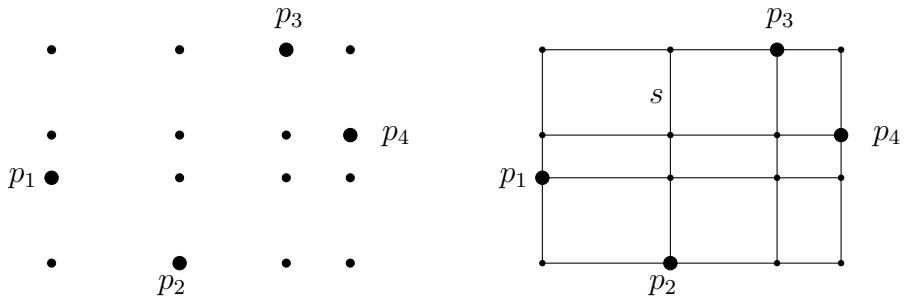


Abbildung 3.2: V_S und E_S für $S = \{p_1, \ldots, p_4\}$.

Offensichtlich ist E_S ein Manhattan-Netzwerk. Die interessante Aufgabe ist also, ein minimales Manhattan-Netzwerk oder zumindest ein Manhattan-Netzwerk mit „möglichst geringer“ Länge zu finden. Dass E_S hierzu ungeeignet ist, macht Abbildung 3.3 deutlich: In dem dort angegebenen Beispiel liegt die Länge eines minimalen Manhattan-Netzwerkes in $\Theta(|S|)$, die Länge von E_S aber in $\Theta(|S|^2)$.

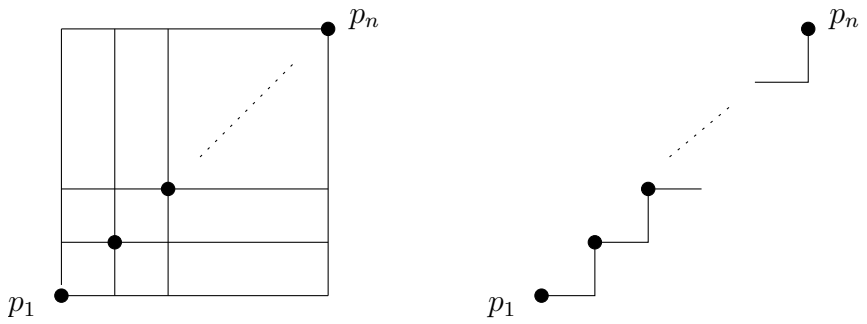


Abbildung 3.3: E_S vs. minimales Manhattan-Netzwerk für $S = \{p_1, \ldots, p_n\}$.

Als nächstes stellen wir eine sehr angenehme Beobachtung vor, die es erlaubt, sich auf eine endliche Teilmenge aller möglichen Manhattan-Netzwerke bezüglich einer Menge S zu beschränken. Sie besagt im Grunde genommen, dass es genügt, nur die (endlich vielen) Manhattan-Netzwerke zu betrachten, deren Strecken vollständig auf dem induzierten Gitter (bezüglich S) liegen. Für alle anderen Manhattan-Netzwerke M_S gibt es nämlich ein Manhattan-Netzwerk mit dieser Eigenschaft, dessen Länge höchstens so groß ist wie die Länge von M_S.

Satz 3.8 *Es sei S eine endliche Menge von Punkten. Dann gilt:*

$$\forall MN\ M_S : \exists MN\ \tilde{M}_S \subseteq E_S : \|\tilde{M}_S\| \leq \|M_S\|.$$

Der Beweis besteht aus zwei Teilen: Zuerst verschieben wir die horizontalen Strecken, sodass sie jeweils auf der Höhe eines Punktes aus S liegen. Danach verschieben wir die vertikalen Strecken, sodass sie jeweils auf der Breite eines Punktes aus S liegen (ohne die horizontalen Strecken wieder zu verschieben). Da beide Teile komplett analog verlaufen, beschränken wir uns darauf, nur den ersten Teil zu behandeln. Dies spiegelt sich auch in den Vorüberlegungen wieder, die eine gewisse Asymmetrie aufweisen. Für den Rest dieses Kapitels sei S eine fest gewählte, endliche Menge von Punkten. Alle Manhattan-Netzwerke, die im Weiteren vorkommen, sind bezüglich S zu verstehen. Zusätzlich nehmen wir an, dass alle Manhattan-Netzwerke folgende Eigenschaften besitzen: Sie enthalten keine Strecken, die nicht für den Nachweis der Manhattan-Eigenschaft gebraucht werden, und ihre Strecken sind nicht „unnötig fein“ aufgespalten. Damit ist gemeint, dass jeder Eckpunkt einer Strecke in S enthalten ist oder zugleich Eckpunkt einer zweiten Strecke *anderer Ausrichtung* ist. Offensichtlich lässt sich jedes Manhattan-Netzwerke so transformieren, dass die Eigenschaften erfüllt sind. Wir nehmen im Weiteren stillschweigend an, dass diese Transformation immer durchgeführt wird, wenn sie nötig ist.

Unser Ziel ist es, alle horizontalen Strecken eines Manhattan-Netzwerkes M, die nicht auf der Höhe eines Punktes aus S liegen, so zu verschieben, dass sie auf der Höhe eines Punktes aus S liegen. Dies erreichen wir durch eine Transformation von M, die wir iterativ anwenden, bis unser Ziel erreicht ist. Die Transformation ist das Thema der nächsten Seiten, wir wollen sie an dieser Stelle aber schon einmal kurz skizzieren: Wir betrachten alle Strecken auf einer Höhe, auf der kein Punkt aus S liegt. Diese Strecken verschieben wir entweder nach oben oder nach unten auf die Höhe eines Punktes aus S. Damit unser transformiertes Manhattan-Netzwerk auch weiterhin die Manhattan-Eigenschaft besitzt und seine Länge nicht vergrößert wird, müssen wir noch einige vertikalen Strecken entsprechend verlängern oder verkürzen. Dies sind gerade die Strecken, die – vor der Verschiebung – unsere horizontalen Strecken berührt haben. Wir werden nur den Fall erklären, dass die horizontalen Strecken nach oben verschoben werden müssen, der andere Fall verläuft analog.

Für die Beweisführung brauchen wir noch einige Notationen, die wir zuerst umgangssprachlich erklären. Die Menge M' besteht aus allen horizontalen Strecken aus M, die nicht auf der Höhe eines Punktes aus S liegen. Für ein $s \in M'$ besteht die Menge M_s^{g} aus s und allen horizontalen Strecken auf *g*leicher Höhe. Die Menge M_s^{o} (M_s^{u}) besteht aus allen vertikalen Strecken in M, die mit einer Strecke aus M_s^{g} einen Endpunkt gemeinsam haben und deren zweiter Endpunkt *o*berhalb (*u*nterhalb) von s liegt. Der Wert $M_s^{\min}$ bezeichnet den kleinsten y-Wert der Endpunkte aus M_s^{o}, der *echt* oberhalb von s liegt. Da die Strecken in M_s^{o} ihren unteren Endpunkt alle auf der gleichen Höhe haben, entspricht $M_s^{\min}$ der Höhe des oberen Endpunktes der kürzesten Strecke in M_s^{o}. Zur Illustration siehe auch Abbildung 3.4. Man beachte, dass der Index s hier nicht die Menge S bezeichnet, sondern die zu verschiebende Strecke s.

Definition 3.9 *Es sei M ein Manhattan-Netzwerk. Wir definieren:*

$$M' := \left\{ s_{p,q} \in M^{\mathrm{hor}} \mid p^y (= q^y) \notin S^y \right\}.$$

Für $s := s_{p,q} \in M'$ sei

- $M_s^{\mathrm{g}} := \left\{ s_{t,u} \in M^{\mathrm{hor}} \mid t^y (= u^y) = p^y \right\}$,

- $M_s^{\mathrm{o}} := \Big\{ s_{t,u} \in M^{\mathrm{ver}} \mid u^y > t^y = p^y \ \wedge\ \exists \tilde{p} \in \mathbb{R}^2 : s_{\tilde{p},t} \in M_s^{\mathrm{g}} \Big\}$,
- $M_s^{\mathrm{u}} := \Big\{ s_{t,u} \in M^{\mathrm{ver}} \mid u^y < t^y = p^y \ \wedge\ \exists \tilde{p} \in \mathbb{R}^2 : s_{\tilde{p},t} \in M_s^{\mathrm{g}} \Big\}$,
- $M_s^{\min} := \min\Big\{ y \in \mathbb{R} \mid y > p^y \ \wedge\ \exists s_{t,u} \in M_s^{\mathrm{o}} : y = t^y \Big\}$.

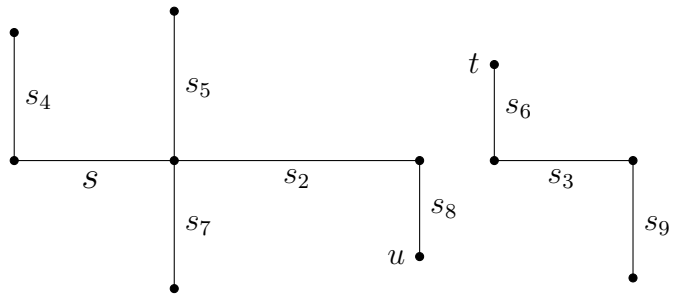


Abbildung 3.4: $M_s^{\mathrm{g}} = \{s, s_2, s_3\}$, $M_s^{\mathrm{o}} = \{s_4, s_5, s_6\}$, $M_s^{\mathrm{u}} = \{s_7, s_8, s_9\}$, $M_s^{\min} = t^y$.

Es muss jeweils einen Punkt aus S geben, der ober- bzw. unterhalb der Höhe von s liegt, ansonsten wären die Strecken aus M_s^{g} unnötig (auf der Höhe von s liegt ja kein Punkt aus S). Folglich muss es auch vertikale Strecken geben, die die Strecken aus M_s^{g} von unten und von oben „anschließen“. Dies kann nur an den Endpunkten der Strecken geschehen. Daraus folgt, dass die beiden Mengen M_s^{o} und M_s^{u} nicht leer sind. Insbesondere sichert dies die Existenz von $M_s^{\min}$.

Für die Strecken in M_s^{g}, M_s^{o} und M_s^{u} wollen wir nun die folgenden Änderungen durchführen: Die Strecken aus M_s^{g} sollen nach oben auf die Höhe $M_s^{\min}$ verschoben werden und die Strecken aus M_s^{o} (M_s^{u}) sollen so verkürzt (verlängert) werden, dass sich ihre unteren (oberen) Endpunkte ebenfalls auf der Höhe $M_s^{\min}$ befinden (siehe auch Abbildung 3.5).

Definition 3.10 *Es sei M ein Manhattan-Netzwerk und $s := s_{p,q} \in M'$ eine Strecke. Die Mengen M_s^{g}, M_s^{o} und M_s^{u} seien von der Gestalt:*

- $M_s^{\mathrm{g}} = \Big\{ s_{p_1,q_1}, \ldots, s_{p_a,q_a} \Big\} \quad mit \quad p_i^y = q_i^y = p^y \quad \forall j \in \{1, \ldots, a\}$,
- $M_s^{\mathrm{o}} = \Big\{ s_{t_1,u_1}, \ldots, s_{t_b,u_b} \Big\} \quad mit \quad t_i^y = p^y < u_i^y \quad \forall j \in \{1, \ldots, b\}$,
- $M_s^{\mathrm{u}} = \Big\{ s_{v_1,w_1}, \ldots, s_{v_c,w_c} \Big\} \quad mit \quad v_i^y = p^y > w_i^y \quad \forall j \in \{1, \ldots, c\}$.

Wir definieren folgende Mengen von Strecken:

- $M_s^{\tilde{\mathrm{g}}} := \Big\{ s_{\tilde{p}_1,\tilde{q}_1}, \ldots, s_{\tilde{p}_a,\tilde{q}_a} \Big\} \quad mit$
  $\tilde{p}_j := (p_j^x, M_s^{\min}), \quad \tilde{q}_j := (q_j^x, M_s^{\min}) \quad \forall j \in \{1, \ldots, a\}$,
- $M_s^{\tilde{\mathrm{o}}} := \Big\{ s_{\tilde{t}_1,u_1}, \ldots, s_{\tilde{t}_b,u_b} \Big\} \quad mit \quad \tilde{t}_j := (t_j^x, M_s^{\min}) \quad \forall j \in \{1, \ldots, b\}$,

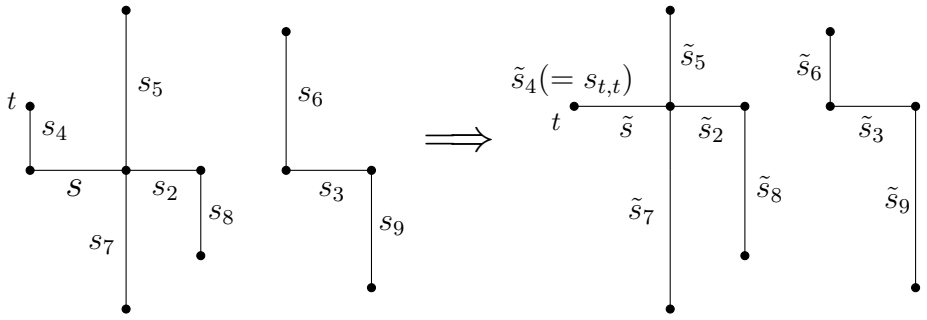


Abbildung 3.5: $M_s^{\tilde{g}} = \{\tilde{s}, \tilde{s}_2, \tilde{s}_3\}$, $M_s^{\tilde{o}} = \{\tilde{s}_4, \tilde{s}_5, \tilde{s}_6\}$, $M_s^{\tilde{u}} = \{\tilde{s}_7, \tilde{s}_8, \tilde{s}_9\}$.

- $M_s^{\tilde{u}} := \left\{ s_{\tilde{v}_1,w_1}, \ldots, s_{\tilde{v}_c,w_c} \right\}$ *mit* $\tilde{v}_j := (v_j^x, M_s^{\min})$ $\forall j \in \{1, \ldots, c\}$.

Es gibt eine kanonische Bijektion zwischen M_s^{g} und $M_s^{\tilde{g}}$, M_s^{o} und $M_s^{\tilde{o}}$ sowie M_s^{u} und $M_s^{\tilde{u}}$. Wir können also jeder Strecke in $M_s^{\tilde{g}}$, $M_s^{\tilde{o}}$ bzw. $M_s^{\tilde{u}}$ eine Ursprungsstrecke aus M_s^{g}, M_s^{o} bzw. M_s^{u} zuordnen. Die Strecken in $M_s^{\tilde{g}}$ sind alle genau so lang wie ihre jeweiligen Ursprungsstrecken aus M_s^{g}. Die Strecken wurden ja nur (um $M_s^{\min} - p^y > 0$) nach oben auf die Höhe $M_s^{\min}$ verschoben (zur Erinnerung: $s := s_{p,q}$). Die Strecken in $M_s^{\tilde{u}}$ sind alle genau um $(M_s^{\min} - p^y)$ länger als ihre jeweiligen Ursprungsstrecken aus M_s^{u}. Die Strecken in $M_s^{\tilde{o}}$ sind alle genau um $(M_s^{\min} - p^y)$ kürzer als ihre jeweiligen Ursprungsstrecken aus M_s^{o}. Dabei wurde $M_s^{\min}$ definitionsgemäß so gewählt, dass alle Strecken aus M_s^{o} mindestens die Länge $(M_s^{\min} - p^y)$ besitzen. Eine Strecke aus M_s^{o} besitzt sogar genau die Länge $(M_s^{\min} - p^y)$, folglich besitzt $M_s^{\tilde{o}}$ eine Strecke der Form $s_{p,p}$ (siehe auch Abbildung 3.5).

Lemma 3.11 *Es sei M ein Manhattan-Netzwerk und $s := s_{p,q} \in M'$ eine Strecke. Dann gilt:*

(i) $\tilde{M}(s) := M \backslash \left(M_s^{g} \cup M_s^{o} \cup M_s^{u} \right) \cup \left(M_s^{\tilde{g}} \cup M_s^{\tilde{o}} \cup M_s^{\tilde{u}} \right)$ *ist ein Manhattan-Netzwerk,*

(ii) $\|\tilde{M}(s)\| \leq \|M\| - \left(|M_s^{o}| - |M_s^{u}| \right) \left(M_s^{\min} - p^y \right)$,

(iii) $|\tilde{M}(s)'| < |M'|$,

(iv) $\tilde{B}(M(s)) \subseteq B(M)$.

Beweis: Es seien a und b zwei Punkte aus S und $W_{a,b} := \{s_{a_1,b_1}, \ldots, s_{a_m,b_m}\}$ ein Manhattan-Weg zwischen a und b. Wir nehmen o.B.d.A. an, dass $W_{a,b}$ eine Teilmenge von M ist. Dabei seien die Strecken in $\{s_{a_1,b_1}, \ldots, s_{a_m,b_m}\}$ wie in Definition 3.4 angeordnet und bezeichnet (d. h. es gilt $a_1 = a$, $b_m = b$, $a_j^x = b_j^x \vee a_j^y = b_j^y$ $\forall j \in \{1, \ldots, m\}$, $b_j = a_{j+1}$ $\forall j \in \{1, \ldots, m-1\}$ und $\|W_{a,b}\| = d(a, b)$). Um zu beweisen, dass $\tilde{M}(s)$ ein Manhattan-Netzwerk ist, genügt es zu zeigen, dass es auch einen Manhattan-Weg zwischen a und b gibt, der eine Teilmenge von $\tilde{M}(s)$ ist. Dabei erinnere man sich daran, dass auf der Höhe von s keine Punkte aus S – also insbesondere auch nicht a oder b – liegen.

1. Fall: Es gibt keine Strecke in $W_{a,b}$, die auch in M_s^{o} enthalten ist. Folglich kann es keine Strecken in $W_{a,b}$ geben, die auch in M_s^{g} oder M_s^{u} enthalten sind. Also ist $W_{a,b}$ schon eine Teilmenge von $\tilde{M}$.

2. Fall: Es gibt ein $k \in \{1, \ldots, m\}$, sodass s_{a_k,b_k} auch in M_s^{o} enthalten ist. Wegen der Längenrestriktion bei Manhattan-Wegen kann es nur eine solche Strecke geben. Folglich muss es auch genau eine Strecke $s_{a_l,b_l} \in W_{a,b}$ geben, die auch in M_s^{u} enthalten ist (es gelte o.B.d.A. $k < l \leq m$). Falls $W_{a,b}$ Strecken aus M_s^{g} enthält, so müssen diese zwischen k und l liegen, d. h. es gilt $M_s^{\mathrm{g}} \cap W_{a,b} = \{s_{a_{k+1},b_{k+1}}, \ldots, s_{a_{l-1},b_{l-1}}\}$. Wenn wir jetzt $s_{a_k,b_k}, \ldots, s_{a_l,b_l}$ durch die ihnen zugeordneten Strecken aus $M_s^{\tilde{\mathrm{g}}}$, $M_s^{\tilde{\mathrm{o}}}$ bzw. $M_s^{\tilde{\mathrm{u}}}$ ersetzen, so prüft man leicht nach, dass das Ergebnis wieder ein Manhattan-Weg ist.

Die Abschätzung der Länge in (ii) folgt unmittelbar aus den Bemerkungen über die Bijektionen und Längen der Strecken in dem Abschnitt direkt vor diesem Satz. Das „$\leq$“ kommt daher, dass Strecken aus $M_s^{\tilde{\mathrm{g}}}$ schon in $M \backslash (M_s^{\mathrm{g}} \cup M_s^{\mathrm{o}} \cup M_s^{\mathrm{u}})$ enthalten sein können. Für Strecken aus $M_s^{\tilde{\mathrm{o}}}$ und $M_s^{\tilde{\mathrm{u}}}$ ist dies nicht möglich, da sie sich mit ihren Ursprungsstrecken aus M_s^{o} bzw. M_s^{u} überlappen.

Die Strecken aus $M_s^{\tilde{\mathrm{g}}}$ sind die einzigen horizontalen Strecken in $\tilde{M}(s)$, die nicht (notwendigerweise) in M enthalten sind. Sie liegen definitionsgemäß auf der Höhe des oberen Endpunktes q einer vertikalen Strecke aus M_s^{o}. Nach den vorausgesetzten Eigenschaften von M muss gelten: Der Punkt q ist in S enthalten oder q ist Endpunkt einer horizontalen Strecke aus M. Also liegen die Strecken aus $M_s^{\tilde{\mathrm{g}}}$ auf der Höhe eines Punktes aus S oder einer horizontalen Strecke aus M. Auf der Höhe von s liegen in $\tilde{M}(s)$ keine Strecken mehr, diese wurden ja verschoben. Dies beweist (iii).

Für den Nachweis von (iii) sind nur die Strecken aus $M_s^{\tilde{\mathrm{o}}}$ und $M_s^{\tilde{\mathrm{u}}}$ zu betrachten, da dies die einzigen vertikalen Strecken in $\tilde{M}(s)$ sind, die nicht auch zu M gehören. Da diese Strecken aber verkürzte bzw. verlängerte Strecken von M_s^{o} bzw. M_s^{u} sind und somit die gleiche Lage haben, folgt die Aussage unmittelbar. ■

Die Schreibweise $\tilde{M}(s)$ soll verdeutlichen, dass man $\tilde{M}(\cdot)$ auch als eine Art Funktion verstehen kann, die M – abhängig von s – in ein anderes Manhattan-Netzwerk transformiert. Dabei werden s und die angrenzenden Strecken (d. h. M_s^{g}) um $(M_s^{\min} - p^y)$ nach oben verschoben und die an sie anschließenden vertikalen Strecken (d. h. M_s^{o} und M_s^{u}) werden dementsprechend in der Länge angepasst. Mit diesem Rüstzeug können wir uns jetzt dem Beweis von Satz 3.8 widmen.

Beweis von Satz 3.8: Ein Manhattan-Netzwerk, welches eine Teilmenge des induzierten Gitters E_S ist, darf insbesondere keine horizontalen Strecken enthalten, die nicht auf der Höhe eines Punktes aus S liegen. Deshalb werden wir M mit Hilfe von Lemma 3.11 so transformieren, dass diese notwendige Bedingung erfüllt ist.

Es sei $s \in M'$ eine Strecke (wenn $M' = \emptyset$ gilt, gehe zum nächsten Schritt im Beweis). Es gelte o.B.d.A. $|M_s^{\mathrm{o}}| \geq |M_s^{\mathrm{u}}|$ (anderenfalls schieben wir die Strecken aus M_s^{g} nach unten). Wir bilden das Manhattan-Netzwerk $\tilde{M} := \tilde{M}(s)$ wie in Lemma 3.11 (i) beschrieben. Aus Lemma 3.11 (ii) folgt sofort, dass $\|\tilde{M}\| \leq \|M\|$ gilt.

Diesen Vorgang wiederholen wir – natürlich mit dem jeweiligen Zwischenergebnis $\tilde{M}$ statt M – so lange, bis $\tilde{M}'$ die leere Menge ist. Lemma 3.11 (iii) garantiert uns, dass dieses Vorgehen nach endlich vielen Wiederholungen terminiert. Danach beseitigen wir völlig analog die vertikalen Strecken, die nicht auf der x-Koordinate eines

Punktes aus S liegen. Aus Lemma 3.11 (iv) – in seiner analogen Form für vertikale Strecken – folgt, dass die zweite Transformation nicht die wesentlichen Ergebnisse der ersten Transformation zerstört. Da die Endpunkte der Strecken in $\tilde{M}$ in S liegen oder Endpunkte von einer horizontalen *und* einer vertikalen Strecke in $\tilde{M}$ sind, liegen alle Endpunkte in V_S. Daraus folgt die Behauptung (gegebenenfalls müssen die Strecken noch geeignet aufgespalten werden). ■

Satz 3.8 kann auch direkt dazu benutzt werden, eine wichtige Frage zu beantworten, die wir bisher geschickt umgangen haben.

Korollar 3.12 *Zu jeder endlichen Menge S von Punkten gibt es ein minimales Manhattan-Netzwerk.*

Beweis: Es gelte o.B.d.A. $|S| \geq 2$. Nach Satz 3.8 gibt es nur endlich viele Manhattan-Netzwerke, die betrachtet werden müssen (alle anderen sind ja mindestens so groß wie ein Manhattan-Netzwerk aus dieser Menge). Bei einer endliche Menge, die nicht die leere Menge ist, wird aber immer ein Minimum angenommen. ■

Dabei ist sehr leicht einzusehen, dass es im Allgemeinen kein eindeutiges minimales Manhattan-Netzwerk bezüglich einer Menge von Punkten gibt. Als Beispiel betrachte man eine Menge mit genau zwei Punkten, die nicht auf derselben horizontalen oder vertikalen Gerade liegen.

3.3 Über die Approximation minimaler Manhattan-Netzwerke

Die Frage nach der Komplexität des Problems „finde ein minimales Manhattan-Netzwerk zu einer gegebenen Menge von Punkten“ ist leider noch unbeantwortet. So wurde weder ein Algorithmus mit – bezüglich der Anzahl der Punkte – polynomieller Laufzeit gefunden, noch gelang es zu beweisen, dass das Problem NP-schwer ist. Das „Ausprobieren“, d. h. das Testen aller Teilmengen des induzierten Gitters, liefert zwar nach Satz 3.8 eine korrekte Lösung, hat aber offensichtlich eine exponentielle Laufzeit. Hier besteht also weiterhin Forschungsbedarf.

Durch die unbefriedigende Lage bei der Komplexität gewinnt die Frage nach der Möglichkeit, minimale Manhattan-Netzwerke wenigstens gut zu approximieren, natürlich an Bedeutung. Auch hier gibt es zuerst einmal eine Ernüchterung: Es ist – zumindest bis zum Zeitpunkt der Niederschrift dieser Arbeit – noch kein polynomielles Approximationsschema (PTAS) bekannt. Allerdings haben wir in Kapitel 2 gesehen, dass es eine Reihe von Approximationsalgorithmen mit unterschiedlichen Approximationsfaktoren und (asymptotischen) Laufzeiten gibt.

4. Ein Approximationsalgorithmus für Stufenpolygone

Der in diesem Kapitel vorgestellte Algorithmus ist ein Teil eines „größeren“ Algorithmus, der minimale Manhattan-Netzwerke approximiert. Der „große“ Algorithmus, den wir im Weiteren *ApproxMMN* nennen wollen, wird in [BWW04a] beschrieben. Er besitzt einen Approximationsfaktor von 3 und seine Laufzeit liegt in $O(n \log n)$. Das Teilproblem, welches unser Algorithmus behandelt, lässt sich allerdings sehr gut separieren und als eigenständiges Problem beschreiben; dabei werden keine Kenntnisse über den Algorithmus *ApproxMMN* benötigt. Deshalb werden wir – zumindest in diesem Kapitel – nicht auf *ApproxMMN* eingehen.

Im ersten Teil wird das grundsätzliche Problem beschrieben. Die Aufgabe besteht darin, einen Algorithmus zu finden, der dieses Problem löst und zusätzlich zwei Nebenbedingungen erfüllt: Seine Laufzeit soll in $O(n \log n)$ liegen und seine Lösung höchstens zweimal so groß wie eine optimale Lösung sein. Die Nebenbedingungen leiten sich aus den Anforderungen an den oben erwähnten Algorithmus *ApproxMMN* ab. Im zweiten Teil präsentieren wir unseren Algorithmus, der die Aufgabe aus dem ersten Teil löst. Zuerst beschreiben wir den Algorithmus und beweisen, dass sein Ergebnis eine korrekte Lösung des Problems unter den gestellten Anforderungen liefert. Danach widmen wir uns dem Beweis, dass das Ergebnis höchstens zweimal die Länge einer minimalen Lösung besitzt. Letztendlich geben wir zwei Implementierungen an, deren Laufzeiten in $O(n \log n)$ bzw. sogar in $O(n)$ liegen. Die erste Implementierung ist relativ einfach und wird in dem Algorithmus *ApproxMMN* verwendet; die zweite Implementierung ist etwas komplizierter, hat aber ein besseres asymptotisches Laufzeitverhalten.

4.1 Die Aufgabe

Um die Aufgabe beschreiben zu können, brauchen wir als erstes eine Sequenz von Punkten, die gewisse Eigenschaften bezüglich der Lage ihrer Punkte besitzt.

Definition 4.1 (Stufenpolygonsequenz) *Es sei $S := (p_0, p_1, \ldots, p_n)$ eine endliche Sequenz von Punkten. Die Sequenz S ist eine* Stufenpolygonsequenz*, wenn gilt:*

(i) $p_0 < p_j \qquad \forall j \in \{1, \ldots, n\},$

(ii) $p_j^x > p_{j+1}^x \ \wedge \ p_j^y < p_{j+1}^y \qquad \forall j \in \{1, \ldots, n-1\}.$

Wir bezeichnen p_0 auch als den *Eckpunkt* von S. Als nächstes definieren wir das *Manhattan-Polygon*. Wir beschränken uns auf geschlossene Streckenzüge, die nur aus Manhattan-Strecken bestehen (siehe Abbildung 4.1).

Definition 4.2 (Manhattan-Polygon) *Es sei $S := (p_1, \ldots, p_n)$ eine endliche Sequenz von Punkten, für die gilt:*

(i) $p_j^x = p_{j+1}^x \ \vee \ p_j^y = p_{j+1}^y \quad \forall j \in \{1, \ldots, n-1\},$

(ii) $p_n^x = p_1^x \ \vee \ p_n^y = p_1^y.$

Das Manhattan-Polygon Q_S *bezüglich S ist die Menge $\{s_1, \ldots, s_n\}$ von Manhattan-Strecken, wobei $s_j := s_{p_j, p_{j+1}}$ $(j = 1, \ldots, n-1)$ und $s_n := s_{p_n, p_1}$ ist.*

Wie schon bei den Manhattan-Strecken lassen wir den Zusatz „Manhattan“ meistens der Bequemlichkeit halber weg, da in dieser Arbeit ausschließlich Manhattan-Polygone vorkommen. Die Strecken in Q_S sind offensichtlich alle wohldefiniert.

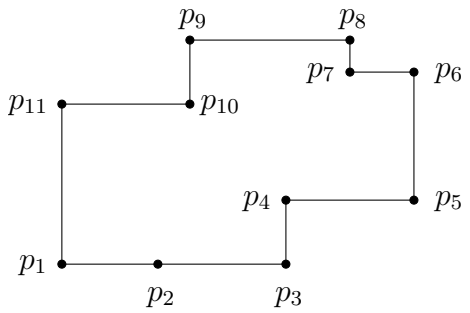


Abbildung 4.1: Das Manhattan-Polygon Q_S für $S = (p_1, \ldots, p_{11})$.

Jetzt haben wir alles beisammen, um den Begriff *Stufenpolygon* definieren zu können. Abbildung 4.2 erklärt die relativ kompliziert auszudrückende formellen Bedingungen illustrativ.

Definition 4.3 (Stufenpolygon) *Es sei $S := (p_0, p_1, \ldots, p_n)$ eine Stufenpolygonsequenz. Das* Stufenpolygon P_S *bezüglich S ist das Manhattan-Polygon bezüglich der Sequenz $\tilde{S} := (p_0, q_0, p_1, q_1, \ldots, p_n, q_n)$, wobei $q_j := (p_j^x, p_{j+1}^y)$ $(j = 0, \ldots, n-1)$ und $q_n := (p_n^x, p_0^y)$ ist.*

Das Innere *von P_S ist kanonisch definiert als*

$$I(P_S) := \bigcup_{j=1}^{n} \{t \in \mathbb{R}^2 \mid p_0 \leq t \leq p_j\} \setminus \bigcup_{s \in P_S} s.$$

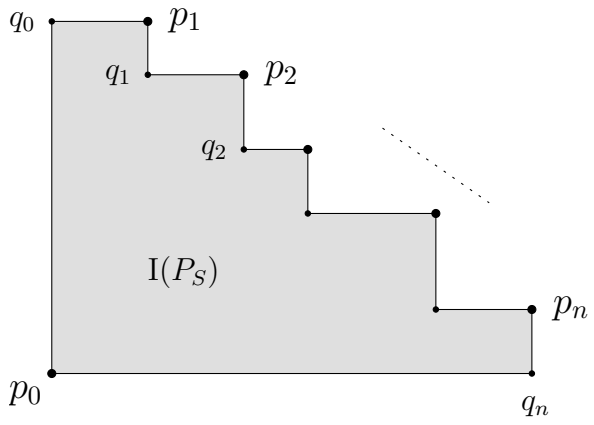


Abbildung 4.2: Das Stufenpolygon P_S für $S = (p_0, \ldots, p_n)$.

Die Aufgabe ist es, Strecken in P_S einzufügen, sodass es von den Punkten $p_1, \ldots, p_n$ jeweils einen Manhattan-Weg nach p_0 gibt. Dabei dürfen – zusätzlich zu den eingeführten Strecken – auch die Strecken von P_S benutzt werden. Eine Menge von Strecken, die dies erfüllt, nennen wir eine *Zerlegung* von P_S (siehe Abbildung 4.3). Von besonderem Interesse sind natürlich die Zerlegungen, bei denen die Gesamtlänge ihrer Strecken minimal ist, sie werden als *minimale Zerlegungen* bezeichnet.

Definition 4.4 (Zerlegung) *Es sei* $S := (p_0, p_1, \ldots, p_n)$ *eine Stufenpolygonsequenz und* P_S *das Stufenpolygon bezüglich* S*. Eine* Zerlegung Z_S *von* P_S *ist eine endliche Menge von Manhattan-Strecken, für die gilt:*

$$\forall j \in \{1, \ldots, n\} : \exists \textit{Manhattan-Weg } W_{p_j,p_0} : \forall s \in W_{p_j,p_0} : \exists \tilde{s} \in (Z_S \cup P_S) : s \subseteq \tilde{s}.$$

Eine Zerlegung Z_S *ist eine* minimale Zerlegung*, wenn gilt:*

$$\forall \textit{Zerlegungen } \tilde{Z}_S : \|Z_S\| \leq \|\tilde{Z}_S\|.$$

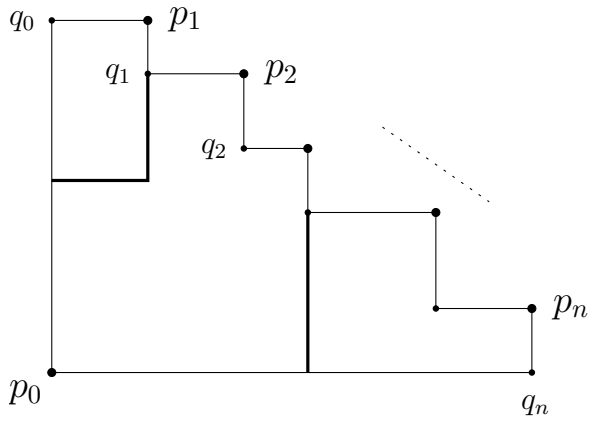


Abbildung 4.3: Eine mögliche Zerlegung von P_S.

Wie schon bei den Manhattan-Netzwerken stellt es offensichtlich keine Schwierigkeit dar, eine Zerlegung zu finden. Überhaupt hat die Definition der Zerlegung große

Ähnlichkeit mit der Definition 3.5 für Manhattan-Netzwerke. Der wesentliche Unterschied ist, dass bei der Zerlegung die Manhattan-Wege auch Teilstrecken aus P_S verwendet werden dürfen. Man hätte bei der Definition der Zerlegung auch fordern können, dass zwischen allen Paaren von Punkten aus S ein geeigneter Manhattan-Weg existieren muss. Dies wäre eine äquivalente Definition gewesen, denn die Paare $\{p_i, p_j\}$ $(1 \leq i \neq j \leq n)$ lassen sich durch Manhattan-Wege verbinden, die nur Strecken aus P_S enthalten. Es ist offensichtlich, dass eine minimale Zerlegung keine Teilstrecken aus P_S enthält. Wenn wir im Folgenden davon schreiben, dass ein Manhattan-Weg W in $Z_S \cup P_S$ liegt, so meinen wir damit, dass W ausschließlich Teilstrecken von Strecken aus Z_S oder aus P_S enthält.

Es existiert ein Algorithmus, der eine minimale Zerlegung in polynomieller Zeit findet. Er basiert auf dynamischer Programmierung und ist eine modifizierte Version eines Algorithmus, welcher in [LPRS82] beschrieben wird. Leider liegt seine Laufzeit in $\Theta(n^3)$, was für unsere Zwecke nicht schnell genug ist. Wir benötigen – wie bereits erwähnt – einen Algorithmus, dessen Laufzeit in $O(n \log n)$ liegt. Dafür muss er keine optimale Lösung liefern, es genügt eine 2-approximative Lösung. Dies bedeutet, dass der Algorithmus für jedes gegebene Stufenpolygon eine Zerlegung liefern muss, deren Länge höchstens zweimal die Länge einer minimalen Zerlegung beträgt. Ein Algorithmus, der diesen Anforderungen genügt, wird im Rest des Kapitels beschrieben.

4.2 Ein Faktor-2-Approximationsalgorithmus

4.2.1 Die Idee

Unser Algorithmus folgt dem Vorgehen des Algorithmus „A Thickest-First Rectangulation“ aus [GLN01]. Dieser berechnet für Stufenpolygone eine Rechteckzerlegung, die höchstens zweimal die Länge einer minimalen Rechteckzerlegung hat. Allerdings unterscheidet sich unser Algorithmus bei der Auswahl der Punkte – dem wesentlichen Schritt des Algorithmus – entscheidend von dem Algorithmus aus [GLN01]. Außerdem werden die Stufenpolygone leicht modifiziert, bevor sie dem Algorithmus aus [GLN01] als Eingabe übergeben werden, während unser Algorithmus die Stufenpolygone unverändert als Eingabe erhält.

Die Grundidee besteht darin, zwei aufeinander folgende Punkte aus dem Stufenpolygon auszuwählen und diese mit Hilfe einer horizontalen und einer vertikalen Strecke mit dem Eckpunkt des Stufenpolygons zu verbinden. Dadurch wird das Stufenpolygon und sein Inneres in (maximal) drei Teile aufgeteilt. Der mittlere Teil muss nicht weiter bearbeitet werden, die anderen beiden Teile lassen sich wieder als Stufenpolygone auffassen. Auf diese wird der Algorithmus rekursiv angewendet (siehe auch Abbildung 4.4). Das Ergebnis enthält alle Strecken, die in dem Algorithmus – inklusive seiner rekursiven Aufrufe – eingefügt werden. Die eigentliche Aufgabe besteht darin, die Auswahl dieser Punkte so geschickt zu wählen, dass der Algorithmus 2-approximativ ist.

4.2.2 Der Algorithmus

Damit wir den Algorithmus anschaulich erklären und seine gewünschten Eigenschaften verständlich beweisen können, definieren wir als erstes einige Punkte und Strecken. Zur Illustration der Bezeichnungen siehe Abbildung 4.5.

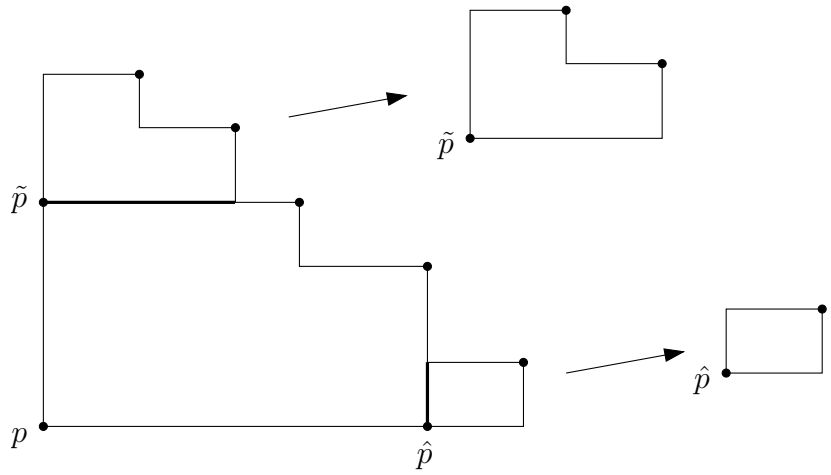


Abbildung 4.4: Grundprinzip des Algorithmus.

Definition 4.5 *Es sei* $S := (p_0, p_1, \ldots, p_n)$ *eine Stufenpolygonsequenz und* P_S *das zugehörige Stufenpolygon. Wir definieren*

- $\tilde{p}_j := (p_0^x, p_j^y) \quad$ *und* $\quad \hat{p}_j := (p_j^x, p_0^y) \qquad (j = 1, \ldots, n),$
- $s_{j,\mathrm{hor}} := s_{q_j, \tilde{p}_{j+1}} \quad$ *und* $\quad s_{j,\mathrm{ver}} := s_{q_j, \hat{p}_j} \qquad (j = 1, \ldots, n-1),$
- $a_j := \|s_{j,\mathrm{hor}}\| \quad$ *und* $\quad b_j := \|s_{j,\mathrm{ver}}\| \qquad (j = 1, \ldots, n-1).$

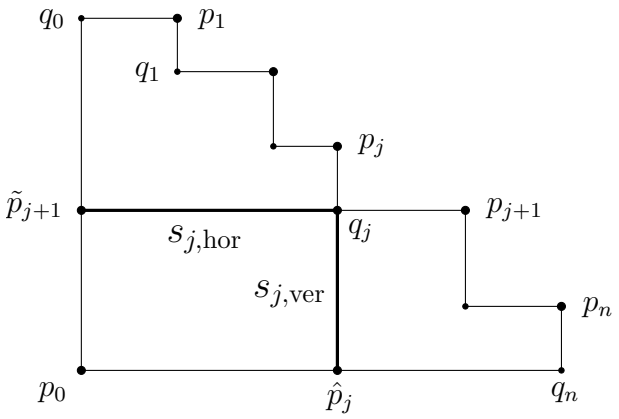


Abbildung 4.5: Bezeichnungen im Stufenpolygon.

Man prüft leicht nach, dass die neuen Strecken alle wohldefiniert sind und dass $a_1 < \ldots < a_{n-1}$ sowie $b_1 > \ldots > b_{n-1}$ gilt. Wir können nun den Pseudocode des Zerlegungsalgorithmus angeben.

Algorithmus Zerlege(S)
Eingabe: Stufenpolygonsequenz $S := (p_0, \ldots, p_n)$ mit $n > 0$
Ausgabe: eine Zerlegung des Stufenpolygons P_S
begin

$\mathcal{Z}^{\mathrm{c}} := \mathcal{Z}^{\mathrm{o}} := \mathcal{Z}^{\mathrm{r}} := \emptyset;$
if $n > 2$ **then** ($*$ Abbruch der Rekursion für $n \leq 2$ $*$)

$M := \{j \in \{1, \ldots, n-1\} \mid a_j \leq b_j\}$;
if $M = \emptyset$ **then** $(*$ d. h. $\forall j \in \{1, \ldots, n-1\} : a_j > b_j *)$
  $k := 1$
else $(* M \neq \emptyset *)$
  $m := \max(M)$;
  if $m = n-1$ **then**
    $k := n-1$
  else $(* 1 \leq m < n-1 *)$
    if $a_m \geq b_{m+1}$ **then**
      $k := m$
    else $(* a_m < b_{m+1} *)$
      $k := m+1$
if $k > 1$ **then**
  $\mathcal{Z}^{\mathrm{c}} := \{s_{k-1,\mathrm{hor}}\}$;
  $\mathcal{Z}^{\mathrm{o}} := \mathrm{Zerlege}\Big((\tilde{p}_k, p_1, \ldots, p_{k-1})\Big)$
if $k < n-1$ **then**
  $\mathcal{Z}^{\mathrm{c}} := \mathcal{Z}^{\mathrm{c}} \cup \{s_{k+1,\mathrm{ver}}\}$;
  $\mathcal{Z}^{\mathrm{r}} := \mathrm{Zerlege}\Big((\hat{p}_{k+1}, p_{k+2}, \ldots, p_n)\Big)$
return $\mathcal{Z}(S) := \mathcal{Z}^{\mathrm{c}} \cup \mathcal{Z}^{\mathrm{o}} \cup \mathcal{Z}^{\mathrm{r}}$

Für die folgenden Erläuterungen ist Abbildung 4.6 hilfreich. Die Rekursion bricht ab, wenn das (Rekursions-)Polygon weniger als drei Punkte hat, wobei der Eckpunkt nicht mitgezählt wird. In diesem Fall ist die leere Menge eine Zerlegung. Allgemein gilt, dass es für p_1 und p_n immer einen Manhattan-Weg nach p_0 gibt, der nur Strecken aus P_S enthält. Falls das Stufenpolygon aus mehr als drei Punkten besteht (d. h. $n > 2$), wird zuerst ein Punkt $p_k \in \{p_1, \ldots, p_{n-1}\}$ ausgewählt. Dies geschieht abhängig von den Längen a_j und b_j $(1 \leq j \leq n-1)$. Die Auswahl ist nicht intuitiv, ermöglicht uns aber später die Abschätzung der Länge. Als Nächstes wird durch das Einfügen der horizontalen Strecke $s_{k-1,\mathrm{hor}}$ dafür gesorgt, dass es einen Manhattan-Weg zwischen p_k und p_0 gibt, der in $\mathcal{Z}^m \cup P_S$ liegt. Dies ist allerdings nur dann nötig, wenn p_k nicht schon der oberste Punkt p_1 ist (s. o.). Wurde eine neue Strecke eingeführt, so lässt sich der obere Teil von P_S zusammen mit $s_{k-1,\mathrm{hor}}$ wieder als Stufenpolygon interpretieren. Auf dieses wird der Algorithmus rekursiv angewandt. Danach wird die vertikale Strecke $s_{k+1,\mathrm{ver}}$ eingefügt, sodass es einen Manhattan-Weg zwischen p_{k+1} und p_0 gibt, der in $\mathcal{Z}^m \cup P_S$ liegt. Zusammen mit $s_{k+1,\mathrm{ver}}$ lässt sich der rechte Teil von P_S wieder als Stufenpolygon interpretieren, welches ebenfalls rekursiv bearbeitet wird. Das Ganze wird analog nur dann ausgeführt, wenn p_{k+1} nicht schon der am weitesten rechts liegende Punkt p_n ist.

Offensichtlich terminiert der Algorithmus, denn die Rekursionstiefe beträgt höchstens n. Es bezeichne $\mathcal{Z}(S)$ das Ergebnis von *Zerlege(S)* bei der Eingabe S. Die Menge $\mathcal{Z}(S)$ ist eine Zerlegung von P_S, da im ersten Schritt die Punkte p_k und p_{k+1}

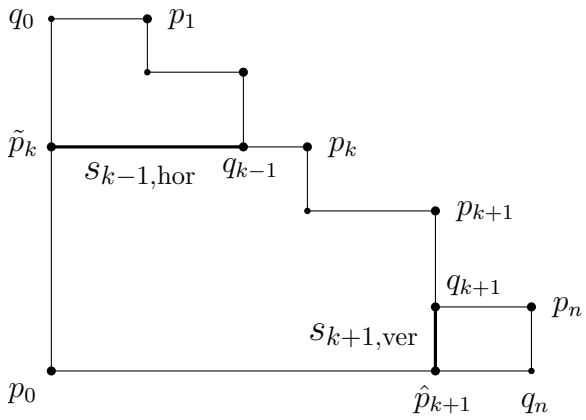


Abbildung 4.6: Illustration zum Algorithmus *Zerlegung*.

mit p_0 verbunden werden. In den beiden rekursiven Aufrufen werden dann die Punkte $p_1, \ldots, p_{k-1}$ und $p_{k+2}, \ldots, p_n$ mit $\tilde{p}_k$ bzw. $\hat{p}_{k+1}$ verbunden. Folglich gibt es Manhattan-Wege zwischen $p_1, \ldots, p_n$ und p_0, die in $\mathcal{Z}(S) \cup P_S$ liegen.

4.2.3 Der Beweis der Faktor-2-Approximation

Der Beweis, dass unser Algorithmus immer eine Zerlegung liefert, deren Länge höchstens zweimal die Länge einer minimalen Zerlegung beträgt, ist etwas aufwändiger. Er erstreckt sich über mehrere Lemmata, deren Ergebnisse dann in dem Satz 4.9 zusammengefasst werden. Auch hier führen wir zuerst einige neue Bezeichnungen ein, die uns das Leben erleichtern werden. Zur Illustration der Bezeichnungen siehe Abbildung 4.7.

Definition 4.6 *Es sei $S := (p_0, p_1, \ldots, p_n)$ eine Stufenpolygonsequenz mit $n > 1$ und P_S das zugehörige Stufenpolygon. Weiter sei $j \in \{1, \ldots, n-1\}$. Wir definieren:*

- $P_j^{\text{c}} := \Big\{ t \in I(P_S) \mid t^x \leq p_{j+1}^x \ \wedge \ t^y \leq p_j^y \Big\}$,
- $P_j^{\text{o}} := \Big\{ t \in I(P_S) \mid t^y > p_j^y \Big\}$,
- $P_j^{\text{r}} := \Big\{ t \in I(P_S) \mid t^x > p_{j+1}^x \Big\}$.

Es sei Z_S eine Zerlegung von P_S. Wir definieren weiterhin:

- $Z_j^{\text{c}} := \Big\{ s_{p,q} \in Z_S \mid s_{p,q} \backslash \{p, q\} \ \cap \ P_j^{\text{c}} \neq \emptyset \Big\}$,
- $Z_j^{\text{o}} := \Big\{ s_{p,q} \in Z_S \mid s_{p,q} \backslash \{p, q\} \ \cap \ P_j^{\text{o}} \neq \emptyset \Big\}$,
- $Z_j^{\text{r}} := \Big\{ s_{p,q} \in Z_S \mid s_{p,q} \backslash \{p, q\} \ \cap \ P_j^{\text{r}} \neq \emptyset \Big\}$.

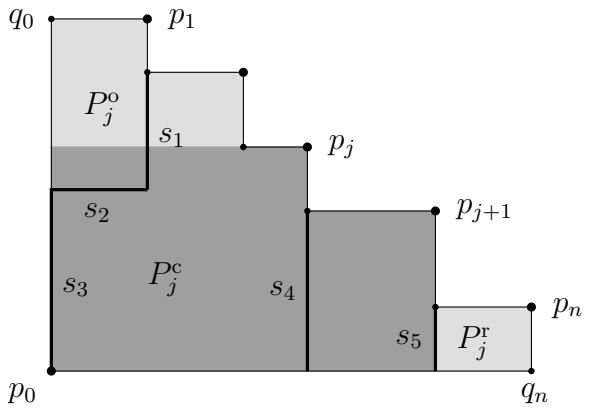


Abbildung 4.7: $Z_j^{\mathrm{c}} = \{s_1, s_2, s_4, s_5\}$, $Z_j^{\mathrm{o}} = \{s_1\}$, $Z_j^{\mathrm{r}} = \emptyset$.

Die Mengen P_j^{c}, P_j^{o} und P_j^{r} lassen sich als Flächen interpretieren. Sie bilden für ein festes $j \in \{1, \ldots, n-1\}$ eine Partitionierung des Inneren von P_S (d. h. $\forall j \in \{1, \ldots, n-1\} : \mathrm{I}(P_S) = P_j^{\mathrm{c}} \uplus P_j^{\mathrm{o}} \uplus P_j^{\mathrm{r}}$). Man beachte, dass P_1^{o} und P_{n-1}^{r} die leere Menge sind. Die Flächen P_j^{c}, P_j^{o} und P_j^{r} sollen uns dabei helfen, die Teilmengen Z_j^{c}, Z_j^{o} und Z_j^{r} einer Zerlegung Z_S anschaulich zu beschreiben. Diese Mengen bestehen jeweils aus allen Strecken der Zerlegung, deren Inneres teilweise in der entsprechenden Fläche P_j^{c}, P_j^{o} bzw. P_j^{r} enthalten ist. Die Namen der Mengen sind übrigens nicht rein zufällig an die Bezeichnungen aus dem Algorithmus angelehnt, wie man bei den späteren Abschätzungen erkennen wird.

Da es für ein festes $j \in \{1, \ldots, n-1\}$ keine zwei Punkte aus P_j^{o} bzw. P_j^{r} gibt, die die gleichen x- oder y-Werte haben, sind Z_j^{o} und Z_j^{r} disjunkt. Es kann aber durchaus Strecken in Z_S geben, die sowohl in Z_j^{c} als auch in Z_j^{o} oder Z_j^{r} liegen. Allerdings kann man die Strecken von Z_S offensichtlich so aufspalten, dass Z_j^{c}, Z_j^{o} und Z_j^{r} disjunkt sind. Wir können also im Folgenden o.B.d.A. annehmen, dass dies für jede Zerlegung der Fall ist. Weiter beachte man, dass Z_j^{c}, Z_j^{o} und Z_j^{r} für ein festes j im Allgemeinen nicht die gesamte Zerlegung überdecken müssen. Allerdings macht man sich leicht klar, dass alle Strecken in Z, die nicht zu Z_j^{c}, Z_j^{o} oder Z_j^{r} gehören, überflüssig sind. Es sind nämlich gerade die Strecken in Z, deren Inneres vollständig außerhalb des Inneren von P_S liegt.

Lemma 4.7 *Es sei $S := (p_0, p_1, \ldots, p_n)$ eine Stufenpolygonsequenz mit $n > 2$ und P_S das zugehörige Stufenpolygon. Weiter sei Z_S eine beliebige Zerlegung von P_S. Dann gilt:*

(i) $\forall k \in \{2, \ldots, n-2\} : \|Z_k^{\mathrm{c}}\| \geq \min\Big\{a_k, b_k, (a_{k-1} + b_{k+1})\Big\}$,

(ii) $\|Z_1^{\mathrm{c}}\| \geq \min\Big\{a_1, b_2\Big\}$,

(iii) $\|Z_{n-1}^{\mathrm{c}}\| \geq \min\Big\{b_{n-1}, a_{n-2}\Big\}$.

Beweis: Es sei $Z := Z_S$ eine Zerlegung von P_S. Weiter seien W_k und W_{k+1} Manhattan-Wege zwischen p_k bzw. p_{k+1} und p_0, die in $Z \cup P_S$ liegen.

„$k \in \{2, \ldots, n-2\}$“: Wir unterscheiden zwei Fälle:

1. Fall: Es gibt keinen Punkt außer p_0, der sowohl in einer Strecke aus W_k als auch in einer Strecke aus W_{k+1} enthalten ist, d. h. es gilt $\forall \tilde{s} \in W_k : \forall \hat{s} \in W_{k+1} : \tilde{s} \cap \hat{s} \subseteq \{p_0\}$. Dann muss W_k offensichtlich mindestens die Distanz a_{k-1} oder b_k in P_k^c zurücklegen. Analog muss W_{k+1} mindestens die Distanz a_k oder b_{k+1} in P_k^c zurücklegen. Damit folgt mit Hilfe der Monotonie der a_i bzw. der b_i die Abschätzung $\|Z_k^c\| \geq \min\{a_{k-1}, b_k\} + \min\{a_k, b_{k+1}\} \geq \min\{a_k, b_k, (a_{k-1} + b_{k+1})\}$.

2. Fall: Es sei $r \neq p_0$ ein Punkt, der sowohl in einer Strecke aus W_k als auch in einer Strecke aus W_{k+1} enthalten ist (d. h. $\exists \tilde{s} \in W_k : \exists \hat{s} \in W_{k+1} : r \in \tilde{s} \cap \hat{s}$). Da $r \in W_k$ und $r \in W_{k+1}$ gilt, folgt $p_0 < r \leq q_k$. Nach Wahl von r existiert (mindestens) ein Manhattan-Weg zwischen r und p_0, der in $Z \cup P_S$ liegt. Dieser muss mindestens die Distanz $(r^x - p_0^x)$ oder $(r^y - p_0^y)$ in P_k^c zurücklegen. Weiter gibt es natürlich Manhattan-Wege $\tilde{W}$ und $\hat{W}$ zwischen p_k bzw. p_{k+1} und r, die in $Z \cup P_S$ liegen. Dabei müssen die vertikalen Strecken in $\tilde{W}$ mindestens die Distanz $(q_k^y - r^y)$ in P_k^c zurücklegen. Analog müssen die horizontalen Strecken in $\hat{W}$ mindestens die Distanz $(q_k^x - r^x)$ in P_k^c zurücklegen. Man beachte dabei, dass alle von uns angegebenen Distanzen von unterschiedlichen Strecken zurückgelegt werden müssen. Zusammengefasst folgt $\|Z_k^c\| \geq \min\{r^x - p_0^x, r^y - p_0^y\} + (q_k^x - r^x) + (q_k^y - r^y) \geq \min\{q_k^x - p_0^x, q_k^y - p_0^y\} = \min\{a_k, b_k\} \geq \min\{a_k, b_k, (a_{k-1} + b_{k+1})\}$.

„$k = 1$“: Der Manhattan-Weg W_{k+1} muss offensichtlich mindestens die Distanz a_k oder b_{k+1} in P_k^c zurücklegen. Daraus folgt bereits unsere gewünschte Abschätzung.

„$k = n-1$“: Der Manhattan-Weg W_k muss offensichtlich mindestens die Distanz a_{k-1} oder b_k in P_k^c zurücklegen. Daraus folgt bereits unsere gewünschte Abschätzung. ■

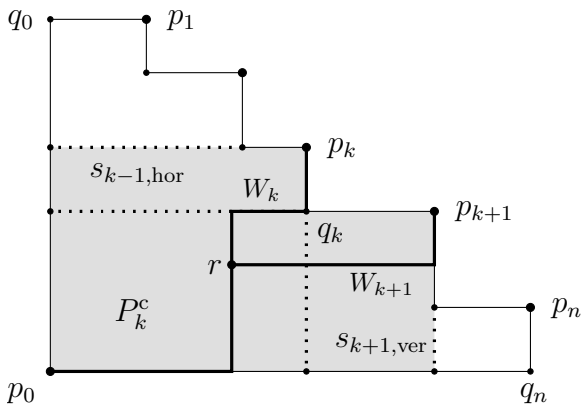


Abbildung 4.8: Zum Beweis von Lemma 4.7.

Das folgende Lemma macht – zusammen mit dem vorherigen Lemma – eine zentrale Aussage über die Länge der Zerlegung, die der Algorithmus *Zerlege* berechnet. Hier spielt zum ersten Mal die konkrete Auswahl von k im Algorithmus eine Rolle.

Lemma 4.8 *Es sei* $S := (p_0, p_1, \ldots, p_n)$ *eine Stufenpolygonsequenz mit* $n > 2$ *und* P_S *das zugehörige Stufenpolygon. Weiter seien die Werte* k *und* $\mathcal{Z}^c$ *definiert wie im Pseudocode von* Zerlege(S). *Dann gilt:*

(i) $k \in \{2, \ldots, n-2\} \implies \|\mathcal{Z}^c\| \leq 2 \min\Big\{a_k, b_k, (a_{k-1} + b_{k+1})\Big\}$,

(ii) $k = 1 \implies \|\mathcal{Z}^{\mathrm{c}}\| = \min\Big\{a_1, b_2\Big\}$,

(iii) $k = n-1 \implies \|\mathcal{Z}^{\mathrm{c}}\| = \min\Big\{b_{n-1}, a_{n-2}\Big\}$.

Beweis: Wie k und $\mathcal{Z}^{\mathrm{c}}$ seien auch M und m definiert wie im Pseudocode von *Zerlege(S)*. Man erinnere man sich, dass $a_1 < \ldots < a_{n-1}$ sowie $b_1 > \ldots > b_{n-1}$ gilt. Wir unterscheiden nach dem Zustandekommen von k:

„$M = \emptyset$“: Es gilt $k = 1$ und $a_1 > b_1$. Daraus folgt $\|\mathcal{Z}^{\mathrm{c}}\| = \|\{s_{2,\mathrm{ver}}\}\| = b_2 = \min\{a_1, b_2\}$.

„$m = n-1$“: Es gilt $k = n-1$ und $a_{n-1} \leq b_{n-1}$. Daraus folgt $\|\mathcal{Z}^{\mathrm{c}}\| = \|\{s_{n-2,\mathrm{hor}}\}\| = a_{n-2} = \min\{b_{n-1}, a_{n-2}\}$.

„$1 \leq m < n-1 \ \wedge \ a_m \geq b_{m+1}$“: Es gilt $k = m$ und $a_m \leq b_m$. Falls $m = 1$ gilt, so gilt $\|\mathcal{Z}^{\mathrm{c}}\| = \|\{s_{2,\mathrm{ver}}\}\| = b_2 = \min\{a_1, b_2\}$. Falls $s > m$ gilt, so gilt $a_{m-1} \leq \min\{a_m, b_m, (a_{m-1} + b_{m+1})\}$ und $b_{m+1} \leq \min\{a_m, b_m, (a_{m-1} + b_{m+1})\}$. Daraus folgt $\|\mathcal{Z}^{\mathrm{c}}\| = \|\{s_{k-1,\mathrm{hor}}, s_{k+1,\mathrm{ver}}\}\| = a_{k-1} + b_{k+1} \leq 2\min\{a_k, b_k, (a_{k-1} + b_{k+1})\}$.

„$1 \leq m < n-1 \ \wedge \ a_m < b_{m+1}$“: Es gilt $k = m+1$ und $a_{m+1} > b_{m+1}$. Falls $m = n-2$ gilt, so gilt $k = n-1$ und $\|\mathcal{Z}^{\mathrm{c}}\| = \|\{s_{s,\mathrm{hor}}\}\| = a_{n-2} = \min\{b_{n-1}, a_{n-2}\}$. Falls $m < n-2$ gilt, so gilt $a_m \leq \min\{a_{m+1}, b_{m+1}, (a_m + b_{m+2})\}$ und $b_{m+2} \leq \min\{a_{m+1}, b_{m+1}, (a_m + b_{m+2})\}$. Daraus folgt $\|\mathcal{Z}^{\mathrm{c}}\| = \|\{s_{k-1,\mathrm{hor}}, s_{k+1,\mathrm{ver}}\}\| = a_{k-1} + b_{k+1} \leq 2\min\{a_k, b_k, (a_{k-1} + b_{k+1})\}$. ■

Nun haben wir alles beisammen, was wir für den Beweis der Faktor-2-Approximation unseres Algorithmus benötigen.

Satz 4.9 *Es bezeichne $\mathcal{Z}(S)$ das Ergebnis des Algorithmus* Zerlege(S) *bei der Eingabe S. Dann gilt:*

$$\forall \textit{Stufenpolygonsequenzen } S : \forall \textit{Zerlegungen } Z_S : \|\mathcal{Z}(S)\| \leq 2\|Z_S\|.$$

Beweis: Es sei $S := (p_0, \ldots, p_n)$ eine Stufenpolygonsequenz und $Z := Z_S$ eine Zerlegung von P_S. Der Beweis wird per vollständiger Induktion über die Länge von S geführt.

$n \leq 2$: Es gilt $\mathcal{Z}(S) = \emptyset$ und damit $\|\mathcal{Z}(S)\| = 0 \leq 2\|Z\|$.

$n > 2$: Es seien k, $\mathcal{Z}^{\mathrm{c}}$, $\mathcal{Z}^{\mathrm{o}}$ und $\mathcal{Z}^{\mathrm{r}}$ definiert wie im Pseudocode von *Zerlege(S)*. Es gilt also $\mathcal{Z}(S) = \mathcal{Z}^{\mathrm{c}} \cup \mathcal{Z}^{\mathrm{o}} \cup \mathcal{Z}^{\mathrm{r}}$. Wir werden $\|\mathcal{Z}^l\| \leq 2\|Z_k^l\|$ für $l \in \{\mathrm{o}, \mathrm{c}, \mathrm{r}\}$ zeigen. Damit folgt $\|\mathcal{Z}\| \leq 2\|Z\|$.

„$l = m$“: Diese ist genau die Aussage, die man durch Kombination der beiden Lemmata 4.8 und 4.7 erhält.

„$l = o$“: Es gelte o.B.d.A. $k > 1$ (sonst gilt $\mathcal{Z}^{\mathrm{o}} = \emptyset$ und wir sind fertig). Weiter sei $P_{\bar{S}}$ das Stufenpolygon bezüglich $\bar{S} := (\tilde{p}_k, p_1, \ldots, p_{k-1})$. Es gilt $|\bar{S}| < |S|$. Laut Algorithmus ist $\mathcal{Z}^{\mathrm{o}}$ das Ergebnis des rekursiven Aufrufs von *Zerlege($\bar{S}$)*. Nach der Induktionsvoraussetzung gilt also $\|\mathcal{Z}^{\mathrm{o}}\| \leq 2\|Z_k^{\mathrm{o}}\|$, denn Z_k^{o} ist offensichtlich eine Zerlegung von $P_{\bar{S}}$.

„$l = r$“: Es gelte o.B.d.A. $k < n - 1$ (sonst gilt $\mathcal{Z}^{\mathrm{r}} = \emptyset$ und wir sind fertig). Weiter sei $P_{\check{S}}$ das Stufenpolygon bezüglich $\check{S} := (\hat{p}_{k+1}, p_{k+2}, \ldots, p_n)$. Es gilt $|\check{S}| < |S|$. Laut Algorithmus ist $\mathcal{Z}^{\mathrm{r}}$ das Ergebnis des rekursiven Aufrufs von *Zerlege($\check{S}$)*. Nach der Induktionsvoraussetzung gilt also $\|\mathcal{Z}^{\mathrm{r}}\| \leq 2\|Z_k^{\mathrm{r}}\|$, denn Z_k^{r} ist offensichtlich eine Zerlegung von $P_{\check{S}}$. ■

4.2.4 Eine $O(n \log n)$-Implementierung

Der im vorherigen Teil des Kapitels angegebene Pseudocode erlaubt – bis auf die Bestimmung des Maximums der Menge M – eine kanonische Implementierung. Wir ergänzen die noch fehlenden Angaben und zeigen, dass die Laufzeit der Implementierung in $O(n \log n)$ und ihr Speicherplatzbedarf in $O(n)$ liegt. Dabei sei $S := (p_0, \ldots, p_n)$ $(n > 0)$ die Stufenpolygonsequenz, für die wir eine Zerlegung finden wollen. Wir setzen voraus, dass S global zugänglich abgespeichert ist, sodass auch die rekursiven Aufrufe des Algorithmus auf sie zugreifen können. Wenn wir im Folgenden von einem Aufruf von *Zerlege($\tilde{S}$)* schreiben, so meinen wir damit den initialen oder einen rekursiven Aufruf, wobei die Stufenpolygonsequenz $\tilde{S}$ das übergebene Argument ist. Im Falle des initialen Aufrufs gilt also $\tilde{S} = S$.

Bei den Aufrufen des Algorithmus *Zerlege($\tilde{S}$)* ist es nicht notwendig, die Stufenpolygonsequenz $\tilde{S}$ explizit zu berechnen und zu übergeben. Da alle Punkte in $\tilde{S}$ außer dem Eckpunkt auch zu der ursprünglichen Stufenpolygonsequenz S gehören und S global zugänglich ist, genügt es, den Eckpunkt und die Indizes des obersten und des am weitesten rechts liegenden Punktes von $\tilde{S}$ zu übergeben. Die restlichen Punkte lassen sich dann bei Bedarf bestimmen. Ein Aufruf des Algorithmus ist folglich in konstanter Zeit möglich.

Es sei nun $\tilde{S} := (p_c, p_l, \ldots, p_r)$ $(1 \leq l \leq r \leq n)$ die Eingabe eines Aufrufs des Algorithmus, die – wie oben beschrieben – in der Form (p_c, l, r) übergeben wird. Die Werte $a_j = p_j^x - p_c^x$ und $b_j = p_{j+1}^y - p_x^y$ $(l \leq j \leq r - 1)$ sind offensichtlich in konstanter Zeit berechenbar. Sie werden aber nur bestimmt, wenn sie tatsächlich benötigt werden. Da $a_l < \ldots < a_{r-1}$ sowie $b_l > \ldots > b_{r-1}$ gilt, gilt auch $a_l - b_l < \ldots < a_{r-1} - b_{r-1}$. Folglich ist M genau dann die leere Menge, wenn $a_l - b_l$ positiv ist. Weiter kann man – falls M nicht die leere Menge ist – das Maximum m von M finden, ohne M explizit berechnen zu müssen. Dazu bestimmt man mit binärer Suche den größten Index $j \in \{l, \ldots, r - 1\}$, für den $a_j - b_j \leq 0$ gilt. Die binäre Suche hat bekanntlich einen Zeitaufwand, der in $O(\log n)$ liegt. Dabei sei noch einmal betont, dass nur die Elemente der Sequenz $a_j - b_j$ berechnet werden, die bei der binären Suche betrachtet werden.

Die Teilmengen $\mathcal{Z}^{\mathrm{c}}$, $\mathcal{Z}^{\mathrm{o}}$ und $\mathcal{Z}^{\mathrm{r}}$ im Pseudocode von *Zerlege* wurden nur verwendet, um die Korrektheit und die Faktor-2-Approximation leichter beweisen zu können. Bei der Implementierung kann man selbstverständlich alle Strecken, die im Verlauf des Algorithmus entstehen, in eine global zugängliche Menge einfügen. Diese kann z. B. als Vektor oder als Liste implementiert werden, sodass das Einfügen in konstanter Zeit möglich ist. Dabei beachte man, dass die Zerlegung, welche der Algorithmus liefert, höchstens so viele Strecken hat, wie es Punkte im Stufenpolygon gibt. Die Größe der Menge und damit der Speicherplatzbedarf des Algorithmus liegen also in $O(n)$.

Ein einzelner Durchgang des Algorithmus hat – ohne die rekursiven Aufrufe – eine Laufzeit, die in $O(\log n)$ liegt, denn die Laufzeit der binären Suche liegt in $O(\log n)$

und alles andere ist in konstanter Zeit berechenbar. Die Anzahl der Aufrufe des Algorithmus liegt in $O(n)$, denn wenn ein Punkt p_k $(1 \leq k \leq n-1)$ in einem Durchgang ausgewählt wird, so wird er in den rekursiven Aufrufen nicht mehr betrachtet. Insgesamt folgt, dass die angegebene Implementierung eine Laufzeit besitzt, die in $O(n \log n)$ liegt.

4.2.5 Eine $O(n)$-Implementierung

Für unsere Zwecke reicht eine Implementierung aus, deren Laufzeit in $O(n \log n)$ liegt. Der oben erwähnte Algorithmus *ApproxMMN*, in den unser Algorithmus eingebettet ist, besitzt nämlich sowieso – d. h. unabhängig von unserem Algorithmus – eine Laufzeit, die in $O(n \log n)$ liegt. Wir geben trotzdem eine weitere Implementierung an, deren Laufzeit in $O(n)$ liegt. Sie unterscheidet sich nur an einer Stelle – der Suche nach dem Maximum von M – von der $O(n \log n)$-Implementierung, erfordert aber eine sorgfältigere Abschätzung. Die grundsätzliche Idee der $O(n)$-Implementierung stammt aus [LÖ96], allerdings ist unser Beweis leichter verständlich.

Eine erste Beobachtung ist, dass man ein Element in einer (geordneten) Sequenz der Größe k per *exponentieller Suche* auch mit einem Zeitaufwand finden kann, der in $O(\log j)$ liegt. Dabei ist $j \in \{1, \ldots, k\}$ die Position des gesuchten Elementes. Dazu startet man am Anfang der Sequenz und geht so lange in 2-er-Potenz-Schritten nach oben, bis man das gesuchte Element gefunden oder übersprungen hat. Danach wendet man eine binäre Suche an, die allerdings nur in dem Bereich sucht, der im letzten Schritt übersprungen wurde (siehe z. B. [Man89, Kapitel 6.2]). Man beachte, dass j natürlich a priori unbekannt ist. Analog zu der eben beschriebenen Methode kann man auch am oberen Ende der Sequenz beginnen und sich nach unten vorarbeiten. Die Laufzeit dieser Variante der exponentiellen Suche liegt dann in $O(\log(k-j+1))$.

Der eigentliche „Trick“ besteht nun darin, die beiden gerade beschriebenen Varianten der exponentiellen Suche parallel auszuführen, bis eine von ihnen erfolgreich ist. Wir bezeichnen diese kombinierte Suche als *Parallelsuche*. Ihre Laufzeit liegt offensichtlich in $O(\log(\min\{j, k-j+1\}))$. Unsere $O(n)$-Implementierung erhalten wir dadurch, dass wir die normale Binärsuche in der $O(n \log n)$-Implementierung durch die Parallelsuche ersetzen. Ansonsten werden keine weiteren Änderungen vorgenommen.

Nun zeigen wir, dass die Laufzeit unserer $O(n)$-Implementierung tatsächlich in $O(n)$ liegt. Dazu führen wir als erstes eine Funktion T ein, die rekursiv definiert ist.

Definition 4.10 *Die Funktion $T : \mathbb{N} \longrightarrow \mathbb{R}$ sei folgendermaßen definiert:*

$$T(0) := 0, \quad T(1) := T(2) := 1,$$

$$T(n) := \max_{1 \leq j \leq n-1} \Big\{ T(j-1) + T(n-j-1) + \log_2\Big(\min\{j, n-j\} + 1\Big) \Big\}$$

$$(n \geq 2).$$

Die Laufzeit unserer Implementierung wird durch die Funktion T beschränkt. Dabei hängt die Eingabe von T nur von der Länge des Stufenpolygons ab, welches die Eingabe des Algorithmus ist.

Satz 4.11 *Es bezeichne* NoOfSteps(S) *die Anzahl der Rechenschritte, die die $O(n)$-Implementierung bei der Eingabe S in unserem Rechnermodell benötigt. Dann gilt:*

$$\forall \textit{Stufenpolygonsequenzen } S \textit{ mit } n := |S| - 1 > 0 : \textit{NoOfSteps}(S) \leq T(n).$$

Beweis: Es sei S eine Stufenpolygonsequenz mit $n := |S| - 1 > 0$. Wir führen den Beweis per vollständiger Induktion über n.

$n \leq 2$: In diesem Fall ist die Laufzeit offensichtlich konstant. Wir dürfen also o.B.d.A. annehmen, dass der Algorithmus in unserem Rechnermodell nur einen Rechenschritt benötigt.

$n > 2$: Es sei $j \in \{1, \ldots, n-1\}$ die Position des – per Parallelsuche gefundenen – Elementes, an dem S (im ersten Durchgang des Algorithmus) „aufgeteilt" wird. Nach den obigen Bemerkungen hat der erste Durchgang des Algorithmus ohne die rekursiven Aufrufe eine Laufzeit, die in $O(\log(\min\{j, (n-1) - j + 1\}))$ liegt. Wir dürfen also o.B.d.A. annehmen, dass der Algorithmus in unserem Rechnermodell $\log_2(\min\{j, n-j\} + 1)$ Rechenschritte benötigt, wenn man die rekursiven Aufrufe nicht mitzählt. Das hintere „+1" wurde dabei hinzugefügt, damit die Anzahl der Rechenschritte immer positiv ist. Die beiden rekursiven Aufrufe im Algorithmus brauchen nach der Induktionsvoraussetzung höchstens $T(j-1)$ bzw. $T(n-j-1)$ Rechenschritte. Es gilt genau dann $j - 1 = 0$, wenn der entsprechende rekursive Aufruf nicht ausgeführt wird (analog für $n - j - 1$). Aus diesem Grund haben wir $T(0) := 0$ gesetzt. Addiert man alle Rechenschritte zusammen, so folgt die Behauptung. ■

Für die Funktion T können wir zeigen, dass sie in $O(n)$ liegt. Dies liefert uns die gewünschte Abschätzung der Laufzeit unserer Implementierung.

Satz 4.12 *Es gilt $T(n) \leq \Big(2n - \log_2(n+2) + 1\Big)$. Damit liegt die Laufzeit der $O(n)$-Implementierung in $O(n)$, wobei $n+1$ die Länge der Eingabe ist.*

Beweis: Wir führen den Beweis per vollständiger Induktion über n.

$$n = 0 : \; T(0) \; = \; 0 \; \leq \; \Big(0 - \log_2(0+2) + 1\Big) \; = \; 0$$

$$n = 1 : \; T(1) \; = \; 1 \; \leq \; \Big(2 - \log_2(1+2) + 1\Big) \; = \; \Big(3 - \log_2(3)\Big)$$

$$n = 2 : \; T(2) \; = \; 1 \; \leq \; \Big(4 - \log_2(2+2) + 1\Big) \; = \; 3$$

$n > 2$: Wir definieren für $j = 1, \ldots, n-1$:

$$U_n(j) \; := \; T(j-1) + T(n-j-1) + \log_2\Big(\min\{j, n-j\} + 1\Big)$$

Es gilt $T(n) = \max_{1 \leq j \leq n-1}\{U_n(j)\}$. Man rechnet leicht nach, dass $U_n(j) = U_n(n-j)$ für alle $j \in (1 \leq j \leq n-1)$ gilt. Folglich reicht es zu zeigen, dass $U_n(j) \leq (2n - \log_2(n+2) + 1$ für alle $j \in \{1, \ldots, \lceil (n-1)/2 \rceil\}$ gilt. Dabei sei $\lceil x \rceil := \min\{m \in \mathbb{N} \mid m \geq x\}$ die Gaußsche Funktion.

Es sei also $j \in \{1, \ldots, \lceil (n-1)/2 \rceil\}$. Dann gilt:

$$\begin{aligned}
U_n(j) &= T(j-1) + T(n-j-1) + \log_2\Big(\min\{j, n-j\} + 1\Big) \\
&\leq T(j-1) + T(n-j-1) + \log_2(j+1) \\
&\overset{\text{I.V.}}{\leq} \Big(2(j-1) - \log_2((j-1)+2) + 1\Big) \; + \\
&\quad \Big(2(n-j-1) - \log_2((n-j-1)+2) + 1\Big) \; + \\
&\quad \log_2(j+1) \\
&= \Big(2n - 2 - \log_2(n-j+1)\Big) \\
&\overset{(*)}{\leq} \left(2n - \log_2(4) - \log_2\left(\frac{n+1}{2}\right)\right) \\
&= \Big(2n - \log_2(2n+2)\Big) \\
&\leq \Big(2n - \log_2(n+2)\Big) \\
&\leq \Big(2n - \log_2(n+2) + 1\Big)
\end{aligned}$$

Dabei folgt $(*)$ aus der leicht einsehbaren Ungleichung

$$n - j + 1 \;\geq\; n - \left\lceil \frac{n-1}{2} \right\rceil + 1 \;\geq\; \left\{ \begin{array}{ll} \frac{n+2}{2} & \text{falls } n \text{ gerade} \\ \frac{n+3}{2} & \text{falls } n \text{ ungerade} \end{array} \right\} \;\geq\; \frac{n+1}{2}$$

■

5. Approximative Manhattan-Netzwerke für einen Spezialfall

In diesem Kapitel beschäftigen wir uns etwas ausführlicher mit dem Algorithmus *ApproxMMN*. Wie bereits erwähnt, approximiert er minimale Manhattan-Netzwerke, wobei er einen Approximationsfaktor von 3 besitzt. Seine Laufzeit liegt in $O(n \log n)$. Wir zeigen, dass der Algorithmus für Mengen, deren Punkte alle auf dem Rand ihrer *Pareto-Hüllen* liegen, sogar ein Manhattan-Netzwerk liefert, welches höchstens zweimal die Länge einer minimalen Lösung hat. Mengen mit dieser Eigenschaft nennen wir *Hüllenmengen*. Genau genommen muss der Algorithmus dazu leicht modifiziert werden. Allerdings sind diese Änderungen von geringer Natur, seine wesentlichen Eigenschaften – insbesondere die Laufzeit und der „normale“ Approximationsfaktor – bleiben dabei erhalten.

Zum Verständnis dieses Kapitels ist eine sehr gute Vertrautheit mit dem Artikel [BWW04a] unbedingt notwendig. Wir werden einzelne Stellen aus dem Beweis der Faktor-3-Approximation gezielt verändern und dabei von den zusätzlichen Eigenschaften der Hüllenmengen Gebrauch machen. Aus diesem Grunde benutzen wir die Bezeichnungen und Definitionen aus [BWW04a], ohne sie erneut formell zu definieren.

Im ersten Teil erklären wir, was orthokonvexe Mengen sind und wie die Pareto-Hülle einer endlichen Menge von Punkten definiert ist. Der zweite Teil beschreibt die Änderungen, die wir an dem Algorithmus *ApproxMMN* vornehmen. Wir werden zeigen, dass die Modifikationen die wesentlichen Eigenschaften des Algorithmus nicht verändern. Im letzten Teil führen wir schließlich den Beweis, dass der Approximationsfaktor des Algorithmus auf Hüllenmengen 2 ist.

5.1 Hüllenmengen

Wir beginnen mit der Definition von *orthokonvex*. Eine Menge M von Punkten ist orthokonvex, falls der Schnitt von M mit einer beliebigen horizontalen oder vertikalen Geraden leer oder zusammenhängend ist. Außerdem muss es zwischen je zwei Punkten aus M einen Manhattan-Weg geben, der vollständig in M liegt. Etwas formeller:

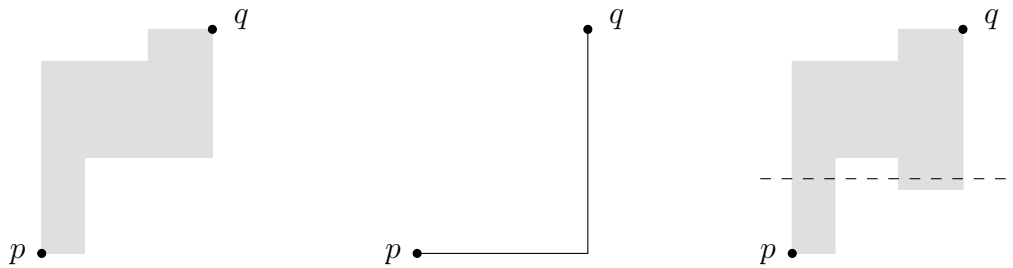
Definition 5.1 *Eine Menge $M \subseteq \mathbb{R}^2$ ist* orthokonvex*, wenn gilt:*

(i) $\forall y, x_1, x_2 \in \mathbb{R} : \Big((x_1, y) \in M \wedge (x_2, y) \in M\Big) \implies \Big(\forall z \in [x_1, x_2] : (z, y) \in M\Big)$,

(ii) $\forall x, y_1, y_2 \in \mathbb{R} : \Big((x, y_1) \in M \wedge (x, y_2) \in M\Big) \implies \Big(\forall z \in [y_1, y_2] : (x, z) \in M\Big)$,

(iii) $\forall p, q \in M, p \neq q : \exists MW\ W_{p,q} : \forall s \in W_{p,q} : s \subseteq M$.

Abbildung 5.1 zeigt zwei Beispiele und ein Gegenbeispiel für orthokonvexe Mengen. Der Schnitt zweier orthokonvexer Mengen ist im Allgemeinen nicht orthokonvex. Als Beispiel betrachte man den Schnitt der Mengen aus Abbildung 5.1 (a) und (b), der nur aus den beiden Punkten p und q besteht (die Menge in Abbildung 5.1 (a) sei offen). Folglich ist Bedingung (iii) aus Definition 5.1 für den Schnitt nicht erfüllt. Fordert man für orthokonvexe Mengen nur die ersten beiden Bedingungen aus Definition 5.1, so sind orthokonvexe Mengen unter dem Schnitt abgeschlossen. Allerdings wäre diese relaxierte Definition für uns ungeeignet.



(a) ok (b) ok (c) nicht ok

Abbildung 5.1: Beispiele für (nicht) orthokonvexe (ok) Mengen.

Da orthokonvexe Mengen unter dem Schnitt nicht abgeschlossen sind, lässt sich die orthokonvexe Hülle einer Menge M nicht „wie üblich“ als Schnitt aller orthokonvexen Mengen, die M enthalten, definieren. Es gibt aber eine bezüglich M ausgezeichnete orthokonvexe Menge, die *Pareto-Hülle* von M. Zur Erinnerung: $d(p, q)$ bezeichnet den Abstand zweier Punkte p und q bezüglich der l_1-Norm.

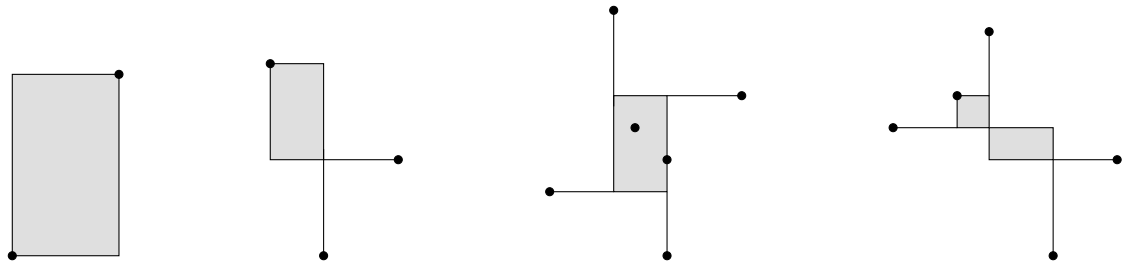
Definition 5.2 (Pareto-Hülle) *Es sei M eine Menge von Punkten. Ein Punkt $p \in \mathbb{R}^2$ ist* effizient *bezüglich M, wenn gilt:*

$$\forall q \in \mathbb{R}^2 : \Big(\exists r \in M : d(p, r) < d(q, r) \quad \vee \quad \forall r \in M : d(p, r) \leq d(q, r)\Big).$$

Die Pareto-Hülle $\mathcal{P}_M$ *von M ist die Menge aller bezüglich M effizienten Punkte.*

Ein Punkt p ist effizient bezüglich M, wenn es *keinen* Punkt q gibt, der die beiden folgenden Eigenschaften hat: Jeder Punkt in M ist höchstens so weit von q entfernt, wie er von p entfernt ist und es gibt einen Punkt in M, der echt näher an q als an p liegt.

Die Bestimmung der Pareto-Hülle per Hand ist nicht ganz einfach. Es empfiehlt sich daher, die Beispiele in Abbildung 5.2 einmal in aller Ruhe zu verifizieren. Aus [CNV04] und den dort angegebenen Referenzen ist bekannt, dass die Pareto-Hülle immer orthokonvex ist.



(a) (b) (c) (d)

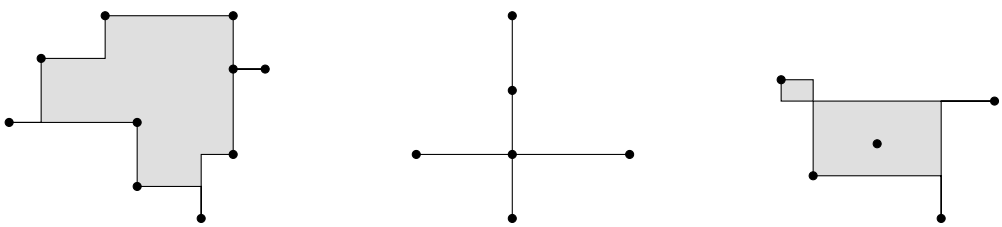
Abbildung 5.2: Beispiele für Pareto-Hüllen.

Wie in der Einleitung erwähnt, interessieren wir uns für endliche Mengen M, deren Punkte alle auf dem Rand der Pareto-Hülle von M liegen. Die Bedingung ist in dieser Formulierung allerdings wenig handlich. Deshalb fordern wir bei der Definition der *Hüllenmengen* eine auf den ersten Blick völlig andere Eigenschaft, die für die späteren Beweise viel praktischer ist. Es stellt sich jedoch heraus, dass beide Bedingungen äquivalent sind. Zur Erinnerung: $Q(p, i)$ für $i \in \{1, 2, 3, 4\}$ ist – wie in [BWW04a] definiert – der i-te abgeschlossene Quadrant mit dem Ursprung p.

Definition 5.3 (Hüllenmenge) *Eine endliche Menge $M \subseteq \mathbb{R}^2$ ist eine* Hüllenmenge*, wenn gilt:*

$$\forall p \in M : \exists i \in \{1, 2, 3, 4\} : I(Q(p, i)) \cap M = \emptyset.$$

Eine Hüllenmenge M darf also keinen Punkt p besitzen, der von allen vier Seiten von Punkten aus M, die nicht auf gleicher Höhe oder Breite wie p liegen, umgeben ist. In Abbildung 5.3 sind zwei Beispiele und ein Gegenbeispiel für Hüllenmengen angegeben.



(a) HM (b) HM (c) keine HM

Abbildung 5.3: (Gegen-)Beispiele für Hüllenmengen (HM).

Der nächste Satz zeigt wie versprochen, dass die Hüllenmengen genau diejenigen Mengen sind, deren Punkte alle auf dem Rand ihrer Pareto-Hülle liegen.

Satz 5.4 *Es sei M eine endliche Menge von Punkten. Dann sind die folgenden Aussagen äquivalent:*

(i) Die Menge M liegt auf dem Rand ihrer Pareto-Hülle, d. h. es gilt $M \subseteq \partial\mathcal{P}_M$.

(ii) $\exists$ orthokonvexes $K \subseteq \mathbb{R}^2 : M \subseteq \partial K$.

(iii) M ist eine Hüllenmenge.

Beweis: (siehe auch Abbildung 5.4)

„$(i) \Longrightarrow (ii)$“: Die Menge $\mathcal{P}_M$ ist orthokonvex (siehe [CNV04]).

„$(ii) \Longrightarrow (iii)$“: Es sei $K \subseteq \mathbb{R}^2$ eine orthokonvexe Menge, auf deren Rand alle Punkte aus M liegen. Wir nehmen an, dass M keine Hüllenmenge ist, und führen dies zum Widerspruch. Dann gilt:

$$\exists p, q_1, \ldots, q_4 \in M : \forall i \in \{1, \ldots, 4\} : q_i \in \Big(\mathrm{I}(Q(p, i)) \cap M\Big)$$

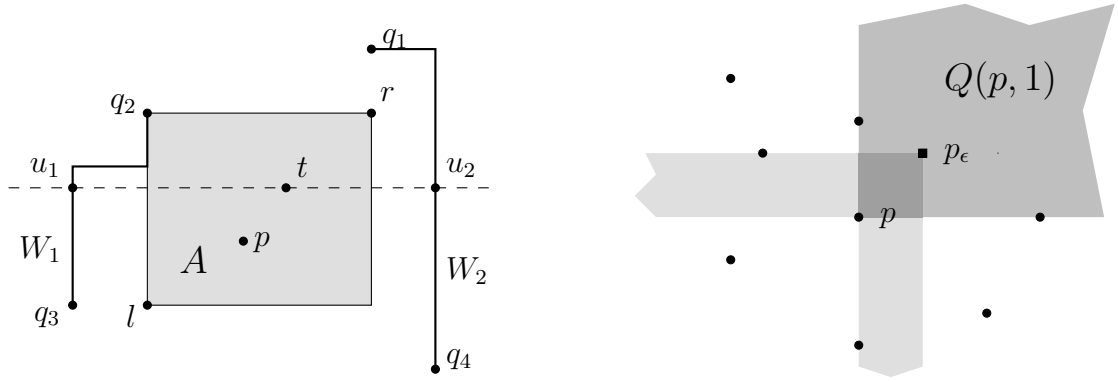
Wir setzen $l := (\max(q_2^x, q_3^x), \max(q_3^y, q_4^y))$ und $r := (\min(q_1^x, q_4^x), \min(q_2^y, q_1^y))$. Weiter sei $A := \{t \in \mathbb{R}^2 \mid l \leq t \leq r\}$. Es gilt $p \in \mathrm{I}(A)$. Wir zeigen, dass A eine Teilmenge von K ist. Folglich kann p nicht auf dem Rand von K liegen, was ein Widerspruch zur Voraussetzung ist. Dazu sei t ein Punkt in A. Da K orthokonvex ist, gibt es einen Manhattan-Weg W_1 zwischen q_2 und q_3 und einen Manhattan-Weg W_2 zwischen q_1 und q_4, die beide vollständig in K liegt. Da $q_3^y \leq t^y \leq q_2^y$ gilt, existiert ein $u_1 \in W_1$ mit $u_1^y = t^y$. Analog existiert ein $u_2 \in W_2$ mit $u_2^y = t^y$. Weiter gilt $u_1^x \leq l^x$, $r^y \leq u_2^x$ und damit $u_1^x \leq t^x \leq u_2^x$. Folglich ist auch t in K enthalten, denn K ist orthokonvex.

„$(iii) \Longrightarrow (i)$“: Es sei p ein Punkt aus M. Da M nach Voraussetzung eine Hüllenmenge ist, gibt es ein $t \in \{1, \ldots, 4\}$, sodass $\mathrm{I}(Q(p, t)) \cap M = \emptyset$ gilt. Es gelte o.B.d.A. $t = 1$. Wir zeigen, dass $p_\epsilon := (p^x + \epsilon, p^y + \epsilon)$ für genügend kleine $\epsilon > 0$ nicht effizient bezüglich M ist. Daraus folgt, dass es in jeder Umgebung von p einen Punkt gibt, der nicht in $\mathcal{P}_M$ liegt. Da p in $\mathcal{P}_M$ enthalten ist, enthält jede Umgebung von p trivialerweise einen Punkt aus $\mathcal{P}_M$. Folglich liegt p auf dem Rand von $\mathcal{P}_M$. Es sei $\epsilon > 0$ so klein gewählt, dass es keinen Punkt $q \in M$ mit $p^x < q^x \leq p_\epsilon^x$ oder $p^y < q^y \leq p_\epsilon^y$ gibt (M ist endlich). Um zu zeigen, dass p_ϵ nicht effizient bezüglich M ist, weisen wir nach, dass unser Punkt p keine der beiden – durch „oder“ verknüpften – Bedingungen in Definition 5.2 erfüllt. Dies ist äquivalent dazu, dass p die beiden negierten Bedingungen erfüllt. Wir zeigen also Folgendes:

(1) $\forall r \in M : d(p_\epsilon, r) \geq d(p, r)$,

(2) $\exists r \in M : d(p_\epsilon, r) > d(p, r)$.

Es gilt $d(p_\epsilon, p) = 2\epsilon > d(p, p) = 0$. Daraus folgt – mit $p = r$ – die zweite Bedingung. Für den Nachweis von (1) sei r ein Punkt aus M. Wir unterscheiden drei Fälle. Falls $r \leq p_\epsilon$ gilt, so gilt nach Wahl von ϵ auch $r \leq p$. Daraus folgt sofort $d(p_\epsilon, r) = d(p, r) + 2\epsilon \geq d(p, r)$. Falls $r^y > p_\epsilon^y (> p^y)$ gilt, so gilt auch $r^x \leq p^x (< p_\epsilon^x)$, denn



(a) Zur Indikation „$(ii) \Longrightarrow (iii)$“.

(b) Zur Indikation „$(iii) \Longrightarrow (i)$“.

Abbildung 5.4: Zum Beweis von Satz 5.4.

$\mathrm{I}(Q(p,1)) \cap M$ ist nach Voraussetzung die leere Menge. Damit gilt $d(p_\epsilon, r) = (r^y - p_\epsilon^y) + (p_\epsilon^x - r^x) = (r^y - (p^y + \epsilon)) + (p^x + \epsilon - r^x) = (r^y - p^y) + (p^x - r^x) = d(p,r)$. Falls letztlich $r^x > p_\epsilon^x$ gilt, so gilt auch $r^y \leq p^y$ und der Beweis verläuft analog zum zweiten Fall. ■

Im weiteren Verlauf des Kapitels werden wir mehrmals einige Schlussfolgerungen verwenden, die vielleicht nicht sofort einsichtig sind. Deshalb formulieren wir sie als Lemmata. Das erste Lemma betrachtet zwei Punkten p und q in P, die nicht auf einer Höhe liegen, und besagt: Alle horizontalen Geraden, die – bezüglich der y-Achse – zwischen p und q liegen, schneiden ein Rechteck aus $\mathcal{R}_{\text{ver}}$, welches – bezüglich der x-Achse – zwischen p und q liegt.

Lemma 5.5 *Es seien p und q zwei Punkte aus der Menge P mit $p^y \neq q^y$. Weiter bezeichne g_y die horizontale Gerade, die den Punkt y enthält. Wir definieren $x_\mathrm{l} := \min\{p^x, q^x\}$, $x_\mathrm{r} := \max\{p^x, q^x\}$, $y_\mathrm{l} := \min\{p^y, q^y\}$ und $y_\mathrm{r} := \max\{p^y, q^y\}$. Es gilt:*

$$\forall y \in [y_\mathrm{l}, y_\mathrm{r}] : \exists R \in \mathcal{R}_{\text{ver}} : \emptyset \neq g_y \cap R \subseteq [x_\mathrm{l}, x_\mathrm{r}] \times \{y\}.$$

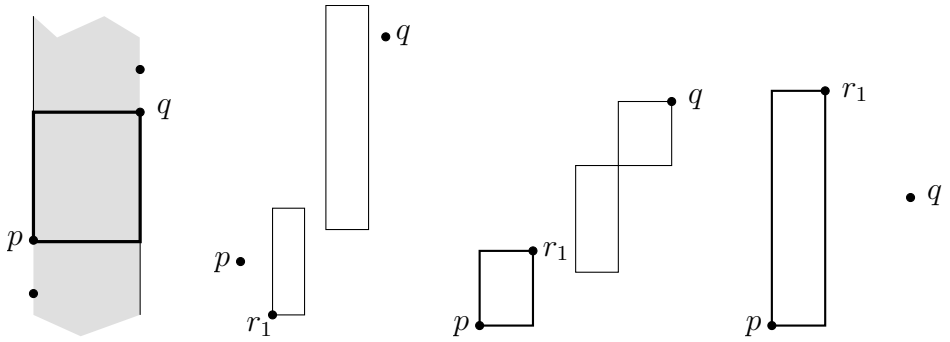
Beweis: Es gelte o.B.d.A. $p^x \leq q^x$. Wir nehmen außerdem $p^y < q^y$ an, der Fall $p^y > q^y$ verläuft analog. Zuerst führen wir eine lineare und strikte Ordnung ein:

$$r \prec s \quad :\Longleftrightarrow \quad r^x < s^x \ \vee \ \Big(r^x = s^x \ \wedge \ r^y < s^y\Big) \qquad (r, s \in \mathbb{R}^2).$$

„$\prec$“ ordnet die Punkte in $\mathbb{R}^2$ nach ihren x-Koordinaten und bei Gleichheit nach ihren y-Koordinaten, insbesondere gilt $p \prec q$. Es sei $(r_1, \ldots, r_m)$ mit $r_i \prec r_{i+1}$ $(i = 1, \ldots, m-1)$ die geordnete Sequenz aller Punkte aus P, die (bezüglich „$\prec$“) zwischen p und q liegen, d. h. es gilt $\{r \in P \mid p \prec r \prec q\} = \{r_1, \ldots, r_m\}$. Wir führen den Beweis per vollständiger Induktion über die Länge m der Sequenz (siehe auch Abbildung 5.5).

$m = 0$: Nach den Voraussetzungen gilt $p^y < q^y$. Außerdem gibt es keinen Punkt in P, der (bezüglich „$\prec$“) zwischen p und q liegt. Man prüft leicht nach, dass in diesem Fall (p, q) in Z_{ver} liegen muss. Also liegt $\mathrm{BBox}(p, q)$ in $\mathcal{R}_{\text{ver}}$, was unsere Behauptung beweist.

$m > 0$: Nach der Induktionsvoraussetzung gilt die Behauptung für die beiden Punkte r_1 und q. Wir machen eine Fallunterscheidung bezüglich der Höhe von r_1. Falls $r_1^y \leq p^y$ gilt, so muss nach der Definition von „$\prec$“ auch $r_1^x > p^x$ gelten. Damit gilt $[p^y, q^y] \subseteq [r_1^y, q^y]$ und $[p^x, q^x] \supseteq [r_1^x, q^x]$. Die Behauptung folgt also direkt aus der Induktionsvoraussetzung. Falls $p^y < r_1^y < q^y$ gilt, so gilt auch $p^x \leq r_1^x \leq q^x$. Die Behauptung folgt dann für $y \in [r_1^y, q^y]$ aus der Induktionsvoraussetzung und für $y \in [p^y, r_1^y]$ wie im Fall $m = 0$. Falls $r_1^y \geq q^y$ gilt, so gilt auch $p^x \leq r_1^x \leq q^x$ und die Behauptung folgt wie im Fall $m = 0$. ■



(a) $m = 0$ (b) $r_1^y \leq p^y$ (c) $p^y < r_1^y < q^y$ (d) $r_1^y \geq q^y$

Abbildung 5.5: Zum Beweis von Lemma 5.5.

Zu Lemma 5.5 gibt es natürlich eine „duale“ Schlussfolgerung, die analoge Aussagen über vertikale Geraden und Rechtecke aus $\mathcal{R}_{\text{hor}}$ macht. Eine unmittelbare Folgerung aus Lemma 5.5 ist, dass (in der Situation von Lemma 5.5) alle horizontalen Geraden, die – bezüglich der y-Achse – zwischen p und q liegen, ein Segment aus $\mathcal{C}_{\text{ver}}$ schneiden, welches – bezüglich der x-Achse – zwischen p und q liegt.

Das nächste Lemma charakterisiert die in [BWW04a] definierten Zusammenhangskomponenten. Zuvor erinnern wir jedoch noch kurz an einige Begriffe aus [BWW04a] (siehe auch Abbildung 5.7). Dazu sei A eine Zusammenhangskomponente aus $\delta(q, t)$ für $q \in P$ und $t \in \{1, 2, 3, 4\}$. Wir beschränken uns auf den Fall $t = 1$. Weiter sei $A \cap P = \{p_1, \ldots, p_m\}$ mit $p_i^x < p_{i+1}^x$ $(i = 1, \ldots, m-1)$. Es gilt $v_A := p_i.\text{ynbor}[3]$, wobei v_A wohldefiniert ist. Der Punkt w_A ist definiert als der (eindeutige) horizontale Vorgänger von v_A. Das von v_A und w_A aufgespannte Rechteck $R := \text{BBox}(v_A, w_A)$ liegt in $\mathcal{R}_{\text{hor}}$ und begrenzt A in Richtung von q.

Lemma 5.6 *Es seien v und w zwei Punkte mit $v^x < w^x$ und $v^y < w^y$. Weiter sei A eine Zusammenhangskomponente aus $\delta(q, t)$ für $q \in P$ und $t \in \{1, 2, 3, 4\}$. Die Punkte v_A und w_A seien definiert wie in [BWW04a]. Es gelte $\{v, w\} = \{v_A, w_A\}$. Weiter sei $R := BBox(v, w)$. Dann gilt:*

(i) Entweder es gilt $t = 1$, $v_A = v$ und $w_A = w$ oder es gilt $t = 3$, $v_A = w$ und $w_A = v$.

(ii) Für $t = 1$ $(t = 3)$ liegt A oberhalb (unterhalb) von R.

Beweis: Nach der Definition von v_A und w_A gilt allgemein

- $t = 1 \implies v_A^x < w_A^x \wedge v_A^y \leq w_A^y$,
- $t = 2 \implies v_A^x > w_A^x \wedge v_A^y \leq w_A^y$,
- $t = 3 \implies v_A^x > w_A^x \wedge v_A^y \geq w_A^y$,
- $t = 4 \implies v_A^x < w_A^x \wedge v_A^y \geq w_A^y$.

Nach der Voraussetzung gilt entweder $v_A = v$ und $w_A = w$ oder aber $v_A = w$ und $w_A = v$. Macht man sich leicht klar, dass beide Möglichkeiten für $t = 2$ und $t = 4$ wegen $v^x < w^x$ und $v^y < w^y$ nicht möglich sind. Für $t = 1$ gilt offensichtlich $v_A = v$ und $w_A = w$ und für $t = 3$ gilt $v_A = w$ und $w_A = v$. Dies beweist (i). In beiden Fällen gilt $R = \mathrm{BBox}(v_A, w_A)$. Mit der Definition von v_A und w_A und den Aussagen aus (i) folgt damit (ii). ■

Wenn man in Lemma 5.6 „$v^y > w^y$“ anstelle von „$v^y < w^y$“ fordert und „t = 1“ durch „t = 2“ sowie „t = 3“ durch „t = 4“ ersetzt, so behält das Lemma seine Gültigkeit. Die Beweise verlaufen vollständig analog.

5.2 Änderungen am Algorithmus *ApproxMMN*

Wir müssen den Algorithmus an drei Stellen leicht modifizieren, damit sein Approximationsfaktor auf Hüllenmengen 2 ist. Dabei werden die wesentlichen Eigenschaften des Algorithmus – die Laufzeit und der Approximationsfaktor auf allgemeinen Instanzen – nicht verändert. Zur visuellen Erklärung der Modifikationen dienen die Abbildungen 5.6, 5.7 und 5.8.

Änderung 1: Es sei N_1 berechnet wie bisher. Wir betrachten jeweils zwei Punkte p und q aus P, für die $\{p, q\}$ sowohl in Z_{ver} als auch in Z_{hor} enthalten ist. In diesem Fall ist das Rechteck $R := \mathrm{BBox}(p, q)$ nicht degeneriert und liegt in $\mathcal{R}_{\mathrm{ver}}$ und in $\mathcal{R}_{\mathrm{hor}}$. Wir werden N_1 so modifizieren, dass unter gewissen Bedingungen – die im Folgenden beschrieben werden – keine Segmente aus $\mathcal{S}$ in R liegen. Diese Eigenschaft werden wir später bei den Abschätzungen der Länge unseres Manhattan-Netzwerkes ausnutzen.

Nach der Konstruktion von N_1 liegt jeweils einer der vertikalen bzw. horizontalen Ränder von R in $\mathcal{C}_{\mathrm{ver}}$ bzw. in $\mathcal{C}_{\mathrm{hor}}$. Wir wollen sie s_{v} und s_{h} nennen. Wir erklären das Vorgehen für den Fall $p^x < q^x$ und $p^y > q^y$, die anderen Fälle gehen analog. Es sei $p' := (p^x, q^y)$ und $q' := (q^x, p^y)$. Falls $\mathrm{I}(Q(p', 1)) \cap P = \emptyset$ gilt, so ersetzen wir die Segmente s_{v} und s_{h} durch $\mathrm{Seg}\,[p, p']$ bzw. $\mathrm{Seg}\,[p', q]$. Dabei ändern sich die Gesamtlängen von $\mathcal{C}_{\mathrm{ver}}$ und von $\mathcal{C}_{\mathrm{hor}}$ nicht. Außerdem entfernen wir die Segmente in $\mathcal{S}$, die für die Verbindung zwischen p und q zuständig waren, da sie offensichtlich nicht mehr gebraucht werden. Aus der Wahl von p und q und den Voraussetzungen folgt, dass es keinen Punkt $r \in \mathbb{R}^2$ mit $r^x > p^x$ und $r^y \geq p^y$ gibt. Dies garantiert, dass wir keine Segmente entfernen, die auch auf dem Rand anderer Rechtecke aus $\mathcal{R}_{\mathrm{ver}}$ bzw. $\mathcal{R}_{\mathrm{hor}}$ liegen. Folglich ist $\mathcal{C}_{\mathrm{hor}}$ weiterhin ein (schönes) MHC. Weiter kann es keinen Punkt $t \in \mathbb{R}^2$ mit $t^y > q^y$ und $t^x \geq q^x$ geben. Damit ist auch $\mathcal{C}_{\mathrm{ver}}$ weiterhin ein (schönes) MVC. Falls $\mathrm{I}(Q(q', 3)) \cap P = \emptyset$ gilt und der erste Fall nicht erfüllt ist, so verfahren wir analog. Ist keiner der beiden Bedingungen erfüllt, so werden keine Änderungen vorgenommen.

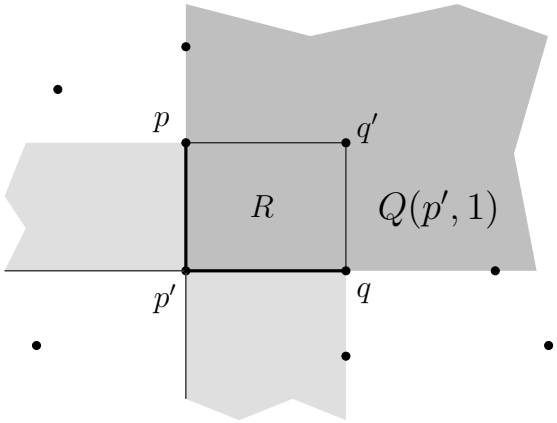


Abbildung 5.6: Illustration zur *Änderung 1* des Algorithmus.

Zur Implementierung: Die Menge N_1 wird wie bisher berechnet, danach werden die Änderungen vorgenommen. Die Anzahl der Paare von Punkten aus P, die in Z_{ver} und Z_{hor} enthalten sind, liegt in $O(n)$. Die Bedingungen für die einzelnen Änderungen sowie die Änderungen selbst können für jedes Paar in konstanter Zeit geprüft bzw. bewerkstelligt werden. So ist z. B. die Überprüfung, ob $\mathrm{I}(Q(q', 3))$ leer ist, mit den bereits berechneten Zeigern *xnbor* und *ynbor* für p bzw. q möglich. Demzufolge liegt die Laufzeit von *Änderung 1* in $O(n)$. Alternativ ist es auch möglich, die Änderungen direkt in die Konstruktion von N_1 einzubauen.

Die beiden folgenden Änderungen werden für jede Zusammenhangskomponente A aus $\delta(q, t)$ für $q \in P$ und $t \in \{1, 2, 3, 4\}$ vorgenommen. Wir erklären die Modifikationen für den Fall $t = 1$, die anderen Fälle verlaufen analog. Es sei $A \cap P = \{p_1, \ldots, p_m\}$ mit $p_i^x < p_{i+1}^x$ $(i = 1, \ldots, m-1)$. Wir definieren p_A als den (eindeutigen) vertikalen Nachfolger von q. (In [BWW04a] heißt dieser Punkt p_0. Wir wollen jedoch die Abhängigkeit von A betonen.)

Änderung 2: Die Punkte $w := w_A$, $v := v_A$ und q seien definiert wie in [BWW04a] (siehe auch Abbildung 5.7). Wegen $t = 1$ gilt definitionsgemäß $v^x < w^x$ und $v^y \leq w^y$. Es gelte sogar $v^y < w^y$, im Falle $v^y = w^y$ nehmen wir keine Änderungen vor. Es seien s_v und s_w die (Teil-)Segmente aus C^{hor}, die auf dem Rand von $R = \mathrm{BBox}(v, w)$ liegen und inzident zu v bzw. w sind. (R ist nach Wahl in $\mathcal{R}_{\text{hor}}$ enthalten). Weiter sei s das Segment aus $\mathcal{S}$, welches in R liegt und w und v miteinander verbindet. Falls s nicht existiert, so fügen wir es in $\mathcal{S}$ ein. Dabei bleiben die bisherigen Abschätzungen bezüglich $\mathcal{S}$ erhalten. Es gibt also ein $x \in \mathbb{R}$ mit $v^x \leq x \leq w^x$, sodass gilt:

$$s_v = \mathrm{Seg}\,[v, (x, v^y)]\,,\ s_w = \mathrm{Seg}\,[(x, w^y), w] \text{ und } s = \mathrm{Seg}\,[(x, v^y), (x, w^y)]\,.$$

Falls (i) $w.\mathrm{xnbor}\,[3] = q$, (ii) $\mathrm{I}(Q(w, 4)) \cap P = \emptyset$ und (iii) $x > p_A^x$ gilt, so ersetzen wir s_v durch das Segment $\mathrm{Seg}\,[v, c_A]$, s_w durch $\mathrm{Seg}\,[q_A, w]$ und s durch $\mathrm{Seg}\,[c_A, q_A]$. Dabei ändern sich die Gesamtlängen von $\mathcal{C}_{\text{hor}}$ und von $\mathcal{S}$ nicht. Aus (i) und (ii) folgt insbesondere, dass es keinen Punkt $r \in P$ mit $r^y \leq c_A^y$ und $r^x > c_A^x$ gibt. Folglich ist $\mathcal{C}_{\text{hor}}$ weiterhin ein (schönes) MHC. Da R nach Wahl nicht in $\mathcal{R}_{\text{ver}}$ enthalten sein kann, operieren *Änderung 1* und *2* auf unterschiedlichen Rechtecken, sie sind also unabhängig voneinander. Als Nächstes zeigen wir per Widerspruch, dass die einzelnen Modifikationen in *Änderung 2* sich nicht gegenseitig beeinflussen. Dafür nehmen wir an, dass die drei Bedingungen erfüllt sind und dass es eine von A verschiedene

Zusammenhangskomponente $\bar{A}$ aus einer der Mengen $\delta(\bar{q}, \bar{t})$ gibt, für welche die oben genannten drei Bedingungen ebenfalls erfüllt sind und $\{v_{\bar{A}}, w_{\bar{A}}\} = \{v, w\}$ gilt. Da nach Voraussetzung $v^x < w^x$ und $v^y < w^y$ gilt, folgt mit Lemma 5.6 (i), dass $\bar{t} = 1$ oder $\bar{t} = 3$ gelten muss. Bedingung (i) schließt $\bar{t} = 1$ aus, denn sonst müsste $q = \bar{q}$ gelten, was nicht möglich ist. Also muss $\bar{t} = 3$ gelten. Somit gilt $v_{\bar{A}} = w$ und $w_{\bar{A}} = v$. Wegen $w_{\bar{A}}.\text{xnbor}\,[1] = \bar{q}$ (der ersten Bedingung für $\bar{t} = 3$) muss $\bar{q}^x \leq p_A^x$ gelten. Damit kann aber Bedingung (iii) für $\bar{A}$ (für $\bar{t} = 3$ ist dies $x < p_{\bar{A}}^x$) nicht erfüllt sein, denn es gilt $p_{\bar{A}}^x \leq \bar{q}^x \leq p_A^x < x$.

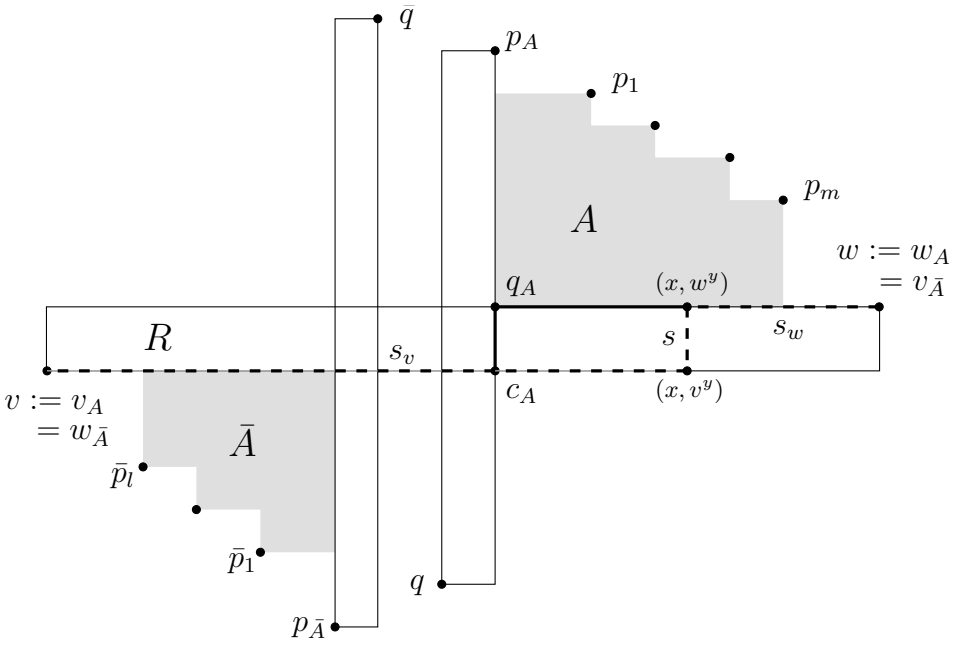


Abbildung 5.7: Illustration zur *Änderung 2* des Algorithmus.

Änderung 2 wird ausgeführt, nachdem die Zusammenhangskomponenten von $\mathcal{A}_3$ bestimmt wurden. Die Anzahl der Zusammenhangskomponenten liegt in $O(n)$, da die Anzahl der Paare aus Z_{quad} in $O(n)$ liegt. Die Bedingungen für die Änderungen sowie die Änderungen selbst können für jede Zusammenhangskomponente in konstanter Zeit geprüft bzw. bewerkstelligt werden. Demzufolge liegt die Laufzeit von *Änderung 2* in $O(n)$. Wie schon erwähnt, gibt es keine Wechselwirkungen mit *Änderung 1*.

Die ersten beiden Änderungen betreffen lediglich die Menge N_1, nachdem diese in *Phase I* des Algorithmus wie bisher berechnet wurde. Dabei bleiben die wesentlichen Eigenschaften von N_1 – wie bereits erläutert – erhalten. Folglich haben die Änderungen keinen negativen Einfluss auf den restlichen Algorithmus. Alle Aussagen und Beweise bleiben erhalten. Insbesondere sind die ersten beiden Änderungen vollständig unabhängig von der dritten Änderung. Wenn wir ab jetzt von der Menge N_1 sprechen, so meinen wir die von *Änderung 1* und *2* modifizierte Menge.

Änderung 3: Mit der Notation von *Änderung 2* definieren wir die drei Punkte $\tilde{q}_A := (p_1^x, q_A^y)$, $\tilde{c}_A := (p_1^x, c_A^y)$ und $\hat{q}_A := (p_1^x, p_m^y)$ sowie das Segment $\tilde{s}_A := \text{Seg}\,[\tilde{q}_A, \tilde{c}_A]$. Weiter definieren wir $p_i' := (p_i^x, p_{i+1}^y)$ $(i = 1, \ldots, m-1)$ und $\tilde{A} := A \setminus (\text{BBox}(q_A, p_1) \cup \text{BBox}(q_A, p_m))$. Man beachte, dass auch $\tilde{A}$ ein Stufenpolygon ist

und dass p_1 und p_m nicht mehr auf dem Rand von $\tilde{A}$ liegen. Falls m echt kleiner als 3 ist, so ist $\tilde{A}$ die leere Menge. Wir nehmen die folgenden Änderungen am Algorithmus vor:

(i) Anstelle von $\partial A \setminus \bigcup N_1$ fügen wir in *Phase II* die Mengen $\partial\tilde{A} \setminus \bigcup N_1$, $\mathrm{Seg}\,[\tilde{q}_A, \hat{q}_A]$ und – falls $m > 1$ gilt – $\big(\mathrm{Seg}\,[p_1, p_1'] \cup \mathrm{Seg}\,\big[p_m, p_{m-1}'\big]\big) \setminus \bigcup N_1$ zu N_2 hinzu.

(ii) Es sei x, s_{v_A}, s_{w_A} und s definiert wie in *Änderung 2*. Anstelle des Segments s_A fügen wir das Segment $\tilde{s}_A$ ein. Allerdings nur, falls $\tilde{q}_A^x < x$ gilt (d. h. falls $\tilde{s}_A$ echt links von s liegt) und falls $\tilde{s}_A$ nicht schon in N_1 enthalten ist.

(iii) In *Phase III* sorgen wir dafür, dass die Punkte auf dem Rand von $\tilde{A}$ mit $\hat{q}_A$ verbunden werden (anstatt die Punkte auf dem Rand von A mit q_A zu verbinden). D. h. wir wenden den Algorithmus *Zerlege* auf $\tilde{A}$ an.

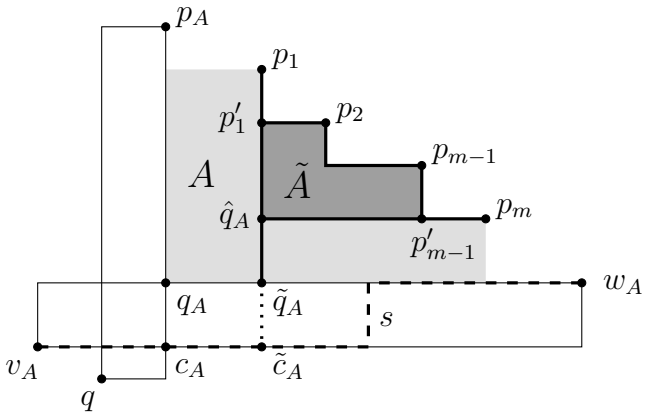


Abbildung 5.8: Illustration zur *Änderung 3* des Algorithmus.

Es ist leicht einzusehen, dass das Ergebnis des modifizierten Algorithmus immer noch ein Manhattan-Netzwerk bezüglich P ist. Auch die Laufzeit liegt offensichtlich weiterhin in $O(n \log n)$. Allerdings ist N_2 im Allgemeinen keine Teilmenge mehr von $\mathcal{A}_{12}$. Deshalb definieren wir $\tilde{\delta}(q,t)$ als die Vereinigung aller Mengen $\tilde{A}$, für welche die entsprechende Menge A eine Zusammenhangskomponente in $\delta(q,t)$ ist. Wegen der Definition von $\tilde{A}$ ist $\tilde{\delta}(q,t)$ eine Teilmenge von $\delta(q,t)$. Deshalb ist Lemma 7 aus [BWW04a] auch für $\tilde{\delta}(q,t)$ gültig. Weiter definieren wir $\tilde{\mathcal{A}}_3 := \bigcup_{t\in\{1,2,3,4\}} \bigcup_{p\in P} \tilde{\delta}(q,t)$ und $\tilde{\mathcal{A}}_{12} := \mathbb{R}^2 \setminus \tilde{\mathcal{A}}_3$. Es gilt $\tilde{\mathcal{A}}_3 \subseteq \mathcal{A}_3$ und $\tilde{\mathcal{A}}_{12} \supseteq \mathcal{A}_{12}$. Man prüft leicht nach, dass $N_1 \subseteq \tilde{\mathcal{A}}_{12}$, $N_2 \subseteq \tilde{\mathcal{A}}_{12}$ und $N_3 \subseteq \tilde{\mathcal{A}}_3$ gilt. Verwendet man nun für den Beweis des Approximationsfaktors aus [BWW04a] die Mengen $\tilde{\mathcal{A}}_{12}$, $\tilde{\mathcal{A}}_3$ und $\tilde{\delta}(q,t)$ anstelle von $\mathcal{A}_{12}$, $\mathcal{A}_3$ und $\delta(q,t)$, so lässt sich der Beweis nach wie vor führen. *Änderung 3* garantiert uns Folgendes: Falls eine horizontale Gerade zwei Segmente schneidet, die während der Bearbeitung von A in N_2 eingeführt wurden, so liegt diese Gerade zwischen p_2 und p_m.

Wir fassen zusammen: Der durch *Änderung 1*, *2* und *3* modifizierte Algorithmus hat die gleiche Abschätzung der Laufzeit und den gleichen Approximationsfaktor wie der ursprüngliche Algorithmus. Selbst die Beweise dieser Eigenschaften bleiben (im Wesentlichen) gleich.

5.3 Der Beweis der Faktor-2-Approximation

Wir kommen nun zu der zentralen Aussage dieses Kapitels. Sie besagt, dass der modifizierte Algorithmus *ApproxMMN* – im Folgenden als *der* Algorithmus bezeichnet – auf Hüllenmengen einen Approximationsfaktor von 2 besitzt. Da der Algorithmus für beliebige Mengen als Eingabe nur einen Approximationsfaktor von 3 garantiert, müssen wir im Beweis die Eigenschaften der Hüllenmengen ausnutzen. Es sei P also im Weiteren eine Hüllenmenge. Der Beweis des Approximationsfaktors aus [BWW04a] beweist im wesentlichen die folgenden Abschätzungen:

(i) $|N_3| \leq 2|N_{\text{opt}} \cap \mathcal{A}_3|$,

(ii) $|N_1| \leq |N_{\text{opt}} \cap \mathcal{A}_{12}| + |\mathcal{S}| = |N_{\text{opt}} \cap \mathcal{A}_{12}| + |\mathcal{S}^{\text{ver}}| + |\mathcal{S}^{\text{hor}}|$,

(iii) $|N_2^{\text{ver}}| \leq 2|\mathcal{C}_{\text{ver}}| - |\mathcal{S}^{\text{ver}}| \leq 2\,|(N_{\text{opt}} \cap \mathcal{A}_{12})^{\text{ver}}| - |\mathcal{S}^{\text{ver}}|$,

(iv) $|N_2^{\text{hor}}| \leq 2|\mathcal{C}_{\text{hor}}| - |\mathcal{S}^{\text{hor}}| \leq 2\left|(N_{\text{opt}} \cap \mathcal{A}_{12})^{\text{hor}}\right| - |\mathcal{S}^{\text{ver}}|$.

Daraus folgt $|N_1|+|N_2| \leq 3|N_{\text{opt}} \cap \mathcal{A}_{12}|$ und wegen der Disjunktheit von $\mathcal{A}_{12}$ und $\mathcal{A}_3$ die Behauptung $|N| = |N_1| + |N_2| + |N_3| \leq 3|N_{\text{opt}}|$. Ersetzt man in den Abschätzungen $\mathcal{A}_{12}$ und $\mathcal{A}_3$ durch $\tilde{\mathcal{A}}_{12}$ bzw. $\tilde{\mathcal{A}}_3$, so sind sie auch für den modifizierten Algorithmus gültig. Wir zeigen, dass die Abschätzungen (iii) und (iv) für Hüllenmengen verschärft werden können.

Lemma 5.7 (vgl. Lemma 11 aus [BWW04a]) *Falls die Eingabe des Algorithmus eine Hüllenmenge ist, so gilt* $|N_2^{\text{ver}}| \leq |\mathcal{C}_{\text{ver}}| - |\mathcal{S}^{\text{ver}}|$ *und* $|N_2^{\text{hor}}| \leq |\mathcal{C}_{\text{hor}}| - |\mathcal{S}^{\text{hor}}|$.

Zusammen mit den (modifizierten) Abschätzungen (i) und (ii), die für alle Eingaben gültig sind, folgt $|N_1| + |N_2| \leq 2|N_{\text{opt}} \cap \tilde{\mathcal{A}}_{12}|$ und somit unsere gewünschte Aussage $|N| \leq 2|N_{\text{opt}}|$. Wir fassen zusammen:

Satz 5.8 *Der (modifizierte) Algorithmus* ApproxMMN *besitzt auf Hüllenmengen einen Approximationsfaktor von 2.*

Bevor wir Lemma 5.7 beweisen, stellen wir noch vier weitere Lemmata auf, wobei sich zwei von ihnen stark an den Lemmata 9 bzw. 10 aus [BWW04a] orientieren. Als Erstes erinnern wir jedoch an einige Bezeichnungen aus [BWW04a], die wir allerdings übersetzt haben. Die *Zwischensegmente* (connecting segments) sind diejenigen Segmente aus N_2, die von der Art $\tilde{s}_A$ sind (siehe *Phase II* und *Änderung 3*). Alle Segmente in N_2, die keine Zwischensegmente sind, heißen *Randsegmente* (boundary segments). Die Segmente aus $\mathcal{C}_{\text{ver}} \cup \mathcal{C}_{\text{hor}}$ werden *$\mathcal{C}$-Segmente* (cover segments) genannt. Zusätzlich führen wir noch eine weitere Bezeichnung ein (siehe auch *Änderung 3*).

Definition 5.9 (Stufensegment) *Es sei A eine Zusammenhangskomponente aus einer der Mengen $\delta(q,t)$. Wir nehmen o.B.d.A. $t = 1$ an. Weiter sei $A \cap P = \{p_1, \ldots, p_m\}$ mit $p_i^x < p_{i+1}^x$ $(i = 1, \ldots, m-1)$. Ein Segment $s \in N_2$ ist ein* Stufensegment*, wenn es ein $k \in \{2, \ldots, m-1\}$ gibt, sodass s ein Teilsegment von $Seg\left[p'_{k-1}, p_k\right]$ oder $Seg[p_k, p'_k]$ ist.*

Die Stufensegmente sind also alle Segmente aus N_2, die auf dem „Stufenteil“ des Randes von $\tilde{A}$ liegen. Sie sind also insbesondere Randsegmente. Die Bedingung *Teil*segment kommt daher, dass nur die Teile des Rands von $\tilde{A}$ in N_2 eingefügt werden, die nicht bereits in N_1 enthalten sind. Das nächste Lemma macht einige Aussagen über Stufensegmente, die wir für die weiteren Beweise benötigen werden.

Lemma 5.10 *Es sei g eine horizontale Gerade mit $g \cap P = \emptyset$. Dann gilt:*

(i) Die Gerade g schneidet höchstens ein vertikales Stufensegment.

(ii) Falls g ein vertikales Stufensegment schneidet, so schneidet g kein Zwischensegment.

(iii) Falls g ein vertikales Stufensegment schneidet, so schneidet g kein vertikales Segment aus $\mathcal{S}$.

Beweis: Zum Verständnis der Argumente dient Abbildung 5.9. Es sei s ein vertikales Stufensegment, das g schneidet, und A die Zusammenhangskomponente aus $\delta(q, 1)$, bei deren Bearbeitung s in N_2 eingefügt wurde. Weiter sei $A \cap P = \{p_1, \ldots, p_m\}$ mit $p_i^x < p_{i+1}^x$. Es sei $k \in \{2, \ldots, m-1\}$ der Index, sodass s ein Teilsegment von $\mathrm{Seg}[p_k, p_k']$ ist. Wir definieren $\alpha := (-\infty, p_k^x] \times [p_{k+1}^y, p_k^y]$, $\beta := \mathrm{BBox}(p_k, p_{k+1})$ und $\gamma := [p_k^x, p_{k+1}^x] \times (-\infty, p_{k+1}^y]$. Nach der Definition von Z_{quad} kann es außer p_k und p_{k+1} keine Punkte in P geben, die in α, $\mathrm{I}(\beta)$ oder γ liegen. Da P eine Hüllenmenge ist, müssen ebenso die Inneren der Quadranten $Q(p_k, 1)$ und $Q(p_{k+1}, 1)$ leer sein (die drei anderen Quadranten scheiden wegen p_0, q und w_A aus). Es sei $r \in P$ der am weitesten rechts liegende Punkt auf dem oberen Rand von β und $t \in P$ der am höchsten liegende Punkt auf dem rechten Rand von β. Dabei ist $r = p_k$, $t = p_{k+1}$ oder auch $r = t$ möglich. Falls $r \neq t$ gilt, so ist $\{r, t\}$ sowohl in Z_{ver} als auch in Z_{hor} enthalten, da der horizontale und der vertikale Streifen zwischen r und t beide leer sind. Aus diesem Grund liegt auch *kein* Zwischensegment in $\mathrm{BBox}(r, t)$. Des Weiteren ist die Bedingung für r und t aus *Änderung 1* erfüllt. Wenn die Gerade g zwischen r und t verläuft, schneidet sie folglich kein Segment aus $\mathcal{S}$ und kein Zwischensegment. Dabei beachte man, dass g maximal ein Rechteck aus $\mathcal{R}_{\mathrm{hor}}$ schneidet (in diesem Fall $\mathrm{BBox}(r, t)$) und dass sowohl die Segmente aus $\mathcal{S}$ als auch die Zwischensegmente immer in einem Rechteck aus $\mathcal{R}_{\mathrm{hor}}$ liegen. Falls g zwischen t und p_{k+1} verläuft, so schneidet g nach Definition von Z_{hor} überhaupt kein Rechteck aus $\mathcal{R}_{\mathrm{hor}}$, da t und p_{k+1} direkt übereinander liegen und der horizontale Streifen zwischen t und p_{k+1} leer ist. Dies beweist (ii) und (iii).

Um (i) zu beweisen, nehmen wir an, dass g ein zweites Stufensegment $\bar{s}$ schneidet, welches bei der Bearbeitung einer Zusammenhangskomponente $\bar{A} \in \delta(\bar{q}, \bar{t})$ eingefügt wurde. Dies führen wir zum Widerspruch. Dazu seien $\bar{p}_j \in P$ und $\bar{p}_{j\pm1} \in P$ die Punkte auf dem Rand von $\bar{A}$, sodass $\bar{s}$ auf einem der vertikalen Ränder von $\mathrm{BBox}(\bar{p}_j, \bar{p}_{j\pm1})$ liegt. Weiter sei $\tau := \mathbb{R} \times [p_{k+1}^y, p_k^y]$. Wenn es Punkte aus P gibt, die im Inneren von τ enthalten sind, so müssen diese auf dem rechten Rand von β liegen. Da sie direkt über p_{k+1} liegen und das Innere von $Q(p_{k+1}, 1)$ leer ist, können diese Punkte nicht auf dem Rand einer Zusammenhangskomponente liegen. Daraus folgt, dass $\bar{p}_j$ und $\bar{p}_{j\pm1}$ nicht im Inneren von τ liegen. Analog wie für τ kann man auch für den horizontalen Streifen zwischen $\bar{p}_j$ und $\bar{p}_{j\pm1}$ schließen, dass in seinem Inneren

keine Punkte aus P enthalten sind, die auf dem Rand einer Zusammenhangskomponente liegen. Insbesondere können p_k und p_{k+1} nicht in seinem Inneren enthalten sein. Somit muss $\bar{p}_j$ auf der Höhe von p_k und $\bar{p}_{j\pm1}$ auf der Höhe von p_{k+1} liegen (o.B.d.A. $\bar{p}_j^y > \bar{p}_{j\pm1}^y$). Also folgt, dass $\bar{p}_j$ auf dem oberen Rand von β liegen muss und $\bar{p}_{j\pm1}$ nicht links von p_{k+1} liegen darf. Aus der Orientierung von $\bar{p}_j$ und $\bar{p}_{j\pm1}$ folgt mit Lemma 5.6 (i), dass $\bar{t} = 1$ oder $\bar{t} = 3$ gelten muss. Die Möglichkeit $\bar{t} = 1$ scheidet aus, da in diesem Fall $\bar{p}_j = p_k$, $\bar{p}_{j\pm1} = p_{k+1}$ und damit $s = \bar{s}$ folgern würde. Aber auch $\bar{t} = 3$ ist nicht möglich, da das Innere von $Q(p_k, 1)$ und damit auch das Innere von $Q(\bar{p}_j, 1)$ leer sind. Dieser Widerspruch beweist (i). ■

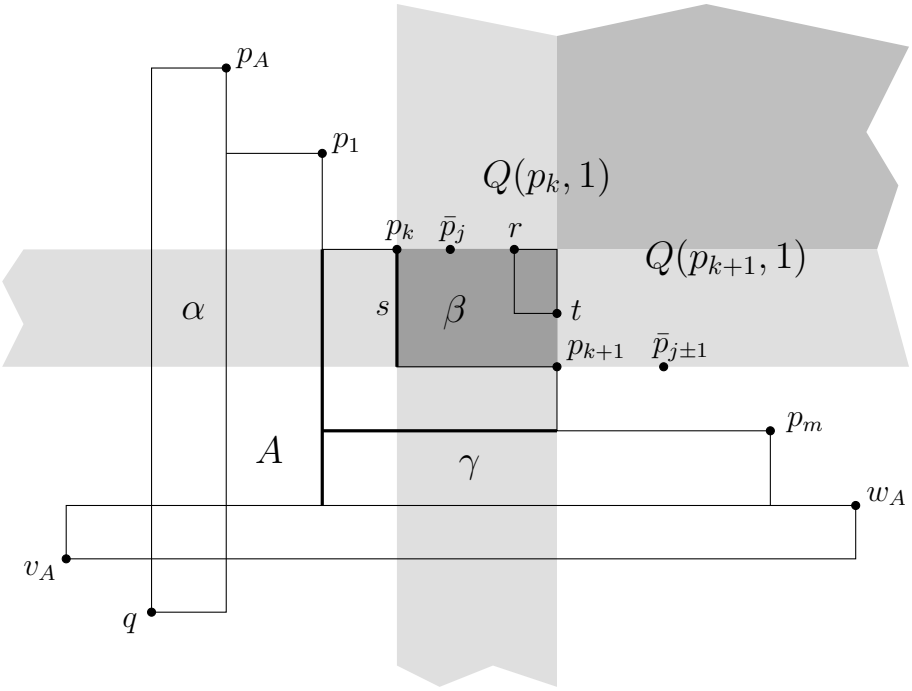


Abbildung 5.9: Zum Beweis von Lemma 5.10.

Lemma 5.11 (vgl. Lemma 9 aus [BWW04a]) *Es sei g eine horizontale Gerade mit $g \cap P = \emptyset$ und $g \cap BBox(P) \neq \emptyset$. Wir betrachten die Sequenz S der vertikalen Randsegmente und der vertikalen $\mathcal{C}$-Segmente, die g schneiden (aufsteigend geordnet nach x-Koordinaten). Dann gilt:*

(i) Es gibt höchstens ein Randsegment s in S mit der Eigenschaft, dass sein direkter Nachfolger s' in S auch ein Randsegment ist. Existiert ein solches Element, so ist entweder s oder s' ein Stufensegment.

(ii) Das erste und *das letzte Segment in S sind $\mathcal{C}$-Segmente.*

Beweis: Behauptung (ii) wurde in Lemma 9 aus [BWW04a] bewiesen. Weiter ist aus [BWW04a] bekannt, dass zwei Randsegmente, die in S direkt nebeneinander liegen, zur gleichen Zusammenhangskomponente gehören. In diesem Fall muss eines der beiden Randsegmente ein Stufensegment sein. Behauptung (i) folgt also direkt aus Lemma 5.10 (i) und Lemma 9 aus [BWW04a]. ■

Lemma 5.12 *Es sei A eine Zusammenhangskomponente aus $\delta(q,1)$, bei deren Bearbeitung das Zwischensegment $\tilde{s}_A$ in N_2 eingefügt wurde. Daraus folgt unmittelbar $v_A^y < w_A^y$. Weiter sei g eine horizontale Gerade, die das Innere von $R := BBox(v_A, w_A)$ schneidet. Wir betrachten die Sequenz S der Zwischensegmente und der vertikalen $\mathcal{C}$-Segmente, die g in R schneiden (aufsteigend geordnet nach x-Koordinaten). Dann gilt, dass der direkte Vorgänger von $\tilde{s}_A$ in S existiert und ein $\mathcal{C}$-Segment ist.*

Beweis: Wir betrachten das Rechteck $R_A := \mathrm{BBox}(q, p_A)$, welches in $\mathcal{R}_{\mathrm{ver}}$ liegt. Da $q^y \leq v_A^y \leq w_A^y \leq p_A^y$ sowie $v_A^x \leq q^x \leq p_A^x < w^x$ gilt, schneidet g beide vertikalen Ränder von R_A in R. Folglich enthält S ein $\mathcal{C}$-Segment, welches links von $\tilde{s}_A$ liegt. Da es im Inneren von R_A und zwischen R_A und $\tilde{s}_A$ nach [BWW04a] keine Zwischensegmente geben kann, ist dieses $\mathcal{C}$-Segment der direkte Vorgänger von $\tilde{s}_A$ in S (siehe auch Abbildung 5.10). ■

Betrachtet man in der Situation von Lemma 5.12 Zusammenhangskomponenten aus $\delta(q,3)$ (und fordert $v_A^y > w_A^y$), so lässt sich vollständig analog schließen, dass er direkte Nachfolger von $\tilde{s}_A$ in S existiert und ein $\mathcal{C}$-Segment ist.

Lemma 5.13 (vgl. Lemma 10 aus [BWW04a]) *Es sei g eine horizontale Gerade, die das Innere eines Rechtecks $R \in \mathcal{R}_{\mathrm{hor}}$ schneidet. Wir betrachten die Sequenz S der Zwischensegmente und der vertikalen $\mathcal{C}$-Segmente, die g in R schneiden (aufsteigend geordnet nach x-Koordinaten). Dann gilt:*

(i) Es gibt keine zwei Zwischensegmente in S, die direkt nebeneinander liegen.

(ii) Das erste oder *das letzte Segment in S ist ein $\mathcal{C}$-Segment.*

Beweis: Es sei $R = \mathrm{BBox}(v, w)$ mit $(v, w) \in \mathcal{C}_{\mathrm{hor}}$, wobei o.B.d.A. $v^x \leq w^x$ gelte. Aus der Konstruktionsvorschrift für die Zwischen- bzw. $\mathcal{C}$-Segmente folgt, dass es keine zwei Segmente in S geben kann, die sich überlappen (d. h. die gleiche x-Koordinate haben). Da g das Innere von R schneidet, muss $v^y \neq w^y$ gelten. Nach Lemma 5.5 – angewandt auf v und w – schneidet g ein vertikales $\mathcal{C}$-Segment, welches in R liegt. Folglich ist S nicht leer. Es sei $S' = (\tilde{s}_{A_1}, \ldots, \tilde{s}_{A_m})$ die – aufsteigend nach x-Koordinaten geordnete – Sequenz der Zwischensegmente, die in S enthalten sind. Dabei seien A_i, q_i und t_i $(i = 1, \ldots, m)$ derart, dass A_i die Zusammenhangskomponente aus $\delta(q_i, t_i)$ ist, bei deren Bearbeitung $\tilde{s}_{A_i}$ in N_2 eingefügt wurde Es gilt also $\{v_{A_i}, w_{A_i}\} = \{v, w\}$. Wir nehmen außerdem $v^y < w^y$ an, der Fall $v^y > w^y$ verläuft analog. In Lemma 5.6 (i) – angewandt auf v und w – wurde gezeigt, dass $t_i = 1$ oder $t_i = 3$ gelten muss. Weiter folgt aus Lemma 5.6 (ii) dass A_i für $t_i = 1$ oberhalb und für $t_i = 3$ unterhalb von R liegt. Wir nehmen o.B.d.A. an, dass $m \geq 1$ gilt (für $m = 1$ ist die Aussage trivial) und dass es ein A_i mit $t_i = 1$ gibt (ansonsten gibt es ein A_i mit $t_i = 3$ und der Beweis verläuft analog), siehe auch Abbildung 5.10.

Es sei $\tilde{s}_{A_i}$ ein Zwischensegment mit $t_i = 1$. Nach Lemma 5.12 ist der direkte Vorgänger von $\tilde{s}_{A_i}$ in S ein $\mathcal{C}$-Segment. Folglich liegt in S zwischen zwei Zwischensegmenten $\tilde{s}_{A_i}$ und $\tilde{s}_{A_{i+1}}$ mit $t_i = t_{i+1} = 1$ immer ein $\mathcal{C}$-Segment. Analoges lässt sich für $t_i = t_{i+1} = 3$ schließen. Zum Nachweis von (i) müssen wir also noch zeigen, das in S auch zwischen zwei Zwischensegmenten $\tilde{s}_{A_i}$ und $\tilde{s}_{A_{i+1}}$ mit $t_i \neq t_{i+1}$ immer ein

$\mathcal{C}$-Segment liegt. Dazu seien x, s_v, s_w und s definiert wie in *Änderung 2*. ($\{v, w\}$ ist in Z_{hor} enthalten, da R zu $\mathcal{R}_{\text{hor}}$ gehört). Für $t_i = 1$ muss $\tilde{s}_{A_i}$ links von s liegen und für $t_i = 3$ muss $\tilde{s}_{A_i}$ rechts von s liegen, ansonsten wäre $\tilde{s}_{A_i}$ laut *Änderung 3* nicht eingefügt worden. Folglich liegen in S' alle $\tilde{s}_{A_i}$ mit $t_i = 1$ links von den Segmenten $\tilde{s}_{A_j}$ mit $t_j = 3$. Aus diesem Grund gibt es höchstens ein $\tilde{s}_{A_i}$ mit $t_i \neq t_{i+1}$. Existiert dieses $\tilde{s}_{A_i}$, so gilt $t_i = 1$ und $t_{i+1} = 3$. Um (i) zu beweisen, müssen wir also noch zeigen, dass es in S ein $\mathcal{C}$-Segment zwischen $\tilde{s}_{A_i}$ und $\tilde{s}_{A_{i+1}}$ gibt. Dazu sei p_{l} der Punkt aus dem Schnitt von A_i und P, der die größte x-Koordinate besitzt. Weiter sei p_{r} der Punkt aus dem Schnitt von A_{i+1} und P, der die kleinste x-Koordinate besitzt. Wegen der Definition von Z_{quad} gilt $p_{\text{l}}^x < p_{\text{r}}^x$ und wegen Lemma 5.6 (ii) gilt $p_{\text{l}}^y > p_{\text{r}}^y$. Nach Lemma 5.5 enthält S ein vertikales $\mathcal{C}$-Segment, welches in $\text{BBox}(p_{\text{l}}, p_{\text{r}})$ liegt. Dieses $\mathcal{C}$-Segment liegt zwischen $\tilde{s}_{A_i}$ und $\tilde{s}_{A_{i+1}}$. Damit ist (i) bewiesen.

Für den Beweis von (ii) erinnern wir daran, dass es nach Voraussetzung ein A_i mit $t_i = 1$ gibt. Folglich gilt insbesondere $t_1 = 1$. Damit hat $\tilde{s}_{A_1}$ in S nach Lemma 5.12 ein $\mathcal{C}$-Segment als Vorgänger. Dies beweist (ii). Gibt es sowohl ein $\tilde{s}_{A_i}$ mit $t_i = 1$ als auch ein $\tilde{s}_{A_j}$ mit $t_j = 3$, so kann man (mit analogen Schlussfolgerungen) sogar zeigen, dass das erste *und* das letzte Segment in S Segmente aus $\mathcal{C}_{\text{ver}}$ sind. ■

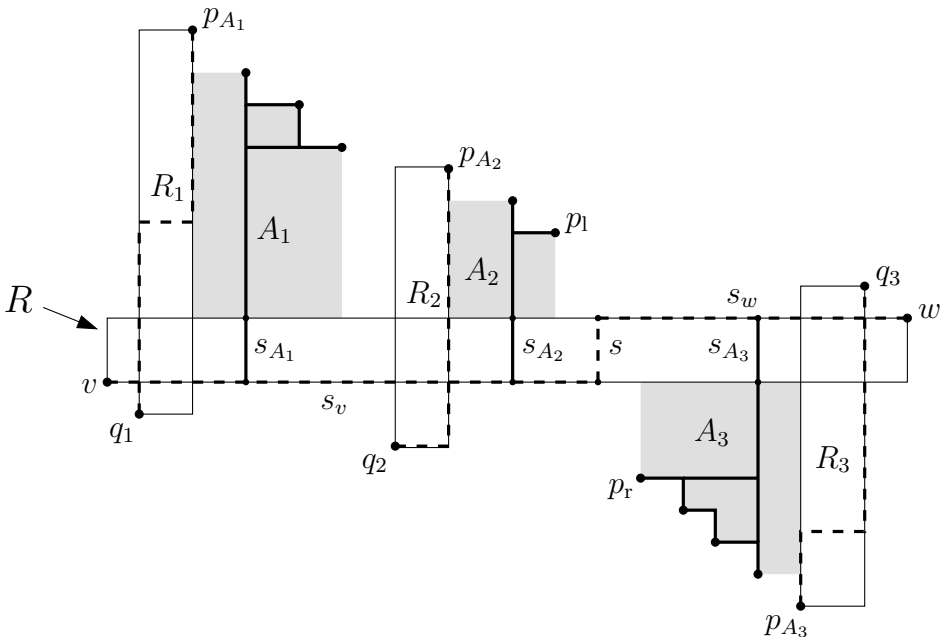


Abbildung 5.10: Zum Beweis von Lemma 5.13.

Wie der interessierte Leser bemerkt hat, haben wir in dem Beweis keinen Gebrauch davon gemacht, dass P eine Hüllenmenge ist. In der Tat gilt diese Verschärfung von Lemma 10 aus [BWW04a] auch im allgemeinen Fall. Nun können wir Lemma 5.7 beweisen.

Beweis von Lemma 5.7: Wir beweisen nur die Behauptung $|N_2^{\text{ver}}| \leq |\mathcal{C}_{\text{ver}}| - |\mathcal{S}^{\text{ver}}|$. Der Beweis der zweiten Behauptung geht nach dem gleichen Prinzip. Er ist allerdings deutlich einfacher, da es keine horizontalen Zwischensegmente oder horizontale Segmente der Art $\text{Seg}\,[\tilde{q}_A, \hat{q}_A]$ gibt (siehe *Änderung 3*). Das Grundprinzip des Beweises ist das gleiche wie in Lemma 11 aus [BWW04a]. Es sei g eine horizontale Gerade

mit $g \cap P = \emptyset$ und $g \cap \mathrm{BBox}(P) \neq \emptyset$. Wir betrachten die Sequenz S der vertikalen Segmente aus N_2 und $\mathcal{C}_{\mathrm{ver}}$, die g schneiden (geordnet nach den x-Koordinaten). Aus den Konstruktionsvorschriften für die Segmente in N_2 und $\mathcal{C}_{\mathrm{ver}}$ folgt, dass es keine zwei Segmente in S geben kann, die sich überlappen (d. h. die gleiche x-Koordinate haben). Die Anzahl der Segmente in S, die in N_2^{ver} bzw. $\mathcal{C}_{\mathrm{ver}}$ liegen, bezeichnen wir mit $\sharp N_2^{\mathrm{ver}}$ bzw. $\sharp \mathcal{C}_{\mathrm{ver}}$. Falls g kein vertikales Segment aus $\mathcal{S}$ schneidet, werden wir die Abschätzung $\sharp N_2^{\mathrm{ver}} \leq \sharp \mathcal{C}_{\mathrm{ver}}$ zeigen. Falls g ein vertikales Segment aus $\mathcal{S}$ schneiden könnte, werden wir $\sharp N_2^{\mathrm{ver}} \leq \sharp \mathcal{C}_{\mathrm{ver}} - 1$ zeigen. Daraus folgt (mit einem „horizontal sweep“) unsere gewünschte Aussage $|N_2^{\mathrm{ver}}| \leq |\mathcal{C}_{\mathrm{ver}}| - |\mathcal{S}^{\mathrm{ver}}|$.

Falls S ein Stufensegment enthält, so kann S nach Lemma 5.10 keine Zwischensegmente enthalten. Folglich liegen nur Rand- und $\mathcal{C}$-Segmente in S und Lemma 5.11 liefert uns die Abschätzung $\sharp N_2^{\mathrm{ver}} \leq \sharp \mathcal{C}_{\mathrm{ver}}$ (siehe auch Abbildung 5.11 (a)). Man beachte, dass Lemma 5.10 und 5.11 nur für Hüllenmengen gelten. Außerdem schneidet g – ebenfalls nach Lemma 5.10 – kein vertikales Segment aus $\mathcal{S}$. Damit ist die erste Abschätzung gezeigt. Im restlichen Beweis nehmen wir an, dass S keine Stufensegmente enthält. Wir zeigen Folgendes:

(1) Es gibt in S keine zwei Segmente aus N_2, die direkt nebeneinander liegen.

(2) Das erste Segment in S ist ein $\mathcal{C}$-Segment.

(3) Das letzte Segment in S ist ein $\mathcal{C}$-Segment.

Dies liefert uns die Abschätzung $\sharp N_2^{\mathrm{ver}} \leq \sharp \mathcal{C}_{\mathrm{ver}} - 1$ (siehe auch Abbildung 5.11 (b)). Falls S keine Zwischensegmente enthält, können wir wieder Lemma 5.11 anwenden. Da S nach Voraussetzung keine Stufensegmente enthält, liefert es uns genau (1), (2) und (3). Falls S (mindestens) ein Zwischensegment $\tilde{s}_A$ enthält, sei $R = \mathrm{BBox}(v, w)$ das Rechteck aus $\mathcal{R}_{\mathrm{hor}}$, in dem $\tilde{s}_A$ liegt. Es gelte o.B.d.A. $v^x < w^x$. Wir nehmen außerdem $v^y < w^y$ an, der Fall $v^y > w^y$ verläuft analog. Siehe auch Abbildung 5.11 (c).

Wir definieren drei Teilsequenzen von S, die insbesondere bezüglich den x-Koordinaten der Segmente geordnet sind. S_{m} sei die Sequenz aller Segmente aus S, die R schneiden und *keine* Randsegmente sind. Da nach Konstruktion keine Randsegmente im *Inneren* von R liegen können, ist S_{m} eine zusammenhängende Teilsequenz von S. Außerdem ist S_{m} nicht die leere Sequenz, denn sie enthält $\tilde{s}_A$. Für S_{m} können wir Lemma 5.13 anwenden. Insbesondere liefert uns Lemma 5.13 (i), dass keine zwei Segmente aus N_2 direkt nebeneinander liegen. Weiter sei S_{l} (S_{r}) die Sequenz aller Segmente aus S, die in S links (rechts) von S_{m} liegen. Aus der Definition von Z_{hor} und der Lage von g folgt, dass g nur ein einziges Rechteck aus $\mathcal{R}_{\mathrm{hor}}$ schneidet (in unserem Fall R). Da Zwischensegmente nach Konstruktion immer in Rechtecken aus $\mathcal{R}_{\mathrm{hor}}$ liegen, sind alle Zwischensegmente aus S auch in S_{m} enthalten. Folglich enthalten S_{l} und S_{r} nur Randsegmente und $\mathcal{C}$-Segmente. Falls S_{l} (S_{r}) nicht die leere Sequenz ist, können wir analog zu dem Beweis von Lemma 9 aus [BWW04a] für S_{l} (S_{r}) zeigen, dass keine zwei Segmente aus N_2 direkt nebeneinander liegen, da S keine Stufensegmente enthält, und dass das erste (letzte) Segment in $\mathcal{C}_{\mathrm{ver}}$ liegt. Dies beweist (2) ((3)), falls S_{l} (S_{r}) nicht leer ist. Das letzte (erste) Segmente von S_{l} (S_{r}) muss allerdings nicht unbedingt ein $\mathcal{C}$-Segment sein.

Bis jetzt haben wir gezeigt, dass in S_{l}, S_{m} und S_{r} keine zwei Segmente aus N_2 direkt nebeneinander liegen. Wegen Lemma 5.13 (ii) dürfe wir außerdem o.B.d.A.

annehmen, dass das erste Segment in S_m ein $\mathcal{C}$-Segment ist (dies beweist auch (2), falls S_l leer ist). Um (1) nachzuweisen müssen wir also noch zeigen, dass das letzte Segment in S_m oder das erste Segment in S_r ein $\mathcal{C}$-Segment ist. Dazu nehmen wir an, dass das letzte Segment von S_m ein Zwischensegment $\tilde{s}_A$ ist und zeigen, dass in diesem Fall das erste Segment in S_r existiert und ein $\mathcal{C}$-Segment ist (siehe auch Abbildung 5.12). Dies beweist auch (3), falls S_r leer ist.

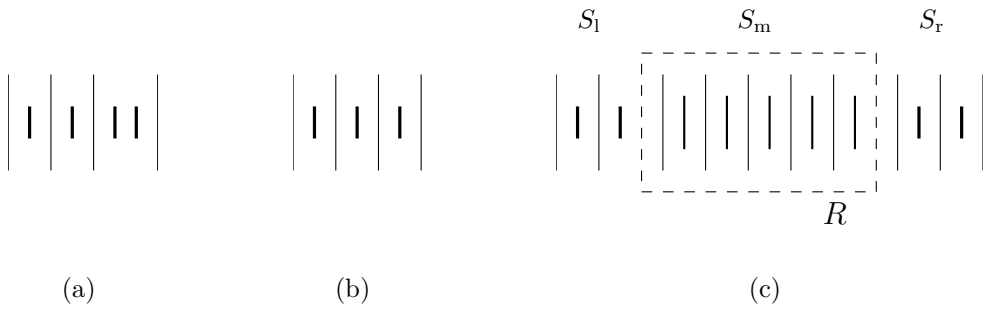


Abbildung 5.11: Worst-Case-Szenarien für S. Die langen Strecken entsprechen $\mathcal{C}$-Segmenten, die mittleren Strecken entsprechen Zwischensegmenten und die kurzen Strecken entsprechen Randsegmenten.

Es sei A die Zusammenhangskomponente aus $\delta(q,t)$, bei deren Bearbeitung $\tilde{s}_A$ in N_2 eingefügt wurde. Aus Lemma 5.12 – in seiner analogen Form für $t = 3$ – folgt, dass $t = 1$ gelten muss (ansonsten wäre das letzte Segment von S_m kein Zwischensegment). Es sei $A \cap P = \{p_1, \ldots, p_m\}$ mit $p_i^x < p_{i+1}^x$ $(i = 1, \ldots, m-1)$. Weiter seien x, s_v, s_w und s definiert wie in *Änderung 2*. Da $\tilde{s}_A$ in N_2 eingefügt wurde, muss $\tilde{s}_A$ links von s liegen. Es gilt also $p_1^x < x$ (siehe *Änderung 3*) und somit auch $p_A^x < x$. Dies bedeutet, dass *Änderung 2* nicht auf R angewandt wurde, ansonsten müsste nämlich $p_A^x = x$ gelten. Folglich muss mindestens eine der Bedingungen (i), (ii) oder (iii) aus *Änderung 2* verletzt sein. Da $x > p_A^x$ – und damit (iii) – gilt, darf höchstens eine der beiden Bedingungen (i) und (ii) erfüllt sein. Im Falle $w.\mathrm{xnbor}\,[3] := r \neq q$ (d. h. (i) ist nicht erfüllt) muss $r^x > p_m^x$ gelten, ansonsten wäre (p_m, q) nicht in Z_quad. Außerdem gilt $v^y > r^y$, da r nicht in R liegen kann. Damit gibt es nach Lemma 5.5 – angewandt auf w und r – ein vertikales $\mathcal{C}$-Segment, das rechts von $\tilde{s}_A$ und in R liegt. Dies ist ein Widerspruch zu der Annahme, dass $\tilde{s}_A$ das letzte Segment in S_m ist. Also bleibt nur noch die Möglichkeit, dass (ii) nicht erfüllt ist. Dies bedeutet, dass das Innere von $Q(w,4)$ nicht leer ist. Folglich gibt es einen Punkt $\tilde{r} \in P$ mit $\tilde{r}^x > w^x$ und $\tilde{r}^y < v^y$ (im Falle $v^y \leq \tilde{r}^y < w^y$ wäre (v,w) nicht in $\mathcal{C}_\mathrm{hor}$ enthalten). Wir nehmen o.B.d.A. an, dass $\tilde{r}$ der Punkt mit der größten y-Koordinate unter allen Punkten aus $\mathrm{I}(Q(w,4)) \cap P$ mit minimaler x-Koordinate ist. Es sei s' der obere Rand von $\mathrm{BBox}(w,\tilde{r})$. Weiter sei $t \in P$ der am weitesten rechts liegende Punkt auf s', wobei $w = t$ möglich ist. Da P eine Hüllenmenge ist, muss $\mathrm{I}(Q(w,1)) \cap P = \emptyset$ gelten (die drei anderen Quadranten scheiden wegen p_0, q und $\tilde{r}$ aus). Folglich ist $\{t, \tilde{r}\}$ in Z_ver enthalten und S_r enthält ein $\mathcal{C}$-Segment $\tilde{s}$, welches auf einem der beiden vertikalen Ränder von $\mathrm{BBox}(t,\tilde{r})$ liegt. Wir zeigen nun, dass $\tilde{s}$ das erste Segment in S_r ist. Dabei dürfen wir o.B.d.A. annehmen, dass $w \neq t$ gilt, ansonsten folgt die Behauptung trivialerweise aus der Disjunktheit von $\tilde{\mathcal{A}}_{12}$ und $\tilde{\mathcal{A}}_3$.

Wir nehmen an, dass es ein vertikales Randsegment $\hat{s}$ gibt, welches in S_r links von $\tilde{s}$ liegt, und führen dies zum Widerspruch. Es sei $\hat{A}$ die Zusammenhangskomponente aus $\delta(\hat{q}, \hat{t})$, bei deren Bearbeitung $\hat{s}$ in N_2 eingefügt wurde. Wie bei allen vertikalen Randsegmenten muss es einen Punkt $\hat{p} \in P$ geben, der die gleiche x-Koordinate wie $\hat{s}$ hat und der auf dem Rand von $\hat{A}$ liegt. Da $\tilde{r}$ (unter allen Punkten aus $\mathrm{I}(Q(w,4)) \cap P$) ein Punkt mit minimaler x-Koordinate ist, und da $\mathrm{I}(Q(w,1)) \cap P = \emptyset$ gilt, kann $\hat{p}$ nur auf s' liegen. Wegen der Lage von $\hat{p}$ und $\hat{s}$ muss $\hat{t} = 1$ oder $\hat{t} = 2$ gelten. Wir zeigen jetzt Folgendes: Für jeden Punkt $u \in P \cap s'$ gibt es einen Punkt $u_l \in Q(u,3) \cap P$ und $u_r \in Q(u,4) \cap P$, sodass $\{u, u_l\}$ und $\{u, u_r\}$ in $Z_\mathrm{hor} \cup Z_\mathrm{ver}$ liegen. Daraus folgt aus der Definition von Z_quad, dass keiner der Punkte aus $P \cap s'$ als Randpunkt von $\hat{A}$ in Frage kommt. Dies ergibt unseren Widerspruch. Somit ist $\tilde{s}$ das erste Segment in S_r.

Für $u \in (P \cap s') \backslash \{w, t\}$ sei u_l der horizontale Vorgänger und u_r der horizontale Nachfolger von u. Beide Punkte sind eindeutig und liegen in s'. Folglich gilt $\{u, u_l\} \in Z_\mathrm{hor}$ und $\{u, u_r\} \in Z_\mathrm{hor}$. Für $u = w$ sei u_r der horizontale Nachfolger von u. Dieser liegt in s'. Somit gilt $\{u, u_r\} \in Z_\mathrm{hor}$. Weiter setze man $u_l := v$ (es gilt $\{v, w\} \in Z_\mathrm{hor}$). Für $u = t$ sei u_l der horizontale Vorgänger von u. Dieser liegt in s'. Folglich gilt $\{u, u_l\} \in Z_\mathrm{hor}$. Weiter setze man $u_r := \tilde{r}$ (es gilt $\{t, \tilde{r}\} \in Z_\mathrm{ver}$). Offensichtlich liegt in allen drei Fällen u_l in $Q(u,3)$ und u_r in $Q(u,4)$. Dies beweist die letzte offene Behauptung und damit das Lemma. ■

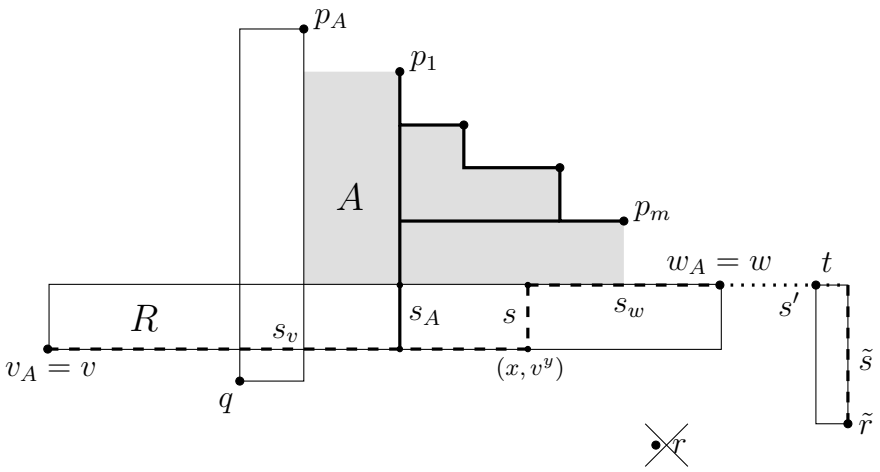


Abbildung 5.12: Zum Beweis von Lemma 5.7.

6. Zusammenfassung und Ausblick

Wir haben in dieser Studienarbeit einen Algorithmus entwickelt, der im Inneren eines Stufenpolygons eine minimale Zerlegung approximiert. Die von uns definierte (minimale) Zerlegung eines Stufenpolygons ist ein Teilproblem, welches bei der Approximation minimaler Manhattan-Netzwerke in dem Algorithmus *ApproxMMN* aus [BWW04a] auftritt. Da unser Algorithmus einen Approximationsfaktor von 2 besitzt und seine Laufzeit in $O(n \log n)$ liegt, erfüllt er die Bedingungen, die notwendig sind, um dieses Teilproblem geeignet zu lösen. Der Algorithmus *ApproxMMN* hat einen Approximationsfaktor von 3 und seine Laufzeit liegt in $O(n \log n)$.

Weiter haben wir gezeigt, dass der Algorithmus *ApproxMMN* auf Hüllenmengen sogar einen Approximationsfaktor von 2 besitzt. Hüllenmengen sind Mengen, deren Punkte sämtlich auf dem Rand der Pareto-Hülle liegen. In diesem Fall ist der Algorithmus *ApproxMMN* – zumindest asymptotisch – deutlich schneller als der Algorithmus aus [CNV04], da dieser ein lineares Programm lösen muss. Allerdings besitzt der Algorithmus aus [CNV04] für beliebige Eingaben einen Approximationsfaktor von 2.

Eine interessante Frage, die – wie bereits erwähnt – bis jetzt noch völlig offen ist, ist die Komplexität der Suche nach minimalen Manhattan-Netzwerken. Ist das Problem NP-schwer, liegt es in P oder in einer Klasse zwischen P und NP (falls diese überhaupt existieren)? Auch wäre es von Interesse, ein PTAS zu finden oder aber nachzuweisen, dass es ein solches nicht geben kann.

Manhattan-Netzwerke lassen sich folgendermaßen verallgemeinern: Anstatt für alle Punktepaare zu verlangen, dass zwischen ihnen ein Manhattan-Weg existiert, fordert man dies nur noch für bestimmte Punktepaare. Dazu erweitert man die Menge P von Punkten um eine Menge M_P von Punktepaaren aus P. Ein *verallgemeinertes Manhattan-Netzwerk* muss dann alle Punktepaare aus M_P durch Manhattan-Wege verbinden. Ist M_P eine generierende Menge, so stimmen das Manhattan-Netzwerk und das verallgemeinerte Manhattan-Netzwerk überein. Das Manhattan-Netzwerk-Problem ist also ein Spezialfall des verallgemeinerten Manhattan-Netzwerk-Problems. Ein anderer Spezialfall wäre zu fordern, dass alle Punkte aus P mit einem Punkt aus P per Manhattan-Weg verbunden sind.

Auf das (minimale) verallgemeinerte Manhattan-Netzwerk lässt sich keiner der bisherigen Approximationsalgorithmen für Manhattan-Netzwerke kanonisch übertragen. Alle Algorithmen arbeiten nämlich mit einer bestimmten generierenden Menge, die im allgemeinen Fall aber nicht mehr zur Verfügung steht.

Literaturverzeichnis

[BWW04a] Benkert, Marc, Florian Widmann und Alexander Wolff: *The Minimum Manhattan Network Problem: A Fast Factor-3 Approximation.* Technischer Bericht 2004-16, Fakultät für Informatik, Universität Karlsruhe, 2004. `http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=/ira/2004/16` (Februar 2005).

[BWW04b] Benkert, Marc, Florian Widmann und Alexander Wolff: *The Minimum Manhattan Network Problem: A Fast Factor-3 Approximation.* In: *Proc. 8th Japanese Conf. on Discrete and Computational Geometry (JCDCG'04)*, Seiten 85–86, Tokyo, 8.–11. Oktober 2004.

[CNV04] Chepoi, Victor, Karim Nouioua und Yann Vaxès: *A rounding algorithm for approximating minimum Manhattan networks.* 2004. `http://www.lif-sud.univ-mrs.fr/~chepoi/l1_network5.ps` (Januar 2005).

[GLN01] Gudmundsson, Joachim, Christos Levcopoulos und Giri Narasimhan: *Approximating a Minimum Manhattan Network.* Nordic Journal of Computing, 8:219–232, 2001.

[KIA02] Kato, Ryo, Keiko Imai und Takao Asano: *An Improved Algorithm for the Minimum Manhattan Network Problem.* In: *Proc. 13th Annual International Symposium on Algorithms and Computation (ISAAC'02)*, Band 2518 der Reihe *Lecture Notes in Computer Science*, Seiten 344–356. Springer-Verlag, 20.–23. November 2002.

[LAP03] Lam, Fumei, Marina Alexandersson und Lior Pachter: *Picking Alignments from (Steiner) Trees.* Journal of Computational Biology, 10:509–520, 2003.

[LÖ96] Levcopoulos, Christos und Anna Östlin: *Linear-Time Heuristics for Minimum Weight Rectangulation (Extended Abstract).* In: *SWAT '96, 5th Scandinavian Workshop on Algorithm Theory*, Band 1097 der Reihe *Lecture Notes in Computer Science*, Seiten 271–283. Springer, 3.–5. Juli 1996.

[LPRS82] Lingas, Andrzej, Ron Pinter, Ronald Rivest und Adi Shamir: *Minimum Edge Length Partitioning of Rectilinear Polygons.* In: *Proceedings of the 20th Annual Allerton Conference on Communication, Control and Computing*, Seiten 53–63, 1982.

[Man89] MANBER, UDI: *Introduction to Algorithms.* Addison-Wesley, Reading, Massachusetts, 1989.

Index

Breiten, 6

effiziente Punkte, 30

Höhen, 6
horizontale Strecken, 6
Hüllenmenge, 31

induzierte Gitterpunkte, 8
induziertes Gitter, 9

Länge
    einer Manhattan-Strecke, 6
    einer Menge von
        Manhattan-Strecken, 6

Manhattan-Netzwerk, 8
    minimales MN, 8
Manhattan-Polygon, 16
Manhattan-Strecke, 6
    Berührung, 6
    innere Punkte, 6
    Kreuzung, 6
    Länge, 6
    Teilstrecke, 6
    Überlappung, 6
Manhattan-Weg, 7

orthokonvex, 30

Pareto-Hülle, 30

Stufenpolygon, 16
    Inneres, 16
Stufenpolygonsequenz, 15
    Eckpunkt, 16
Stufensegment, 39

vertikale Strecken, 6

x-Werte, 5

y-Werte, 5

Zerlegung, 17
    minimale Zerlegung, 17