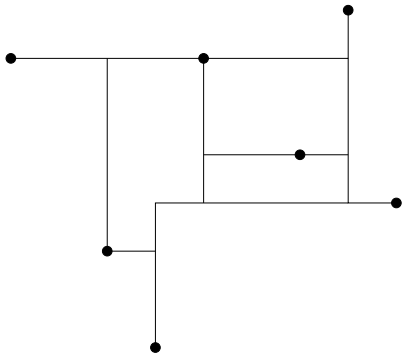
Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik I


Martin Köhler

Berechnung minimaler Manhattan-Netzwerke

12. Juni 2006



Diplomarbeit

Betreuer: Prof. Dr. Rolf Klein

Erklärung

Mit der Abgabe der Diplomarbeit versichere ich, gemäß §19 Absatz 7 der DPO vom 15. August 1998, dass ich die Arbeit selbstständig durchgeführt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Wesseling, den 12. Juni 2006 Martin Köhler

Inhaltsverzeichnis

Abbildungsverzeichnis

Algorithmenverzeichnis

1 Einführung

Angenommen, die Stadtverwaltung von Manhattan möchte Geld bei der Sanierung der Straßen sparen. Andererseits möchte sie bei Touristen auch keinen schlechten Eindruck hinterlassen. Da stellt sich dann die Frage, wieviele und welche Kilometer Straße müssen instand gehalten werden, damit die Touristen von jeder Sehenswürdigkeit zu jeder anderen Sehenswürdigkeit kommen, ohne einen Umweg oder über nicht instand gesetzte Straßen fahren zu müssen.

Ein Manhattan-Netzwerk zu einer Punktmenge ist die Verbindung der Punkte untereinander nur mit vertikalen und horizontalen Liniensegmenten, so dass jeder Punkt mit jedem auf einem kürzesten, rechtwinkligen Weg verbunden ist. Minimiert man die Gesamtlänge des Netzwerkes, so erhält man ein minimales Manhattan-Netzwerk (vgl. Abbildung 1.1).

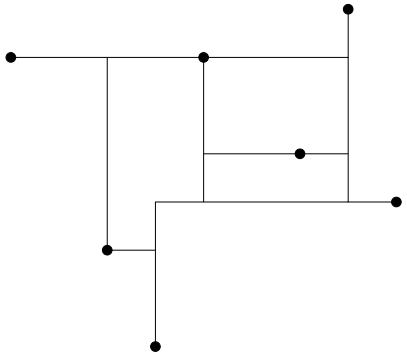


Abbildung 1.1: Ein minimales Manhattan-Netzwerk

Minimale Manhattan-Netzwerke haben Anwendungszwecke beispielsweise im Chip-Design. Insbesondere wenn, wie bei der „very-large-scale integration“ (VLSI), die Anzahl der Transistoren auf einem Chip extrem hoch ist. Diese werden meistens auch nur auf rechtwinkligen Pfaden verbunden. Wenn auch nicht jeder Transistor mit jedem verbunden wird, können die Algorithmen aus dem Bereich der minimalen Manhattan-Netzwerke dafür unter Umständen dennoch interessante Ansätze liefern. Auch in der Stadtplanung können Techniken aus dem Bereich von minimalen Manhattan-Netzwerken Anwendung finden.

Aber auch in Gebieten, wo man es nicht direkt vermutet, gibt es Anwendungszwecke für minimale Manhattan-Netzwerke. Für die Anordnung von Gen-Sequenzen schlagen Lam, Alexandersson und Pachter [LAP03] einen drei-Phasen Algorithmus vor. In der ersten Phase nutzen sie einen lokalen Anordnungs-Algorithmus, um Sequenzen von hoher

Ähnlichkeit zu identifizieren, die so genannten „high-scoring-pairs“ (HSP). Im zweiten Schritt wird ein Netzwerk von bestimmten Punkten dieser HSP konstruiert. Dabei muss nicht jeder Punkt mit jedem auf einem Manhattan-Pfad verbunden werden, sondern nur mit Punkten, die größere x-Koordinaten oder größere y-Koordinaten haben. Dazu nutzen sie einen modifizierten Algorithmus von Gudmundsson et. al [GLN01], der für ihr Problem eine 2-Approximation liefert und in Zeit $O(n^3)$ läuft.

1.1 Gliederung

In dieser Diplomarbeit geht um die effiziente Berechnung von minimalen Manhattan-Netzwerken. Bis jetzt ist nicht bekannt, ob diese Berechnung NP-hart ist, oder nicht. In diesem Kapitel werden im folgenden erstmal grundlegende, meist geometrische, Definition gegeben, die immer wieder benötigt werden. Im darauf folgenden Kapitel 2 werden verschiedene Eigenschaften von minimalen Manhattan-Netzwerken bewiesen. Mit ihrer Hilfe lässt sich das Problem der minimalen Manhattan-Netzwerke besser charakterisieren und verstehen. In Kapitel 3 werden dann die bekannten Approximations-Algorithmen vorgestellt und miteinander verglichen. Im Kapitel 4 geht es dann um die exakte Berechnung von speziellen, minimalen Manhattan-Netzwerken, die in dieser Form bis jetzt noch nicht betrachtet worden sind. Abschließend wird in Kapitel 5 ein Ausblick auf die offenen Probleme gegeben.

1.2 Definitionen

Im Folgenden gilt generell folgende Notation:

S	bezeichnet eine Punktmenge
v, p, q, r	sind Punkte
a, b, c, d, g, k, l	sind Kanten oder Geraden
x_p	ist die x-Koordinate des Punktes p
y_p	ist die y-Koordinate des Punktes p
x_l	ist die x-Koordinate der vertikalen Kante l
y_l	ist die y-Koordinate der horizontalen Kante l
$\mathrm{Seg}[p, q]$	ist das Liniensegment zwischen p und q

Definition 1.1 (Vergleich von Punkt und Menge) *Sei S eine Menge von Punkten und p ein Punkt. Dann ist der Vergleich wie folgt definiert:*

$$\begin{array}{lllll} x_p & > & x_S & \Leftrightarrow & \forall q \in S: \; x_p > x_q \\ x_p & < & x_S & \Leftrightarrow & \forall q \in S: \; x_p < x_q \\ y_p & > & y_S & \Leftrightarrow & \forall q \in S: \; y_p > y_q \\ y_p & < & y_S & \Leftrightarrow & \forall q \in S: \; y_p < y_q \end{array}$$

Weiterhin gilt $x_p \gtrless x_S \Leftrightarrow x_S \lessgtr x_p$ und $y_p \gtrless y_S \Leftrightarrow y_S \lessgtr y_p$.

Dieser Vergleich ist nicht für alle möglichen Fälle definiert. Sollte

$$\min\{x_q | q \in S\} \leq x_p \leq \max\{x_q | q \in S\}$$

gelten, so gilt weder $x_p > x_S$, noch $x_p < x_S$.

Definition 1.2 (Vergleich zweier Mengen) *Seien P und S zwei Punktmengen. Der Vergleich ist dann wie folgt definiert:*

$$\begin{array}{lllll} x_P & > & x_S & \Leftrightarrow & \forall q \in S: \quad x_P > x_q \\ x_P & < & x_S & \Leftrightarrow & \forall q \in S: \quad x_P < x_q \\ y_P & > & y_S & \Leftrightarrow & \forall q \in S: \quad y_P > y_q \\ y_P & < & y_S & \Leftrightarrow & \forall q \in S: \quad y_P < y_q \end{array}$$

Ebenso wie beim Vergleich von Punkt und Menge, ist auch dieser Vergleich nicht in allen Fällen definiert.

Definition 1.3 (Quadranten) *Zu einem Punkt p seien die vier Quadranten wie folgt definiert (vgl. Abbildung 1.2):*

$$\begin{array}{lll} Q(p,1) & := & \left\{s \in \mathbb{R}^2 | x_p \leq x_s \wedge y_p \leq y_s\right\} \\ Q(p,2) & := & \left\{s \in \mathbb{R}^2 | x_p \geq x_s \wedge y_p \leq y_s\right\} \\ Q(p,3) & := & \left\{s \in \mathbb{R}^2 | x_p \geq x_s \wedge y_p \geq y_s\right\} \\ Q(p,4) & := & \left\{s \in \mathbb{R}^2 | x_p \leq x_s \wedge y_p \geq y_s\right\} \end{array}$$

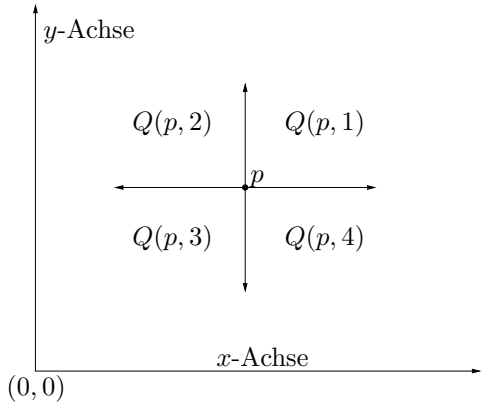


Abbildung 1.2: Die vier Quadranten bezüglich des Punktes p. Der Punkt muss nicht dem Ursprung entsprechen.

Diese Quadranten sind nicht disjunkt, ein Punkt q, der die gleiche x- oder y-Koordinate wie ein Punkt p hat, liegt in zwei Quadranten bezüglich p.

Definition 1.4 (Manhattan-Metrik) *In der Manhattan- oder L_1-Metrik ist der Abstand zwischen zwei Punkten p und q im $\mathbb{R}^2$ definiert als:*

$$d_{L_1}(p,q) := |x_p - x_q| + |y_p - y_q|$$

Definition 1.5 (Euklidische Metrik) *In der euklidischen oder L_2-Metrik ist der Abstand zwischen zwei Punkten p und q im $\mathbb{R}^2$ definiert als:*

$$d_{L_2}(p,q) := \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$$

Sowohl die Manhattan-Metrik, als auch die euklidische Metrik gehören zur Familie der Minkowski-Metriken:

Definition 1.6 (Minkowski-Metrik) *Die Minkowski-Metrik zu einer festen, natürlichen Zahl n ist definiert als:*

$$d_{L_n}(p,q) := \sqrt[n]{|x_p - x_q|^n + |y_p - y_q|^n}$$

Im Folgenden ist aber unter Abstand zweier Punkte oder Länge eines Pfades immer der Abstand bzw. die Länge bezüglich der Manhattan-Metrik zu verstehen, sofern nicht explizit etwas anderes gesagt wird:

$$d(p,q) := d_{L_1}(p,q)$$

Definition 1.7 (ϵ-Umgebung) *Eine ϵ-Umgebung zu einem Punkt p im $\mathbb{R}^2$ ist die wie folgt definierte Menge:*

$$U_\epsilon(p) := \left\{ q \in \mathbb{R}^2 \mid d(p,q) < \epsilon \right\}$$

Zu einer Punktmenge lässt sich auch der Rand der Punktmenge definieren. Dieser muss nicht in der Punktmenge enthalten sein.

Definition 1.8 (Rand einer Punktmenge) *Zu einer Punktmenge $P \subset \mathbb{R}^2$ ist der Rand ∂P wie folgt definiert:*

$$\partial P := \left\{ p \in \mathbb{R}^2 \mid \forall \epsilon > 0 : U_\epsilon(p) \cap P \neq \emptyset \wedge U_\epsilon(p) \cap (\mathbb{R}^2 \setminus P) \neq \emptyset \right\}$$

Jede ϵ-Umgebung um einen Punkt p, der auf dem Rand liegt, muss sowohl innerhalb, als auch außerhalb der Punktmenge liegen.

Definition 1.9 (Manhattan-Pfad) *Ein Manhattan-Pfad zwischen den Punkten p und q, $p \neq q$ ist eine Menge L von n Liniensegmenten mit folgenden Eigenschaften (vgl. Abbildung 1.3):*

1. *$L = (l_1, \ldots, l_n)$ (L besteht aus Liniensegmenten.)*
2. *$l_i = Seg[p_i, p_{i+1}]$ (Die Liniensegmente bilden eine Kette.)*
3. *$p_1 = p$ (Das erste Liniensegment startet bei p.)*
4. *$p_{n+1} = q$ (Das letzte Liniensegment endet bei q.)*
5. *für p_i, p_{i+1} gilt: $x_{p_i} = x_{p_{i+1}} \vee y_{p_i} = y_{p_{i+1}}$ (Die Liniensegmente sind achsenparallel.)*

6. $\sum_{i=1}^{n} d(p_i, p_{i+1}) = d(p, q)$ *(Der Weg ist genauso lang wie der Abstand von p und q.)*

Aus dieser Definition lassen sich folgende Eigenschaften ableiten:

- $\forall i, 1 \leq i \leq n+1$ gilt: $x_{p_i} \in \left[\min\left(x_{p_1}, x_{p_{n+1}}\right), \max\left(x_{p_1}, x_{p_{n+1}}\right)\right]$ (Alle Punkte des Pfades befinden sich im durch x_{p_1} und $x_{p_{n+1}}$ begrenzten Intervall)
- $\forall i, 1 \leq i \leq n+1$ gilt: $y_{p_i} \in \left[\min\left(y_{p_1}, y_{p_{n+1}}\right), \max\left(y_{p_1}, y_{p_{n+1}}\right)\right]$ (Alle Punkte des Pfades befinden sich im durch y_{p_1} und $y_{p_{n+1}}$ begrenzten Intervall)
- Eine Menge von achsenparallelen, verbundenen Liniensegmente ist genau dann ein Manhattan-Pfad, wenn diese sowohl x-, als auch y-monoton sind. Dies heißt, dass sowohl die x-, als auch die y-Koordinaten der Punkte entlang des Pfades monoton sind.

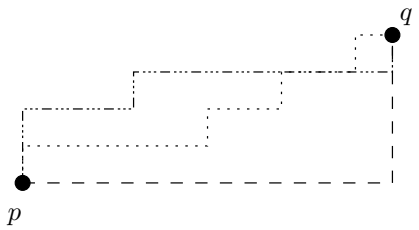


Abbildung 1.3: Drei verschiedene Manhattan-Pfade von p nach q, die die gleiche Länge haben

Wenn die zwei Punkte p und q nicht auf einer horizontalen oder vertikalen Geraden liegen, dann ist der Manhattan-Pfad zwischen diesen Punkten nicht eindeutig (vgl. Abbildung 1.3).

Definition 1.10 (Manhattan-Netzwerk) *Ein Manhattan-Netzwerk zu einer Punktmenge S ist eine Menge M von achsenparallelen Liniensegmenten mit der Eigenschaft:*

$$\forall p, q \in S, p \neq q : \exists \textit{Manhattan-Pfad zwischen } p \textit{ und } q \textit{ in } M$$

Definition 1.11 (Länge eines Manhattan-Netzwerks) *Die Länge oder Größe eines Manhattan-Netzwerkes M ist definiert als die Gesamtlänge aller Liniensegmente des Manhattan-Netzwerkes.*

$$|M| := \sum_{l \in M} |l|$$

Definition 1.12 (Minimales Manhattan-Netzwerk)
Zu einer Punktmenge S ist eine Menge M von Liniensegmenten genau dann ein minimales Manhattan-Netzwerk (vgl. Abbildung 1.1 auf Seite 7), wenn es folgende Eigenschaften erfüllt:

- *M ist ein Manhattan-Netzwerk zu S*
- *Für alle Manhattan-Netzwerke M' zu S gilt: $|M| \leq |M'|$*

Definition 1.13 (Approximation eines minimalen Manhattan-Netzwerkes) *Gegeben sei ein Algorithmus* Alg. *Dieser berechnet ein Manhattan-Netzwerk M'. Er erreicht einen Approximations-Faktor von n für das minimale Manhattan-Netzwerk-Problem, wenn für alle Punktmengen S gilt:*

$$|M'| \leq r \cdot |M_{opt}|$$

M_{opt} bezeichnet das minimale Manhattan-Netzwerk zu einer Punktmenge S.

2 Eigenschaften von minimalen Manhattan-Netzwerken

In diesem Kapitel werden wir verschiedene Eigenschaften von minimalen Manhattan-Netzwerken beweisen. Diese werden für die kommenden Kapitel, insbesondere für Kapitel 4, gebraucht. Dabei handelt es sich um Eigenschaften von minimalen Manhattan-Netzwerken zu beliebigen Punktmengen.

2.1 Ausdehnung eines Manhattan-Netzwerkes

Zuerst beschäftigen wir uns mit der Frage, innerhalb welcher Grenzen sich ein minimales Manhattan-Netzwerk aufhalten kann. Der erste Ansatz dafür ist, das umschließende Rechteck zu betrachten (vgl. Abbildung 2.1).

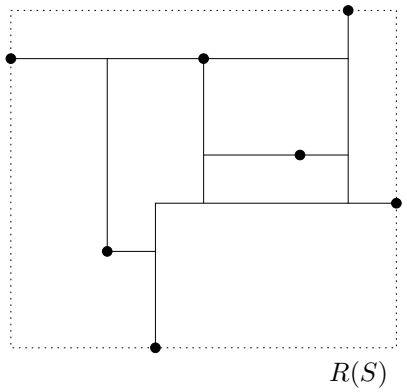


Abbildung 2.1: Minimales Manhattan-Netzwerk mit umschließendem Rechteck. Dabei liegen alle Punkte innerhalb des Rechtecks.

Definition 2.1 (Umschließendes Rechteck) *Das kleinste, achsenparallele Rechteck $R(S)$ mit der Eigenschaft, dass alle Punkt aus S in $R(S)$ enthalten sind, wird umschließendes Rechteck genannt und ist definiert als:*

$$\begin{aligned} R(S) \quad := \quad & [\min\{x_{p_1}, \ldots, x_{p_n}\}, \max\{x_{p_1}, \ldots, x_{p_n}\}] \\ \times \quad & [\min\{y_{p_1}, \ldots, y_{p_n}\}, \max\{y_{p_1}, \ldots, y_{p_n}\}] \end{aligned}$$

Definition 2.2 (Degeneriertes Rechteck) *Sollten alle Punkte aus S die gleiche x- oder die gleiche y-Koordinate haben, so besteht $R(S)$ nur aus einem Liniensegment. In diesem Fall heißt $R(S)$ degeneriert.*

Lemma 2.3 *Ein minimales Manhattan-Netzwerk muss sich im kleinsten, achsenparallelen, die Punktmenge S umschließenden Rechteck $R(S)$ befinden.*

Beweis Lemma 2.3 Gegeben sei eine Punktmenge S. Dann gilt:

- $\forall p, q \in S$: Nach Definition 1.9 gilt: Jeder Manhattan-Pfad zwischen p und q liegt in $R(\{p, q\})$.
- $\forall p, q \in S$: Da sowohl p, als auch q in $R(S)$ enthalten sind, liegt auch $R(\{p, q\})$ in $R(S)$.

□

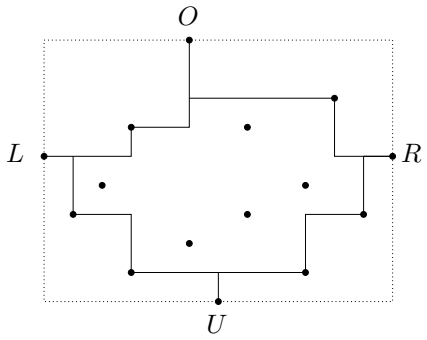


Abbildung 2.2: Kontur einer Punktmenge mit umschließendem Rechteck. Dabei liegen die vier Extrempunkte dieser Kontur auf dem umschließenden Rechteck.

Allerdings lassen sich die äußeren Grenzen für das minimale Manhattan-Netzwerk noch deutlich verfeinern. Anstelle des umschließenden Rechtecks nimmt man die Kontur der Punktmenge bezüglich der Manhattan-Metrik. Dies wurde bis jetzt noch von keinem Algorithmus in der Literatur ausgenutzt. Chepoi, Nouioua und Vaxès [CNV05] verwenden ein ähnliches Polygon, die so genannte Pareto-Hülle, was allerdings nicht so enge Grenzen wie die Kontur hat. Dieser Algorithmus wird später in Kapitel 3.3 vorgestellt.

Definition 2.4 (Kontur einer Punktmenge in der Manhattan-Metrik)
Zu einer Punktmenge S ist eine Kontur (vgl. Abbildung 2.2) ein Polygon P, mit minimaler Fläche und der Eigenschaft, dass alle Punkte aus S im Polygon enthalten sind und es zwischen je zwei Punkten $p, q \in S$ einen Manhattan-Pfad zwischen diesen Punkten gibt:

$$area(P) := \min\{area(P') | \forall p, q \in S \; \exists \textit{Manhattan-Pfad zwischen } p \textit{ und } q \textit{ in } P'\}$$

Dass das so definierte Polygon den Namen Kontur verdient, zeigt das folgende Konstruktionsverfahren, welches analog zu dem Verfahren von Rolf Klein [Kle05, Seite 167ff] ist. Da allerdings in der Manhattan-Metrik der Pfad zwischen zwei Punkten nicht immer eindeutig ist, muss das Verfahren modifiziert werden. Die zusätzliche Idee hinter dem modifizierten Verfahren ist, die Fläche des Konturpolygons zu minimieren.

Die Konstruktion einer Kontur geht in zwei Schritten. Im ersten Schritt wird eine linke Kontur erzeugt. Diese besteht aus einem Pfad vom linken Extrempunkt L zum oberen Extrempunkt O und aus einem Pfad von L zum unteren Extrempunkt U. Im zweiten Schritt wird dann die rechte Kontur, bestehend aus dem Pfad vom rechten Extrempunkt R zu O und aus dem Pfad von R zu U konstruiert.

Zur Konstruktion der linken Kontur werden die Punkte zuerst nach x-Koordinaten aufsteigend sortiert. Dann gehen wir mit einer senkrechten Sweep-Line von links nach rechts durch die Punktmenge. In der Sweep-Status-Struktur wird der Punkt mit maximaler y-Koordinate (MaxYSoFar) und der Punkt mit minimaler y-Koordinate (MinYSoFar) gespeichert. Am Anfang stimmen beide Punkte mit dem linken Extrempunkt L überein. Nach dem Sweep ist MaxYSoFar gleich O und MinYSoFar gleich U. Wenn ein neuer Punkt MaxYSoFar oder MinYSoFar gefunden wird, wird er durch einen Pfad mit seinem Vorgänger verbunden. In der Konstruktion dieses Pfades liegt der Unterschied zu dem Verfahren von Rolf Klein [Kle05]. In der L_2-Metrik ist der Pfad zwischen den beiden Punkten eine eindeutige Kante, in der L_1-Metrik ist er jedoch nicht immer eindeutig.

Sei der alte Punkt mit p_{old}, der neue Punkt mit p_{new} bezeichnet. Dann wird in dem Rechteck $R(\{p_{\text{old}}, p_{\text{new}}\})$ die rechte Kante des Rechtecks, sowie die dann nötige horizontale Kante genommen. Formal sind die beiden Kanten wie folgt definiert:

- $l_1 = \text{Seg}\,[p_{\text{old}}, (x_{p_{\text{new}}}, y_{p_{\text{old}}})]$
- $l_2 = \text{Seg}\,[(x_{p_{\text{new}}}, y_{p_{\text{old}}}), p_{\text{new}}]$

Nachdem die linke Seite der Kontur K_l berechnet ist, wird mit einer Sweep-Line von rechts nach links die rechte Seite der Kontur konstruiert. Dabei wird allerdings der Pfad zwischen zwei Punkten so konstruiert, dass die rechte Kontur die linke Kontur an möglichst vielen Stellen berührt, aber nicht kreuzt (vgl. Abbildung 2.3). Das Verfahren für die Konstruktion der linken Kontur kann nicht direkt übernommen werden, da sich sonst die linke und rechte Kontur schneiden könnte (vgl. Abbildung 2.3).

Sei der alte Punkt mit p_{old}, der neue Punkt mit p_{new} bezeichnet. Dabei wird zwischen p_{old} und p_{new} der Pfad konstruiert, der möglichst viele Kanten mit der linken Kontur gemeinsam hat. Dies geschieht nach folgendem Verfahren:

- Füge eine horizontale Kante $l_1 = \text{Seg}\,[p_{\text{old}}, (x, y_{p_{\text{old}}})]$ ein. Dabei ist x der Wert $x_{p_{\text{new}}}$, wenn dabei die linke Kontur nicht geschnitten wird, ansonsten der x-Wert des Schnittpunkts.
- Füge eine vertikale Kante l_2 mit $l_2 = \text{Seg}[(p_{\text{new}}, (x_{p_{\text{new}}}, y)]$ ein. Dabei ist y der Wert $y_{p_{\text{old}}}$, wenn dabei die linke Kontur nicht geschnitten wird, ansonsten der y-Wert des Schnittpunkts.

Sollten die Punkte $(x, y_{p_{\text{old}}})$ und $(x_{p_{\text{new}}}, y)$ nicht zusammenfallen, so verläuft zwischen diesen Punkten die rechte Kontur entlang der linken Kontur. In der Manhattan-Metrik

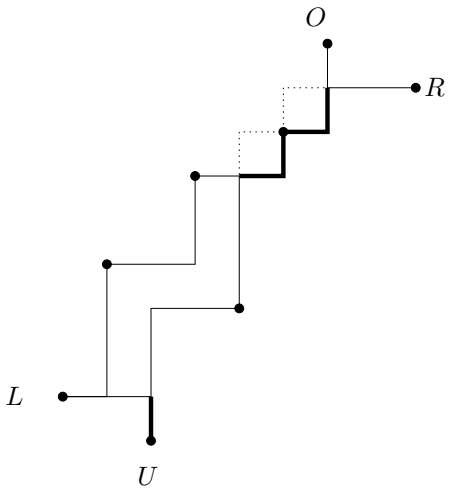


Abbildung 2.3: An den fetten Liniensegmenten verläuft die rechte Kontur entlang der linken Kontur. Bei dem oberen Teil ist der Verlauf der Kontur nicht eindeutig, die gestrichelten Kanten wären, anstelle der fetten Kanten, ebenfalls eine gültige Kontur. Wäre das gleiche Verfahren für die rechte Seite, wie für die linke Seite angewandt worden, so wären sowohl die gestrichelten als auch die fetten Segmente enthalten.

ist die Kontur, genauso wie die Manhattan-Pfade, nicht immer eindeutig (vgl. Abbildung 2.3). Im wesentlichen entspricht damit die Kontur der konvexen Hülle bezüglich der Manhattan-Metrik. Allerdings wäre die konvexe Hülle, als Schnitt aller konvexen Mengen, die S enthalten, nicht immer zusammenhängend. In Abbildung 2.3 wären die oberen, fett markierten Kanten nicht in der konvexen Hülle enthalten.

Lemma 2.5 *Dieses Polygon entspricht einer Kontur.*

Beweis Lemma 2.5 Angenommen, das so berechnete Polygon P, ist keine Kontur. Dies bedeutet, es gibt ein Polygon P' mit kleinerer Fläche, das ebenfalls für jedes Paar (p, q) einen Manhattan-Pfad enthält.

Betrachten wir zunächst die linke Seite der Kontur (vgl. Abbildung 2.4). Die Pfade entlang des linken Randes der Kontur sind so definiert, dass bei zwei aufeinanderfolgenden Punkten p und q das Innere des Rechtecks $R(\{p, q\})$ nicht im Konturpolygon enthalten ist. Damit gilt für jede vertikale Kante l der linken Seite der Kontur, dass sie sich nicht durch eine Kante l' mit $x_{l'} > x_l$ ersetzt werden kann. Weiterhin lässt sich keine horizontale Kante l des Pfades von L nach O durch eine Kante l' mit $y_{l'} < y_l$ ersetzen und es lässt sich keine horizontale Kante l des Pfades von L nach U durch eine Kante l' mit $y_{l'} > y_l$ ersetzen.

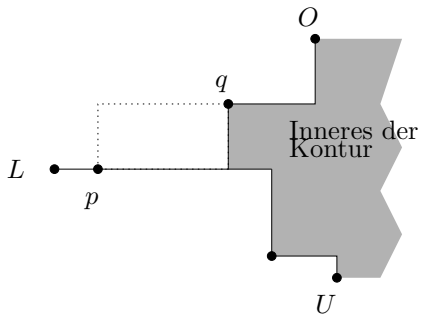


Abbildung 2.4: Die Pfade von L nach O und von L nach U werden so konstruiert, dass die Fläche des Konturpolygons dabei minimal wird. Deshalb verläuft der Pfad zwischen p und q am unteren und rechten Rand des Rechtecks.

Nun betrachten wir die rechte Seite der Kontur. Sollte diese analog zur linken Seite definiert sein, würden dann analoge Argumente für die vertikalen und horizontalen Kanten gelten. Betrachten wir eine vertikale Kante l auf der linken Seite der Kontur und eine vertikale Kante l' auf der rechten Seite der Kontur, die sich genau gegenüberliegen. Wenn $x_l < x_{l'}$ gilt, dann lässt sich der Abstand von l und l' nicht verringern, ohne dass die Kontureigenschaft verloren geht. Aufgrund der Konstruktion wird für den Fall, dass $x_l \geq x_{l'}$ gelten würde, die Kante l' nicht eingefügt sondern stattdessen die Kante l wiederverwendet. In diesem Fall gibt es aber nur ein vertikale Gerade ohne Ausdehnung und diese ist damit automatisch minimal. Analoge Argumente greifen für die obere und untere Seite der Kontur. Damit ist das so definierte Polygon eine Kontur.

□

Lemma 2.6 *Es gibt kein Polygon P, dessen Umfang kleiner ist als der Umfang eines Kontur-Polygons und das es einen Manhattan-Pfad für alle Punkte aus S in diesem Polygon gibt.*

Beweis Lemma 2.6 Dieser Beweis geht über die gleiche Argumentation wie der Beweis für das Lemma 2.5. Daher wird hier der Beweis nur kurz skizziert. Die Pfade auf der linken Seite der Kontur sind alle nötig und sollte dann für eine Kante l der linken Seite der Kontur und für die parallel Kante l' der rechten Seite der Kontur $x_l < x_{l'}$ gelten, so lässt sich der Abstand nicht verringern, womit beide Kanten nötig sind.

□

Lemma 2.7 *Es gibt zu jeder Punktmenge S ein Manhattan-Netzwerk, das nicht außerhalb der Kontur der Punktmenge liegt.*

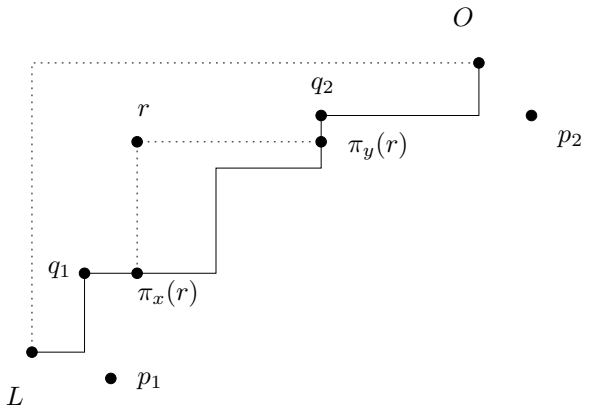


Abbildung 2.5: Der Pfad von p_1 nach p_2 geht über einen Punkt r außerhalb der Kontur. Dann ist aber auch der Pfad von p_1 über $\pi_x(r), \pi_y(r)$ nach p_2 ein Manhattan-Pfad, wobei das Stück zwischen $\pi_x(r)$ und $\pi_y(r)$ entlang der Kontur verläuft.

Beweis Lemma 2.7 Der Beweis geht über Widerspruch. Angenommen, es gibt ein Manhattan-Netzwerk, das einen Pfad enthält, der außerhalb der Kontur verläuft und nicht im Innern bzw. auf der Kontur verlaufen kann. Dieser verbindet die Punkte $p_1 \in S$ und $p_2 \in S$ und geht dabei über einen Punkt r, der außerhalb der Kontur liegt. Dieser Punkt r muss allerdings laut Lemma 2.3 innerhalb des umschließenden Rechtecks liegen. Ohne Beschränkung der Allgemeinheit gelte im Folgenden $x_{p_1} \leq x_{p_2}$.

Fall a) Der Punkt r liegt in $R(\{L, O\})$:
Bilde nun die vertikale Projektion $\pi_x(r)$ auf die Kontur und betrachte dann den linken Nachbarn q_1 auf der Kontur. Ebenso wird die horizontale Projektion $\pi_y(r)$ auf die Kontur gebildet. Dort wird dann der obere Nachbar q_2 betrachtet (vgl. Abbildung 2.5). Da p_1 und p_2 nicht außerhalb der Kontur liegen, müssen folgende Bedingungen erfüllt sein:

(i) $x_{p_1} \leq x_r < x_{q_2}$
(ii) Aus (i) folgt: $y_{p_1} \leq y_{q_1}$, da p_1 innerhalb der Kontur liegt.
(iii) Aus (ii) folgt: $y_{q_1} < y_r \leq y_{p_2}$
(iv) Aus (i) und (iii) folgt: $x_r < x_{q_2} \leq x_{p_2}$, da p_2 innerhalb der Kontur liegt.

Damit ergibt sich, dass auch der Pfad entlang der Punkte $(p_1, \pi_x(r), \pi_y(r), p_2)$ ein gültiger Manhattan-Pfad ist, wobei das Stück zwischen $\pi_x(r)$ und $\pi_y(r)$ entlang der Kontur verläuft. Dies ist jedoch ein Widerspruch zu der Annahme, dass der Pfad nicht im Innern oder auf der Kontur verlaufen kann.

Fall b) Der Punkt r liegt in einem der drei anderen Rechtecke. Diese Fälle sind analog zum ersten Fall.

□

Dieses Netzwerk ist nicht notwendigerweise minimal, es ist im Beweis nicht ausgeschlossen, dass die Gesamtlänge des Netzwerkes durch die Konstruktion wächst.

Lemma 2.8 *Es gibt zu jeder Punktmenge S ein minimales Manhattan-Netzwerk, das den Rand der Kontur enthält.*

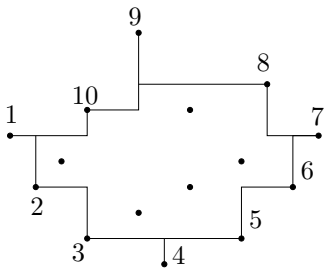


Abbildung 2.6: Punkte auf der Kontur entgegen dem Uhrzeigersinn nummeriert

Beweis Lemma 2.8 Betrachte ein minimales Manhattan-Netzwerk zu einer beliebigen Punktmenge S. Seien die Punkte auf der Kontur K gegen den Uhrzeigersinn von 1 bis k durchnummeriert. (vgl. Abbildung 2.6). Es werden nun zwei Mengen von Teil-Pfaden betrachtet.

- Die Menge A, die die Teile von Manhattan-Pfaden, die außerhalb der Kontur verlaufen, enthält.
- Die Menge B, die die Teile von Manhattan-Pfaden, die nicht außerhalb der Kontur verlaufen und die Punktpaare $\left(p_i, p_{(i \mod k)+1}\right), 1 \leq i \leq k$ verbinden.

Aufgrund der Konstruktion der Kontur verlaufen die Teil-Pfade in B entlang der Kontur. Damit sind in $A \cup B$ sämtliche Manhatten-Pfade zwischen p_i und $p_{(i \mod k)+1}, 1 \leq i \leq k$ enthalten. Da sich aufgrund von Lemma 2.7 jeder Pfad außerhalb der Kontur auf die Kontur abbilden lässt, gilt: $|K| \leq |A| + |B|$.

Mit der Konstruktion, die auch in Lemma 2.7 verwendet wird, werden die Teil-Pfade aus der Menge A auf die Kontur abgebildet. Damit verlaufen sämtliche Teil-Pfade, die vorher in der Menge A enthalten waren, jetzt entlang der Kontur. Sei nun B' definiert als die Manhatten-Pfade zwischen p_i und $p_{(i \mod k)+1}, 1 \leq i \leq k$, die auf der Kontur verlaufen. Dann gilt:

$$B' = K \Rightarrow |B'| \leq |A| + |B|$$

Da durch diese Konstruktion die Länge des Netzwerkes nicht steigen kann, bedeutet dies, dass es ein minimales Manhattan-Netzwerk gibt, das die Kontur enthält.

□

Korollar 2.9 *Es gibt ein minimales Manhattan-Netzwerk, dass nicht außerhalb der Kontur verläuft.*

Beweis Korollar 2.9 Ergibt sich aus der Konstruktion für den Beweis von Lemma 2.8. Danach sind keine Pfade außerhalb der Kontur mehr vorhanden.

□

2.2 Verlauf der Manhattan-Pfade

Auch der Verlauf der Manhattan-Pfade eines minimalen Manhattan-Netzwerkes lässt sich einschränken.

Lemma 2.10 *Es gibt ein minimales Manhattan-Netzwerk, bei dem alle Pfade zwischen zwei Kreuzungen aus maximal zwei Liniensegmenten bestehen, und zwar einem horizontalen und einem vertikalen.*

Beweis Lemma 2.10 Seien p und q zwei Kreuzungspunkte von Manhattan-Pfaden. Der Pfad zwischen p und q enthält keinen weiteren Kreuzungspunkt. Damit kann der Pfad zwischen p und q auf dem Rand des Rechtecks $R(\{p, q\})$ entlanglaufen, ohne dass sich seine Länge noch seine Monotonie-Richtung ändert. Da es nur endlich viele Kreuzungspunkte gibt, haben wir nach endlich vielen Ersetzungen das gewünschte Netzwerk erreicht.

□

Definition 2.11 (induziertes Gitter) *Das durch eine Punktmenge $S = \{p_1, \ldots, p_n\}$ induzierte Gitter Γ ist definiert durch:*

$$\begin{aligned} v(x) &:= \{(x_p, y_p) | x_p = x\} \\ h(y) &:= \{(x_p, y_p) | y_p = y\} \end{aligned}$$

$$\Gamma := \left(\bigcup_{i=1}^{n} v(x_{p_i}) \right) \cup \left(\bigcup_{i=1}^{n} h(y_{p_i}) \right)$$

Für dieses Gitter lässt sich nun folgendes Lemma zeigen, was auf den ersten Blick offensichtlich ist, bis jetzt jedoch noch nicht in dieser Allgemeinheit bewiesen wurde.

Lemma 2.12 *Es gibt ein minimales Manhattan-Netzwerk zu einer Punktmenge S, bei dem alle Manhattan-Pfade auf dem durch S induzierten Gitter Γ verlaufen.*

Beweis Lemma 2.12 Gegeben sei ein beliebiges, minimales Manhattan-Netzwerk zu einer Punktmenge S. Wir betrachten jetzt nur den horizontalen Fall, der vertikale geht analog. Betrachte ein horizontales Liniensegment l aus dem Manhattan-Netzwerk, das nicht auf dem Gitter liegt. Dieses Liniensegment hat sowohl am linken als auch am rechten Ende Segmente, die nach oben und/oder unten weggehen. Zwischen dem linken und rechten Ende gibt es n_1 Segmente, die nach oben weggehen und n_2 Segmente, die nach unten weggehen. Betrachte nun das horizontale Liniensegment l_o, das folgende Eigenschaften erfüllt:

(i) Es gibt eine vertikale Gerade g mit $g \cap l \neq \emptyset$ und $g \cap l_o \neq \emptyset$ (Die Liniensegmente liegen an mindestens einem Punkt übereinander)

(ii) Entweder l_o ist ein horizontales Liniensegment des Manhattan-Netzwerkes oder l_o entspricht dem oberen Rand des Gitterstreifens, in dem l liegt.

(iii) $y_{l_o} > y_l$ (Das Liniensegment l_o liegt über l)

(iv) Es gibt kein l'_o, dass die gleichen Eigenschaften wie l_o besitzt und für das gilt: $y_{l'_o} < y_{l_o}$. (Zwischen l und l_o liegt kein weiteres Liniensegment.)

Analog sei l_u definiert als das Liniensegment, das unterhalb von l liegt. Aus der Eigenschaft (iv) folgt, dass sowohl l_o als auch l_u in der gleichen Gitterzeile wie l liegen (vgl. Abbildung 2.7).

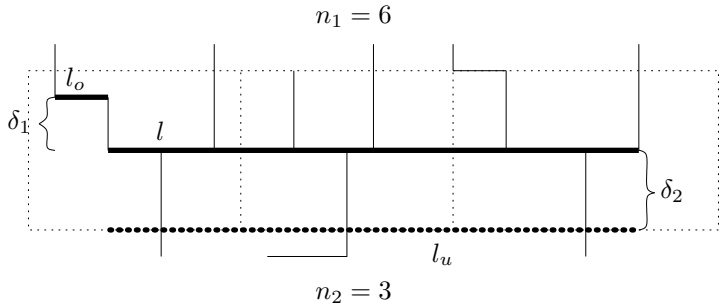


Abbildung 2.7: Das Liniensegment l lässt sich entweder bis auf die Höhe des Liniensegments l_o oder l_u verschieben. l_o ist dabei ein Teil des Manhattan-Netzwerkes, während l_u dem Gitterrand entspricht.

Der vertikale Abstand von l zu l_o sei $\delta_1 > 0$, zu l_u sei er $\delta_2 > 0$. Das Liniensegment l lässt sich nun nach oben auf jeden Fall bis auf die Höhe von l_o verschieben, ohne dass die Manhattan-Eigenschaft verletzt wird. Dabei werden die vertikalen Segmente, die oberhalb von l liegen jeweils um die Länge δ_1 gekürzt, während die vertikalen Liniensegmente, die unterhalb von l liegen um die Länge δ_1 verlängert werden. Analoges gilt für das Verschieben nach unten auf die Höhe von l_u. Beim Verschieben bis zu l_o ändert sich die Länge des Manhattan-Netzwerks um $\Delta_1 = \delta_1 \cdot (n_2 - n_1)$, beim Verschieben hin zu l_u ändert sich die Länge um: $\Delta_2 = \delta_1 \cdot (n_1 - n_2)$. Sollte sowohl Δ_1 als auch Δ_2 echt größer als Null sein, so muss $n_2 > n_1$ und $n_1 > n_2$ gelten, was nicht sein kann. Also lässt sich das Liniensegment entweder nach l_u oder l_o verschieben, ohne dass sich die Gesamtlänge des Manhattan-Netzwerk ändert. Sei l' das verschobene Liniensegment. Nun können zwei Fälle auftreten

Fall 1) l' liegt auf dem Gitter. Dann sind wir für dieses Liniensegment fertig.

Fall 2) l' liegt nicht auf dem Gitter. Dann bildet $l'' = l' \cup l_o$ bzw. $l' \cup l_u$ ein neues Liniensegment. Man betrachte nun dieses Liniensegment l''. Dafür gelten die gleichen

Argumente wie für l. Da es nur eine endliche Anzahl von horizontalen Liniensegmenten gibt und jedes mal, wenn Fall 2) auftritt, zwei horizontale Liniensegmente zu einem verschmolzen werden, muss irgendwann der Fall 1) eintreten.

Analog lässt sich dieses Verfahren danach für die vertikalen Kanten anwenden. Somit liegen danach alle Kanten des minimalen Manhattan-Netzwerkes auf dem Gitter. Die Größe des Netzwerkes ist dabei nicht gestiegen, somit ist dieses Netzwerk, was nur entlang des Gitters verläuft, ebenfalls ein minimales Manhattan-Netzwerk.

□

2.3 Konstruktion eines Netzwerkes

Auch bei der Konstruktion des minimalen Manhattan-Netzwerkes lassen sich Eigenschaften finden, die es erleichtern, ein minimale Manhattan-Netzwerk zu berechnen.

Definition 2.13 (Erzeugende Menge) *Sei S eine Punktmenge. Sei Z eine Teilmenge von $S \times S$ und M ein Netzwerk, dass für alle Paare aus Z einen Manhattan-Pfad enthält. Z ist genau dann eine erzeugende Menge, wenn M auch Pfade für Paare $(p, q), p, q \in S$ enthält.*

Definition 2.14 (Kritisches Rechteck) *Zu einer Punktmenge S ist ein umschließendes Rechteck $R(\{p, q\}), p, q \in S$ genau dann ein kritisches Rechteck, wenn gilt:*

$$R(\{p, q\}) \cap S = \{p, q\}$$

Lemma 2.15 *Die Menge $Z_\oslash$, die die Paare (p, q) enthält, für die gilt,*

$$R(\{p, q\}) \text{ ist ein kritisches Rechteck,}$$

ist eine erzeugende Menge.

Beweis Lemma 2.15 Sei M ein Netzwerk, das Manhattan-Pfade für alle Paare aus $Z_\oslash$ enthält.

Der Beweis, dass M einen Manhattan-Pfad für ein beliebiges Paar $(p, q) \in S \times S$ enthält, geht mittels Induktion über $|S \cap R(\{p, q\})|$.

Induktionsanfang: $|S \cap R(\{p, q\})| = 2$. In diesem Fall ist $R(\{p, q\})$ ein kritisches Rechteck und M enthält nach Definition einen Manhattan-Pfad für das Paar (p, q).

Induktionsschritt: Sei nun $|S \cap R(\{p, q\})| = n > 2$. Damit enthält das Rechteck $R(\{p, q\})$ mindestens einen weiteren Punkt r. Betrachte nun die Rechtecke $R(\{p, r\})$ und $R(\{r, q\})$. Da $R(\{p, r\})$ den Punkt q nicht enthalten kann und $R(\{r, q\})$ den Punkt p nicht enthalten kann, gilt für beide Rechtecke: $|S \cap R(\{p, r\})| < n$ und $|S \cap R(\{r, q\})| < 2$. Damit enthält M nach Induktionsvoraussetzung einen Pfad von p nach r und einen Pfad von r nach q. Diese kombiniert ergeben dann einen Pfad von p nach q.

□

Interessant ist, dass die Paare (p, q), die auf dem Rand einer Kontur benachbart sind, nicht zwangsläufig in einer erzeugenden Menge enthalten sein müssen. Im Regelfall sind sie enthalten, es gibt jedoch Punktmengen und erzeugende Mengen zu diesen Punktmenge, wo nicht alle Paare auf der Kontur nötig sind (vgl. Abbildung 2.8).

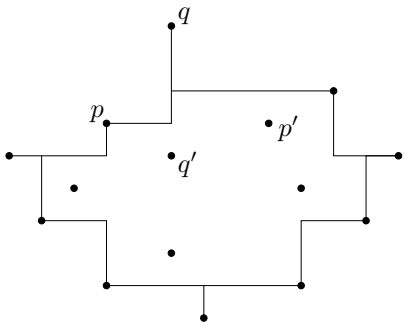


Abbildung 2.8: Das Paar (p, q) ist nicht nötig für eine erzeugende Menge, sofern das Paar (p, p') und (q, q') enthalten ist.

3 Approximations-Algorithmen

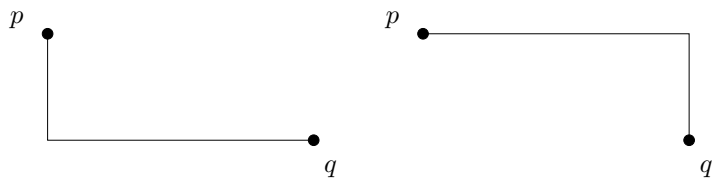
In diesem Kapitel wird ein Überblick über die bisher veröffentlichten Approximations-Algorithmen gegeben.

3.1 Eine 4-Approximation

Der hier vorgestellte Algorithmus stammt aus dem Paper „Approximating Minimum Manhattan Networks“ von Gudmundsson, Levcopoulos und Narasimhan [GLN01]. Dieser Algorithmus führt das Problem minimaler Manhattan-Netzwerke zurück auf das Problem einer minimalen Rectangulation einer Menge von Treppen-Polygonen. Wenn der Rectangulation Algorithmus in Zeit $O(R(n))$ läuft und einen Approximationsfaktor von r für die Rectangulation liefert, so wird der präsentierte Algorithmus ein Netzwerk berechnen, welches maximal um den Faktor $4r$ schlechter, als das minimale Manhattan-Netzwerk ist und in Zeit $O(n \log n + R(n))$ läuft. Unter Benutzung von zwei verschiedenen Approximations-Algorithmen für das Rectangulation Problem, erhalten wir dann zwei Approximations-Algorithmen für das minimale Manhattan-Netzwerk Problem. Der erste läuft in Zeit $O(n^3)$ und hat einen Approximationsfaktor von vier, während der zweite in Zeit $O(n \log n)$ läuft und einen Approximationsfaktor von acht hat.

3.1.1 Definitionen

Seien p und q Punkte einer Punktmenge S, wobei p links oben von q liegt. Ein $\llcorner$-Pfad zwischen p und q ist ein rechtwinkliger Pfad, der aus einem vertikalen Liniensegment mit Endpunkt in p und einem horizontalen Liniensegment mit Endpunkt in q besteht. Dabei handelt es sich natürlich um einen Manhattan-Pfad zwischen p und q. Der $\urcorner$-Pfad besteht dann aus dem horizontalen Liniensegment mit Endpunkt in p und dem vertikalen Liniensegment mit Endpunkt in q (vgl. Abbildung 3.1). Wenn p links unten von q liegt, sei der $\lrcorner$-Pfad und der $\ulcorner$-Pfad analog definiert.



(a) Ein $\llcorner$-Pfad (b) Ein $\urcorner$-Pfad

Abbildung 3.1: Zwei verschiedene, kanonische Pfade zwischen den Punkten p und q

Definition 3.1 (Rectangulation [LÖ96])
Eine Rectangulation von einem rechtwinkligen Polygon P ist eine Menge von Liniensegmenten, die P in Rechtecke unterteilt (vgl. Abbildung 3.2). Eine optimale Rectangulation ist eine Menge von Liniensegmenten mit minimaler Gesamtlänge, die P in Rechtecke unterteilt. Dabei ist der Rand des Polygons nicht enthalten.

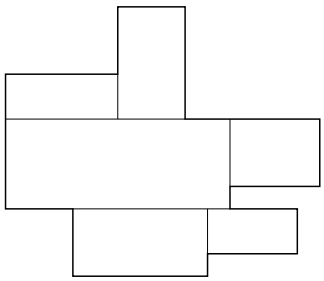


Abbildung 3.2: Rectangulation eines Polygons

3.1.2 Der Approximations-Algorithmus

Der Algorithmus berechnet die Segmente für das Manhattan-Netzwerk in vier verschiedenen, unabhängigen Schritten. In jedem Schritt und für jeden Punkt verbindet der Algorithmus diesen Punkt mit einer – möglicherweise leeren – Menge von bestimmten, benachbarten Punkten. Diese vier Schritte sind bis auf Symmetrie gleich, daher wird der erste Schritt ausführlicher erklärt, als die anderen drei Schritte.

1. Gehe mit einer Sweep-Line von links nach rechts durch die Punktmenge. Für jeden Punkt $p' \in S$ sei $p \in S$ der Punkt, der unterhalb und nicht links von p' liegt, sowie minimalen x-Abstand von p' hat. Sollte es mehrere Kandidaten geben, so ist p derjenige, der zusätzlich den minimalen y-Abstand von p' hat. Wir sagen dann im Folgenden, dass p' zu p *1-gehört*. Es ist offensichtlich, dass das Rechteck $R(\{p, p'\})$ ein kritisches Rechteck ist. Für jeden Punkt p sei $B_1(p)$ die Menge der Punkte von S, die zu p 1-gehören. Obwohl jeder Punkt p' zu höchstens einem Punkt p 1-gehören, kann es mehrere Punkte geben, die zu einem p 1-gehören. Seien $p_1, \ldots, p_m$ die Punkte, die zu p 1-gehören, sortiert nach ihren x-Koordinaten. Da jedes Rechteck $R(\{p_i, p\})$ ein kritisches Rechteck ist und alle p_i links oberhalb von p liegen, bilden diese Punkte eine Treppe (vgl. Abbildung 3.3). Als Erstes wird der $\neg$-Pfad e_1 zwischen p_1 und p eingefügt. Sollte p_1 die gleiche y-Koordinate wie p haben, so handelt es sich um ein einfaches Liniensegment. Falls $m > 1$ gilt, so wird eine senkrechte Kante e_2 von p_m mit Endpunkt auf e_1 eingefügt. Nun werden „lokale“ Manhattan-Netzwerke konstruiert, so dass von jedem Punkt $p_1, \ldots, p_m$ ein Manhattan-Pfad über e_1 oder e_2 zu p existiert. Dieser Schritt wird später noch genauer beschrieben. Alle Kanten, die in diesem Schritt erzeugt werden, werden der Menge N_1 hinzugefügt.

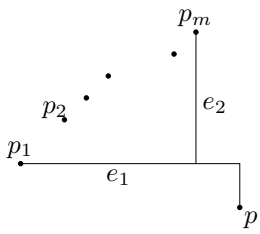


Abbildung 3.3: Die Punkte $p_1, \ldots, p_m$ 1-gehören zu p

2. Gehe mit einer Sweep-Line von links nach rechts durch die Punktmenge. Für jeden Punkt $p' \in S$ sei $p \in S$ der Punkt, der oberhalb, und nicht links von p' liegt, sowie minimalen x-Abstand von p' hat. Sollte es mehrere Kandidaten geben, so ist p derjenige, der zusätzlich den minimalen y-Abstand von p' hat. Wir sagen im Folgenden, dass p' zu p 2-gehört. Für jeden Punkt p sei $B_2(p)$ die Menge der Punkte von S, die zu p 2-gehören. Erzeuge dann, wie im ersten Schritt, eine Menge von Kanten N_2.

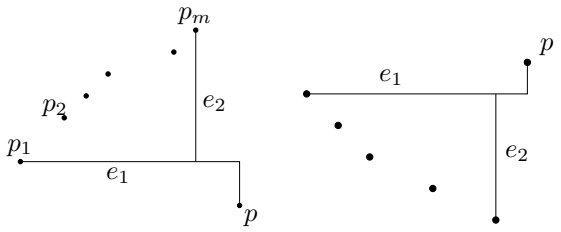
3. Gehe mit einer Sweep-Line von unten nach oben durch die Punktmenge. Für jeden Punkt $p' \in S$ sei $p \in S$ der Punkt, der links, und nicht unterhalb von p' liegt, sowie minimalen y-Abstand von p' hat. Sollte es mehrere Kandidaten geben, so ist p derjenige, der zusätzlich den maximalen x-Abstand von p' hat. Wir sagen im Folgenden, dass p' zu p 3-gehört. Für jeden Punkt p sei $B_3(p)$ die Menge der Punkte von S, die zu p 3-gehören. Erzeuge dann, wie im ersten Schritt, eine Menge von Kanten N_3.

4. Gehe mit einer Sweep-Line von oben nach unten durch die Punktmenge. Für jeden Punkt $p' \in S$ sei $p \in S$ der Punkt, der links, und nicht oberhalb von p' liegt, sowie minimalen y-Abstand von p' hat. Sollte es mehrere Kandidaten geben, so ist p derjenige, der zusätzlich den maximalen x-Abstand von p' hat. Wir sagen im Folgenden, dass p' zu p 4-gehört. Für jeden Punkt p sei $B_4(p)$ die Menge der Punkte von S, die zu p 4-gehören. Erzeuge dann, wie im ersten Schritt, eine Menge von Kanten N_4.

Nachdem geeignete „lokale“ Netzwerke erzeugt wurden, ist jeder Punkt $p \in S$ mit den Punkten aus $B_1(p) \cup B_2(p) \cup B_3(p) \cup B_4(p)$ direkt verbunden. Diese vier Sweeps erzeugen vier Netzwerke $N_1, \ldots, N_4$. Das Manhatten-Netzwerk N ist dann definiert als $N = N_1 \cup N_2 \cup N_3 \cup N_4$.

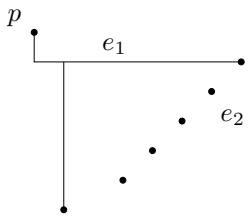
Jeder der vier Schritte ist symmetrisch zu jedem anderen Schritt (vgl. Abbildung 3.4). Durch diese vier Schritte ist bereits gewährleistet, dass Punkte mit gleichen x- oder y-Koordinaten miteinander verbunden sind.

Jeder dieser vier Sweep-Algorithmen hat die Laufzeit $O(n \log n)$. Diese ergibt sich aus folgender Überlegung: Bei jedem Ereignis des Sweep-Algorithmus muss der Punkt gesucht werden, zu dem er gehört. Entweder gibt es keinen solchen Punkt oder es gibt genau einen Punkt. Wenn die Punkte in einer geeigneten Datenstruktur, wie beispielsweise in einem AVL-Baum verwaltet werden, geht das Suchen des Punktes, zu dem der Ereignis-Punkt gehört, in Zeit $O(\log n)$. Es gibt genau n Ereignisse, da es n Punkte

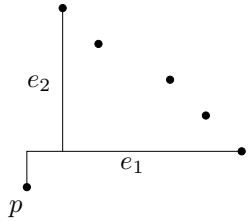


(a) Die Punkte, die zu p 1-gehören

(b) Die Punkte, die zu p 2-gehören



(c) Die Punkte, die zu p 3-gehören



(d) Die Punkte, die zu p 4-gehören

Abbildung 3.4: Die vier verschiedenen Mengen, die von den Sweeps erzeugt werden

gibt. Als Gesamtlaufzeit ergibt sich somit $O(n \log n)$ für jeden der vier verschiedenen Sweep-Algorithmen und auch für alle vier Sweep-Algorithmen zusammen.

Konstruktion der lokalen Netzwerke

Die Konstruktion der lokalen Netzwerke wird für den ersten Schritt genau beschrieben, für die anderen Schritte ist das Verfahren symmetrisch. Sei p ein Punkt aus S und seien $p_1, \ldots, p_m$ die Punkte in $B_1(p)$. Aus der Konstruktion im ersten Schritt ergibt sich, dass p_i links unten von p_{i+1} liegen muss, für $1 \leq i < m$, und dass $x_p \geq x_{p_m}$ und $y_p \leq y_{p_1}$ gelten muss.

Wir konstruieren nun ein lokales Netzwerk, das für jeden Punkt $p_i \in B_1(p)$ einen Manhattan-Pfad zwischen p und p_i enthält. Wir nehmen an, dass $m > 2$ gilt, ansonsten sind wir bereits fertig, da p_1 über die Kante e_1 mit p verbunden ist und p_m über die Kanten e_2 und e_1. Sei e_1' der horizontale Teil von e_1 zwischen p_1 und e_2. Betrachten wir nun das Treppenpolygon, das entsteht, wenn man zwischen p_i und $p_{i+1}, 1 \leq i < m$ je einen $\ulcorner$-Pfad hinzufügt und die Kanten e_1' und e_2 hinzunimmt (vgl. Abbildung 3.5). Dieses Polygon wird im Folgenden die *C-Hülle* von $B_1(p)$ genannt. Die Behauptung ist nun, dass eine Rectangulation dieses Polygons ein Netzwerk liefert, das einen Pfad von jedem $p_i \in B_1(p)$ nach p enthält. Dies ist offensichtlich, da jedes p_i auf einer Ecke eines anderen Rechtecks liegen muss. Somit gibt es immer einen Weg nach unten oder nach rechts vom Punkt aus. Weiterhin muss es einen monotonen Pfad entlang der Kanten der Rechtecke von p_i nach p geben. Sollte es keinen solchen Pfad geben, so müsste es eine Ecke geben, von der es nur einen Pfad nach links und nach oben gibt. Damit würde aber die Fläche, die sich rechts und unterhalb dieses Pfades befindet, einem nicht rechteckigen

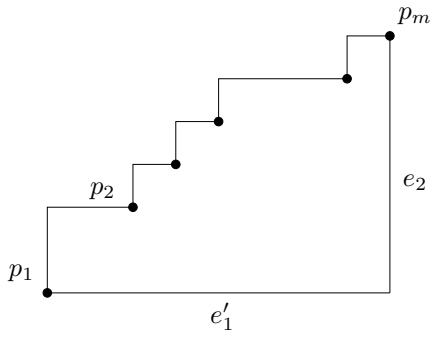


Abbildung 3.5: C-Hülle von $B_1(p) = \{p_1, \ldots p_m\}$

Polygon entsprechen. Dies kann jedoch nicht der Fall sein, da es sich um eine Rectangulation handelt. Sei nun $N_1(p)$ die Menge der Kanten, die bei der Rectangulation erzeugt wird vereinigt mit den Kanten e_1 und e_2. Die $\ulcorner$-Pfade von p_i nach p_{i+1} sind nach Definition nicht in $N_1(p)$ enthalten. Später zeigen wir, dass die minimale Rectangulation eine Abschätzung für die Länge des minimalen Netzwerkes erlaubt. Die folgenden Resultate für minimale Rectangulations sind bereits bekannt:

Theorem 3.2 ([LPRS82]) *Eine optimale Rectangulation von einem Treppenpolygon kann in Zeit $O(n^3)$ berechnet werden.*

Theorem 3.3 ([LÖ96]) *Eine so genannte „thickest-first“ Rectangulation eines Treppenpolygons kann in Zeit $O(n)$ berechnet werden, so dass die Gesamtlänge maximal doppelt so groß wie die Länge einer optimalen Rectangulation ist.*

Für die Gesamtlaufzeit reicht es wieder einen Fall zu betrachten. Die Gesamtlaufzeit ist dann um den konstanten Faktor vier größer und fällt damit in der O-Notation nicht in das Gewicht. Da der Sweep in Zeit $O(n \log n)$ läuft, braucht der Approximations-Algorithmus Zeit $O(n^3)$, wenn die optimale Rectangulation benutzt wird, und Zeit $O(n \log n)$, wenn der „thickest-first“ Algorithmus benutzt wird.

Der Algorithmus berechnet ein Manhattan-Netzwerk

Um zu zeigen, dass der Algorithmus (vgl. Algorithmus 3.1) ein Manhattan-Netzwerk berechnet, reicht es, das folgende Lemma zu zeigen:

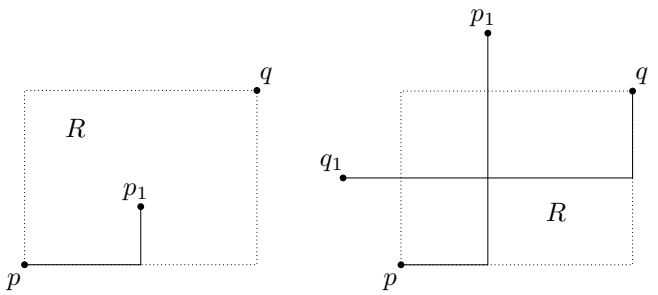
Lemma 3.4 *Für jedes Punktpaar $p, q \in S$ gibt es einen Manhattan-Pfad in N, der p und q verbindet.*

Beweis Lemma 3.4 Ohne Beschränkung der Allgemeinheit können wir annehmen, dass p links von q liegt. Ansonsten vertausche p und q. Weiterhin können wir annehmen, dass keine zwei Punkte die gleichen x- oder y-Koordinaten haben, da der Algorithmus zwischen diesen Punkten automatisch eine Kante einfügt. Nun kann q unter- oder oberhalb von p liegen. Betrachte zunächst den Fall, dass q oberhalb von p liegt.

Algorithmus 3.1 Approximations-Algorithmus mittels Rectangulation

Approximations-Algorithmus mittels Rectangulation

$E'_1 = E'_2 = E'_3 = E'_4 = \emptyset$
Führe einen Sweep von links nach rechts durch
for all $p \in S$ **do**
  Speichere die Menge $B_1(p)$ von Punkten, die zu p 1-gehören
  Konstruiere einen $\urcorner$-Pfad e_1 von $p_1 \in B_1(p)$ nach p
  Konstruiere einen Pfad e_2 von $p_m \in B_1(p)$ nach e_1
  $E'_1 = E'_1 \cup \{e_1, e_2\} \cup$ Rectangulation$(p_2, \ldots, p_{m-1}, e_1, e_2)$
end for
Führe einen Sweep von links nach rechts durch
for all $p \in S$ **do**
  Speichere die Menge $B_2(p)$ von Punkten, die zu p 2-gehören
  Konstruiere einen $\lrcorner$-Pfad e_1 von $p_1 \in B_2(p)$ nach p
  Konstruiere einen Pfad e_2 von $p_m \in B_2(p)$ nach e_1
  $E'_2 = E'_2 \cup \{e_1, e_2\} \cup$ Rectangulation$(p_2, \ldots, p_{m-1}, e_1, e_2)$
end for
Führe einen Sweep von unten nach oben durch
for all $p \in S$ **do**
  Speichere die Menge $B_3(p)$ von Punkten, die zu p 3-gehören
  Konstruiere einen $\llcorner$-Pfad e_1 von $p_1 \in B_3(p)$ nach p
  Konstruiere einen Pfad e_2 von $p_m \in B_3(p)$ nach e_1
  $E'_3 = E'_3 \cup \{e_1, e_2\} \cup$ Rectangulation$(p_2, \ldots, p_{m-1}, e_1, e_2)$
end for
Führe einen Sweep von oben nach unten durch
for all $p \in S$ **do**
  Speichere die Menge $B_4(p)$ von Punkten, die zu p 4-gehören
  Konstruiere einen $\ulcorner$-Pfad e_1 von $p_1 \in B_4(p)$ nach p
  Konstruiere einen Pfad e_2 von $p_m \in B_4(p)$ nach e_1
  $E'_4 = E'_4 \cup \{e_1, e_2\} \cup$ Rectangulation$(p_2, \ldots, p_{m-1}, e_1, e_2)$
end for
$N = E'_1 \cup E'_2 \cup E'_3 \cup E'_4$



(a) Der Punkt p_1 liegt innerhalb des Rechtecks R. Da es einen Pfad von p nach p_1 gibt, ergibt sich mit einem Pfad von p_1 nach q auch ein Pfad von p nach q.

(b) Sowohl p_1 als auch q_1 liegen außerhalb von R, aber der Schnittpunkt der beiden Pfade muss in R liegen.

Abbildung 3.6: Die zwei Fälle für die Lage der Punkte beim Beweis, dass der Algorithmus ein Manhattan-Netzwerk berechnet

Sei nun $R = R(\{p, q\})$ das umschließende Rechteck. Wenn p zu q 2-gehört, also q der Punkt rechts oberhalb von p mit minimalen x-Abstand zu p ist, dann ist p direkt mit q verbunden und wir sind fertig. Analoges gilt, wenn q zu p 4-gehört, also p der Punkt links unterhalb von von q mit minimalen y-Abstand zu q ist. Sollte dies nicht der Fall sein, so muss es einen Punkt p_1 geben, der zu p 4-gehört und verhindert, dass q zu p 4-gehört. Damit muss dieser Punkt rechts oberhalb von p, aber links von q liegen. Analog muss es einen Punkt q_1 geben, der verhindert, dass p zu q 2-gehört. Damit muss q_1 links unterhalb von q, aber oberhalb von p geben, der zu q 2-gehört. Nun werden die folgenden zwei Fälle unterschieden:

p_1 oder q_1 liegt in R (vgl. Abbildung 3.6(a)): Ohne Beschränkung der Allgemeinheit können wir annehmen, dass $p_1 \in R$ gilt. Sei nun p_1 der neue Punkt p und fahre rekursiv fort. Dies ist möglich, da es einen Manhattan-Pfad von p nach p_1 gibt. In Kombination mit einem Manhattan-Pfad von p_1 nach q ergibt sich somit auch ein Pfad von p nach q. Da in jedem Schritt das betrachtete Rechteck R kleiner wird, kann dieser Fall nur endlich oft eintreten, so dass wir nach einer endlichen Anwendung dieses Falles fertig sind.

p_1 und q_1 liegen außerhalb von R (vgl. Abbildung 3.6(b)): Da p_1 links von q liegen muss, kann p_1 nur oberhalb von R liegen. Da außerdem q_1 oberhalb von p liegt, muss q_1 dann links von R liegen. Dann muss sich der Pfad von p nach p_1 mit dem Pfad von q nach q_1 in R schneiden und somit gibt es auch einen Pfad von p nach q.

Der Fall, dass q unterhalb von p liegt, geht analog unter Benutzung der Schritte 1 und 3 des Algorithmus.

□

Die Gesamtlänge des erzeugten Netzwerkes ist durch $4r \times |N_{\mathbf{opt}}|$ beschränkt

Für die Analyse reicht es wieder nur einen Sweep, zum Beispiel den ersten, zu betrachten. Der Approximationsfaktor für diesen Sweep-Algorithmus wird mit vier multipliziert und ist dann der Approximationsfaktor für den gesamten Algorithmus. Sei nun p ein Punkt aus S und $B_1(p) = \{p_1, \ldots, p_m\}$. Definiere nun $S' = B_1(p) \cup \{p\}$. Sei das Einflussgebiet von p definiert als (vgl. Abbildung 3.7):

$$C_1(p) := \bigcup_{i=1}^{m} R(\{p, p_i\})$$

Das Innere eines Gebietes $C_1(p)$ kann durch die Konstruktion keine Punkte aus S

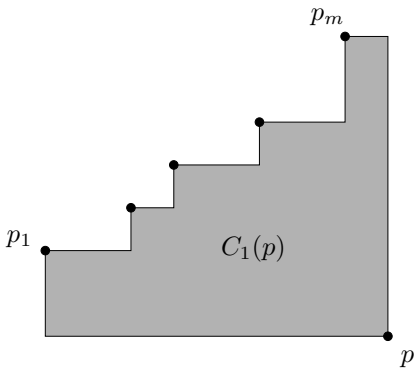


Abbildung 3.7: Das Einflussgebiet $C_1(p)$ des Punktes p

enthalten.

Lemma 3.5 *Für zwei Knoten $p, q \in S$ mit $p \neq q$ haben die Einflussgebiete $C_1(p)$ und $C_1(q)$ keine Punkte gemeinsam, außer möglicherweise p oder q.*

Beweis Lemma 3.5 Da jeder Punkt zu maximal einem Punkt 1-gehören kann, können die Treppenstufen von zwei Einflussgebieten keine Punkte gemeinsam haben. Somit kann höchstens p als Treppenstufe zu $C_1(q)$ gehören oder umgekehrt. Weiterhin enthalten die Einflussgebiete im Inneren keine Punkte aus S.

Angenommen zwei Einflussgebiete $C_1(p)$ und $C_1(q)$ mit $p \neq q$ überlappen sich. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass $x_p > x_q$ gilt. Weiterhin können wir ausschließen, dass q zu p 1-gehört, da sich sonst die Einflussgebiete nicht überlappen können. Sollte q nun überhalb von p liegen, dann muss in $R(\{p, q\})$ einen Punkt p' geben, der zu p 1-gehört, da sonst q zu p 1-gehört. Damit liegt jedoch q links oberhalb der Treppenstufe zu p' und deshalb können sich $C_1(p)$ und $C_1(q)$ in diesem Fall nicht überlappen. Damit bleibt nur noch der Fall, dass q unterhalb von p liegt übrig. Damit müssen jedoch alle Punkte, die zu p 1-gehören, rechts von q liegen, da sie sonst zu q 1-gehören. Damit können sich auch in diesem Fall die Einflussgebiete nicht überlappen.

□

Alle Kanten in $N_1(p)$, die vom Algorithmus hinzugefügt wurden, liegen in $C_1(p)$. Damit können die Kanten in $N_1(p)$ nicht genutzt werden, um Punkte in anderen Einflussgebieten zu verbinden. Wir zeigen nun, dass $|N_1(p)|$ maximal so groß ist, wie die Kanten von N_{opt}, die im Einflussgebiet $C_1(p)$ von p liegen. Als Erstes wird dazu das Einflussgebiet $C_1(p)$ in drei Gebiete aufgeteilt (vgl. Abbildung 3.8):

$$\begin{aligned} R_1 &= R(\{p, p_1\}) \\ R_2 &= R(\{p, p_m\}) \setminus R_1 \\ R_3 &= C_1(p) \setminus (R_1 \cup R_2) \end{aligned}$$

Dabei ist anzumerken, dass die Kanten e_1 und e_2 zwar auf ∂R_3 liegen, selbst jedoch

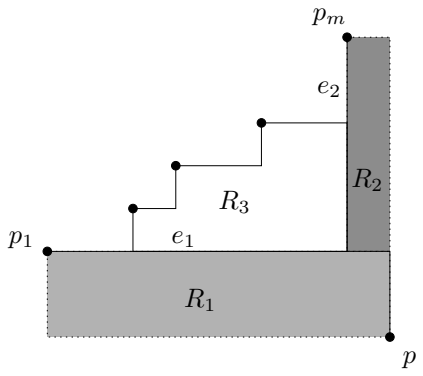


Abbildung 3.8: Partitionierung des Einflussgebietes $C_1(p)$

nicht zu R_3 gehören. Der Abschluss von R_3, der dann wieder die Kanten e_1 und e_2 enthält, entspricht der C-Hülle von von $B_1(p)$.

Betrachte man nun ein minimales Manhattan-Netzwerk N'_{opt}, das Manhattan-Pfade für die Paare aus $B_1(p) \times \{p\}$ enthält. Die Punkte aus $B_1(p)$ werden dabei nicht untereinander verbunden.

1. N'_{opt} muss einen Manhattan-Pfad zwischen p und p_1 in R_1 enthalten. Dieser Pfad muss die selbe Länge wie die Kante e_1 haben, die einen $\urcorner$-Pfad zwischen p und p_1 darstellt.

2. N'_{opt} muss einen Manhattan-Pfad zwischen p und p_m in $R(\{p, p_m\})$ enthalten. Das Stück dieses Pfades, das in R_2 liegt, muss mindestens so lang sein wie e_2.

3. Für $m > 2$ muss das Netzwerk N_{opt} die Punkte $p_2, \ldots, p_{m-1}$ entweder mit e_1 oder e_2 innerhalb R_3 verbinden, so dass sich ein Manhattan-Pfad von diesen Punkten zum Punkt p ergibt. Für diesen Fall muss nun nur gezeigt werden, dass ein minimales Netzwerk in R_3 mindestens so groß ist, wie eine minimale Rectangulation.

Aus Lemma 2.12 auf Seite 20 wissen wir, dass es ein minimales Manhattan-Netzwerk gibt, das auf dem durch die Punkte S erzeugten Gitter verläuft. Damit lässt sich folgendes Lemma zeigen:

Lemma 3.6 *Es gibt eine minimale Rectangulation der C-Hülle, die maximal so groß ist, wie ein minimales Manhattan-Netzwerk für die Paare $B_1(p) \times \{p\}$ innerhalb dieser Hülle.*

Beweis Lemma 3.6 Betrachte ein minimales Manhattan-Netzwerk N'_{opt}, das die Punkte aus $B_1(p)$ mit p verbindet. Nach Lemma 2.12 gibt es so ein Netzwerk, das nur auf dem, durch die Punkte $B_1(p) \cup \{p\}$ erzeugten, Gitter verläuft. Da wir uns jetzt nur auf eine Teilmenge der Pfade beschränken, gilt damit für ein minimales Netzwerk N_{opt} für die gesamte Punktmenge S:

$$|N'_{\text{opt}} \cap R_3| \leq |N_{\text{opt}} \cap R_3|$$

Jeder Pfad zwischen einem inneren Punkt p_i und p bewegt sich, von p_i aus gesehen, nur nach rechts und unten. Die einzige Möglichkeit, dass ein Pfad ein Netzwerk erzeugen würde, das keiner Rectangulation entspricht, ist, wenn er nach unten oder rechts abknickt, ohne auf einen anderen Pfad zu treffen. Angenommen, es gibt so einen Pfad. Betrachte dann die letzte Ecke, die der Pfad gemacht hat, bevor er einen anderen Pfad getroffen hat. Ohne Beschränkung der Allgemeinheit sei dies ein Pfad, der am Ende senkrecht nach unten verläuft und dann nach rechts abknickt, ohne einen horizontalen Pfad zu treffen, aber als nächstes einen vertikalen Pfad trifft. Wenn man nun die letzte Ecke weglässt und stattdessen von der vorherigen Ecke oder Kreuzung direkt nach rechts geht, so trifft man ebenfalls diesen vertikalen Pfad oder einen früheren (vgl. Abbildung 3.9). Damit ist der Pfad um das Stück, das er vorher nach unten gegangen ist, kürzer geworden, was ein Widerspruch zur Annahme ist, dass das Netzwerk ein optimales Netzwerk war.

□

Zusammen ergibt sich nun folgendes Lemma:

Lemma 3.7 *Sei r der Approximationsfaktor, der bei der Rectangulation erreicht wird. Dann gilt:*

$$\left(\sum_{p \in S} |N_1(p)| \right) \leq r|N_{opt}|$$

Beweis Lemma 3.7 Da die Einflussgebiete disjunkt sind, können sie getrennt betrachtet werden. Für jedes Einflussgebiet $C_1(p)$ gilt:

$$\begin{aligned} |e_1| &\leq |N_{\text{opt}} \cap R_1| \\ |e_2| &\leq |N_{\text{opt}} \cap R_2| \\ |\text{Rectangulation}(R_3)| &\leq r|N_{\text{opt}} \cap R_3| \end{aligned}$$

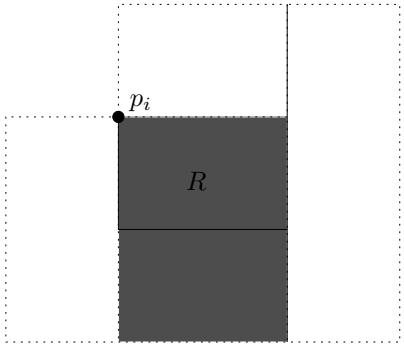


Abbildung 3.9: Der Pfad von p_i verläuft im Inneren des Rechtecks R der Rectangulation und trifft auf den vertikalen Pfad, der am rechten Rand von R entlangläuft. Dann ist es günstiger, direkt von p_i nach rechts zu gehen, bis man auf diesen Pfad trifft.

Aufsummiert über alle Einflussgebiete ergibt sich dann die gewünschte Formel:

$$\left(\sum_{p \in S} |N_1(p)|\right) \leq r|N_{\text{opt}}|$$

□

Um den Approximationsfaktor für den gesamten Algorithmus zu erreichen, muss der Approximationsfaktor für den einzelnen Sweep mit vier multipliziert werden, da es vier Sweeps gibt. Damit ergibt sich für den hier vorgestellten Algorithmus folgendes Theorem:

Theorem 3.8 *Gegeben sei eine Menge S von n Punkten in der Ebene und ein Algorithmus mit der Laufzeit $O(R(n))$ zur minimalen Rectangulation eines Treppenpolygons mit dem Approximationsfaktor r. Dann existiert ein Algorithmus mit der Laufzeit $O(n \log n + R(n))$, der ein Manhattan-Netzwerk berechnet, das maximal um den Faktor $4r$ schlechter ist, als ein minimales Manhattan-Netzwerk.*

Damit ergibt sich mit den zwei, auf Seite 25 vorgestellten, Resultaten für eine minimale Rectangulation folgendes Korollar:

Korollar 3.9 *Man kann in Zeit $O(n^3)$ eine 4-Approximation und in Zeit $O(n \log n)$ eine 8-Approximation eines minimalen Manhattan-Netzwerkes zu einer Punktmenge S berechnen.*

3.2 Eine schnelle 3-Approximation

Der folgende Algorithmus ist aus dem Paper „The Minimum Manhattan Network Problem: A Fast-Factor-3 Approximation“ von Benkert, Widmann und Wolff [BWW04] entnommen.

3.2.1 Definitionen für den Algorithmus

Gegeben ist eine Punktmenge S, für die ein Manhattan-Netzwerk berechnet werden soll. Dieser Algorithmus benutzt eine spezielle, erzeugende Menge Z. Diese ist dabei die Vereinigung von drei disjunkten Untermengen:

$Z_{\mathbf{ver}}$ Diese Menge enthält die vertikalen Paare. Sei die Menge $S = \{p_1, \ldots, p_n\}$ der Punkte lexikographisch sortiert. Weiterhin seien $x_1, \ldots, x_u$ die verschiedenen x-Koordinaten der Punkte aus der Punktmenge. Für $i = 1, \ldots, u$ ist die Menge $S_i = \{p_{a(i)}, p_{a(i)+1}, \ldots, p_{b(i)}\}$ die Menge der Punkte p aus S, die die x-Koordinate x_i haben. Diese Menge ist dann nach y-Koordinaten aufsteigend sortiert. Dann ist Z_{ver} definiert als:

$$\begin{aligned} Z_{\text{ver}} \quad := \quad & \left\{(p_j, p_{j+1}) \,|x_{p_j} = x_{p_{j+1}}\right\} \\ \cup \quad & \left\{\left(p_{a(i)}, p_{b(i+1)}\right) |y_{p_{a(i)}} > y_{p_{b(i+1)}} \wedge 1 \leq i \leq u\right\} \\ \cup \quad & \left\{\left(p_{b(i)}, p_{a(i+1)}\right) |y_{p_{b(i)}} < y_{p_{a(i+1)}} \wedge 1 \leq i \leq u\right\} \end{aligned}$$

Dies heißt, die Menge Z_{ver} (vgl. Abbildung 3.10(a)) besteht aus folgenden Punktpaaren:

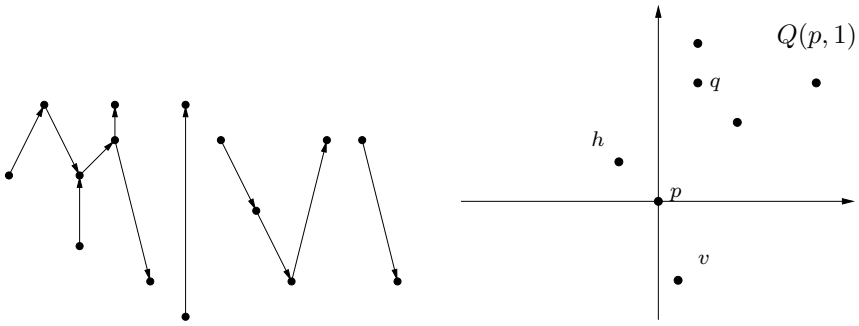
- Punktpaare, bei denen die beiden Punkte die gleichen x-Koordinaten haben. Dabei gilt für jedes dieser Paare (p, q): $y_p < y_q$ und auf dem Liniensegment zwischen p und q liegt kein weiterer Punkt.
- Von der Menge S_i wird der Punkt $p_{a(i)}$ mit der kleinsten y-Koordinate genau dann mit einem Punkt rechts verbunden, wenn der Punkt mit der größten y-Koordinate von der Menge S_{i+1} eine kleinere y-Koordinate hat (wenn S_{i+1} ganz unterhalb von S_i liegt).
- Von der Menge S_i wird der Punkt $p_{b(i)}$ mit der größten y-Koordinate genau dann mit einem Punkt rechts verbunden, wenn der Punkt mit der kleinsten y-Koordinate aus S_{i+1} eine größere y-Koordinate hat (wenn S_{i+1} ganz oberhalb von S_i liegt).

Z_{ver} besteht dabei aus maximal $n - 1$ Paaren.

Sollte es keine Punkte mit gleichen x-Koordinaten geben, so besteht Z_{ver} aus den Paaren der Punkte mit aufeinanderfolgenden x-Koordinaten und hat dann die Form $Z_{\text{ver}} = \{(p_i, p_{i+1}) | 1 \leq i < n\}$.

$Z_{\mathbf{hor}}$ Diese Menge ist analog zu Z_{ver} definiert, indem in der Definition x und y getauscht wird.

$Z_{\mathbf{quad}}$ Diese Menge enthält die fehlenden Paare, die nicht von Z_{ver} oder Z_{hor} abgedeckt werden (vgl. Abbildung 3.10(b)). Diese wird aufbauend auf den vier Quadranten (vgl. Definition 1.3 auf Seite 9) definiert. Z_{quad} besteht aus Paaren (p, q) mit $p, q \in S$, und $q \in Q(p, t)$ für $t \in \{1, 2, 3, 4\}$, für die gilt:



(a) Punktpaare der Menge Z_{ver}

(b) Das Punktpaar (p,q) ist in Z_{quad}, da $(h,p) \in Z_{\text{hor}}$ und $(p,v) \in Z_{\text{ver}}$ gilt und p somit keinen Partner in $Q(p,1)$ hat, so dass das Paar in $Z_{\text{ver}} \cup Z_{\text{hor}}$ wäre.

Abbildung 3.10: Die Mengen Z_{ver} und Z_{quad}

(a) Der Punkt q ist der Punkt in $Q(p,t) \cap S$ mit minimalem x-Abstand von p. Sollte es mehrere solcher Punkte geben, so sei S' die Menge der Punkte aus $Q(p,t) \cap S$ mit minimalem x-Abstand von p. Der Punkt q ist dann der Punkt aus S' mit minimalem y-Abstand von p.

(b) Es gibt kein $q' \in Q(p,t) \cap S$ mit (p,q') in $Z_{\text{hor}} \cup Z_{\text{ver}}$

Für jeden Punkt $p \in S$ kann es dabei pro $Q(p,t)$ maximal einen Partner geben, der diese Eigenschaft hat. Damit gibt es dann pro Punkt maximal vier Paare. Allerdings kann ein Punkt mehrmals als Partner eines anderen Punktes auftauchen.

Das folgende Lemma ist von Kato, Imai und Asano [KIA02], die dies in ihrem Paper erstmalig bewiesen haben. Dieser Beweis hier ist aber einfacher als der im Paper vorgestellte Beweis.

Lemma 3.10 *$Z = Z_{ver} \cup Z_{hor} \cup Z_{quad}$ ist eine erzeugende Menge.*

Beweis Lemma 3.10 Da nach Lemma 2.15 auf Seite 22 die Menge $Z_{\oslash}$ von kritischen Rechtecken eine erzeugende Menge ist, reicht es zu zeigen, dass durch Z für alle kritischen Rechtecke ein Manhattan-Pfad definiert wird. Wir führen nun einen Widerspruchs-Beweis.

Sei (p,q) ein Punktpaar, für das gilt: $(p,q) \notin Z$ und $R = R(\{p,q\})$ ist ein kritisches Rechteck. Weiterhin gebe es keinen Manhattan-Pfad von p nach q im Manhattan-Netzwerk zu der erzeugenden Menge Z. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass $x_p < x_q$ und $y_p < y_q$ gilt. Das Rechteck kann nicht degeneriert sein, da das Paar sonst sonst in $Z_{\text{ver}} \cup Z_{\text{hor}}$ enthalten wäre. Die anderen Lagen der Punkte sind symmetrisch (vgl. Abbildung 3.11). Sei nun S_1 der horizontale Streifen durch R, und sei S_2 der vertikale Streifen durch R. Nun gilt $(p,q) \notin Z_{\text{quad}}$. Damit muss p einen Partner p' in $Q(p,1)$ und q einen Partner q' in $Q(q,3)$ haben. p' muss im Streifen S_1 rechts von

R oder im Streifen S_2 oberhalb von R liegen. Analog muss q im Streifen S_1 links von R oder im Streifen S_2 unterhalb von R liegen. Sollten p' und q' in unterschiedlichen Streifen liegen, so schneiden sich die Pfade in R und somit gibt es einen Manhattan-Pfad von p nach q. Nehmen wir zunächst an, dass p' und q' in S_1 liegen. Damit gibt es automatisch einen Manhattan-Pfad von p nach q, wenn es einen vertikalen Pfad durch R gibt, der sowohl in x-, als auch in y-Richtung wächst. Wäre S_2 ganz leer, dann würde jedoch (p, q) in Z_{ver} liegen. Da dies nicht sein kann, muss es einen Punkt r in S_2 geben, der dies verhindert. Sei r der Partner von p in Z_{ver}. Wenn r in S_2 oberhalb von des Rechtecks R liegt, dann schneidet der Pfad von p nach r den Pfad von q nach q' und es gibt auch einen Pfad von p nach q. Also muss der Punkt r in S_2 unterhalb von R liegen. Analog muss der Partner r' von q in Z_{ver} in S_2 oberhalb von R liegen. Einer dieser Partner oder beide können nur dann nicht existieren, wenn es ein degeneriertes, vertikales Rechteck gibt, das R komplett schneidet. Durch dieses degenerierte Rechteck würde aber ebenfalls ein Manhattan-Pfad von p nach q definiert. Auf r und r' lassen sich nun die gleichen Argumente wie auf p und q anwenden. Wir können diese Rekursion ein paar mal durchlaufen. Da es nur eine endliche Anzahl von Punkten gibt, muss es schließlich einen x- und y-monoton steigenden Pfad durch R geben. Damit gibt es aber auch einen Manhattan-Pfad von p nach q. Dies ist jedoch ein Widerspruch zu der Annahme.

Sollten p' und q' in S_2 liegen, so lässt sich die gleiche Argumentation mit den Paaren aus Z_{hor} führen.

□

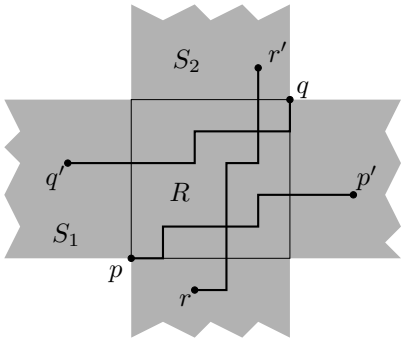


Abbildung 3.11: Beweis, dass Z eine erzeugende Menge ist. Die Pfade von p nach p' und q nach q' müssen existieren. Da auch noch ein vertikaler Pfad das Rechteck R schneidet, ergibt sich ein Manhattan-Pfad von p nach q.

Für die Gesamtzahl der Paare gilt:

$$|Z_{\text{hor}}| + |Z_{\text{ver}}| + |Z_{\text{quad}}| \leq n + n + 4n = 6n = O(n)$$

Weiterhin werden noch folgende Mengen von Rechtecken definiert:

$$\begin{aligned} R_{\text{hor}} &:= \{R(\{p,q\}) \mid (p,q) \in Z_{\text{hor}}\} \\ R_{\text{ver}} &:= \{R(\{p,q\}) \mid (p,q) \in Z_{\text{ver}}\} \\ R_{\text{quad}} &:= \{R(\{p,q\}) \mid (p,q) \in Z_{\text{quad}}\} \end{aligned}$$

Die Flächen, die diese Rechtecke einnehmen, werden jeweils mit $A_{\text{hor}}, A_{\text{ver}}$ und A_{quad} bezeichnet:

$$\begin{aligned} A_{\text{hor}} &= \bigcup_{R \in R_{\text{hor}}} R \\ A_{\text{ver}} &= \bigcup_{R \in R_{\text{ver}}} R \\ A_{\text{quad}} &= \bigcup_{R \in R_{\text{quad}}} R \end{aligned}$$

Definition 3.11 (Cover [KIA02]) *Ein Cover V von R_{ver} ist eine Menge von vertikalen Liniensegmenten. Diese haben die Eigenschaft, dass es für jede horizontale Gerade l und jedes Rechteck $R \in R_{ver}$ mit $l \cap R \neq \emptyset$ ein vertikales Liniensegment $k \in V$ gibt, so dass $l \cap R \cap k \neq \emptyset$ gilt.*

V ist ein minimales vertikales Cover (MVC), wenn gilt:

$$\forall V', V' \text{ ist ein Cover von } R_{ver} : |V| \leq |V'|$$

Das minimale, horizontale Cover (MHC) ist analog definiert.

Dies bedeutet, dass es für ein Rechteck $R \in R_{\text{ver}}$ Kanten im Cover V gibt, so dass jede horizontale Kante, die durch das ganze Rechteck R geht, von Kanten des Covers geschnitten wird. Damit wird das Rechteck auf der gesamten Höhe von Kanten „überdeckt“. Ein Beispiel eines solchen Covers findet sich in Abbildung 3.18 auf Seite 48.

Lemma 3.12 ([KIA02]) *Zu einer gegebenen Punktmenge S ist die Vereinigung eines minimalen vertikalen Covers mit einem minimalen horizontalen Cover maximal so groß wie ein minimales Manhattan-Netzwerk M:*

$$|MVC| + |MHC| \leq |M|$$

Beweis Lemma 3.12 Sei M ein beliebiges, minimales Manhattan-Netzwerk, dessen Pfade nach Lemma 2.12 auf Seite 20 entlang des durch S erzeugten Gitters verlaufen. Betrachte nun alle Paare $(p,q) \in Z_{\text{ver}}$. Das Netzwerk M muss einen Manhattan-Pfad von p nach q enthalten. Da der vertikale, offene Streifen zwischen p und q nach der Definition von Z_{ver} keine weiteren Punkte aus S enthalten kann, muss der Pfad das Rechteck $R(\{p,q\})$ vertikal abdecken. Analog muss der Pfad zwischen p' und q', für $(p',q') \in Z_{\text{hor}}$ das Rechteck $R(\{p',q'\})$ horizontal abdecken. Damit muss das minimale Netzwerk mindestens so groß sein, wie das minimale vertikale Cover vereinigt mit dem minimalen horizontalen Cover.

□

Der Algorithmus geht in vier Phasen vor. In Phase 0 wird Z berechnet. In Phase 1 wird ein Netzwerk N_1 berechnet, das die Vereinigung eines speziellen MVC und eines speziellen MVH enthält und ein Manhattan-Netzwerk für die Paare aus $Z_{\text{ver}} \cup Z_{\text{hor}}$ ist. Ein Manhattan-Netzwerk bezüglich Punktpaaren enthält für jedes Paar einen Manhattan-Pfad. In Phase 2 wird eine Menge $\mathcal{R}$ von offenen Regionen in A_{quad} identifiziert, die N_1 nicht schneiden, aber überbrückt werden müssen, um ein Manhattan-Netzwerk für die Paare aus Z_{quad} zu erhalten. Diese Regionen sind Treppenpolygone. Daraus ergeben sich zwei Mengen N_2 und N_3 von Segmenten, die ein Manhattan-Netzwerk für Z_{quad} ergeben. Für jede Region $R \in \mathcal{R}$ kommen die Segmente aus $\partial R \setminus \bigcup N_1$ nach N_2, sowie Segmente, um R mit N_1 zu verbinden. In der letzten Phase 3 wird das Innere der Regionen aus $\mathcal{R}$ durch die Berechnung der Menge N_3 von Segmenten im Inneren dieser Regionen überbrückt. Damit ergibt sich ein Manhattan-Netzwerk: $N = N_1 \cup N_2 \cup N_3$.

Die Neuheit dieser Analyse ist, dass die Ebene in zwei Gebiete aufgeteilt wird und N mit dem minimalen Manhattan-Netzwerk in jedem Gebiet separat verglichen wird. Das Gebiet A_3 enthält das Innere der Regionen $R \in \mathcal{R}$ und enthält somit N_3. Das andere Gebiet A_{12} ist das Komplement zu A_3 und enthält $N_1 \cup N_2$. Für ein fixes minimales Manhattan-Netzwerk N_{opt} zeigen wir:

$$\begin{aligned} |N \cap A_{12}| &\leq 3|N_{\text{opt}} \cap A_{12}| \\ |N \cap A_3| &\leq 2|N_{\text{opt}} \cap A_3| \end{aligned}$$

Daraus ergibt sich dann der kompetitive Faktor von drei:

$$|N| \leq 3|N_{\text{opt}}|$$

3.2.2 Nachbarn und die erzeugende Menge

Definition 3.13 (Nachbarn) *Für einen Punkt $p \in S$ und $t \in \{1,2,3,4\}$ sind die horizontalen Nachbarn definiert als:*

$$p.xnbor[t] := \begin{cases} nil & wenn & Q(p,t) \cap S = \{p\} \\ q & sonst,\ mit & |y_q - y_p| = \min\{|y_r - y_p|, r \in Q(p,t) \cap S \setminus \{p\}\} \\ & und & \nexists r \in S : y_r = y_q \wedge |x_q - x_p| > |x_r - x_p| \end{cases}$$

Der horizontale Nachbar von p im Quadranten $Q(p,t)$ ist der Punkt in dem Quadranten, der den geringsten y-Abstand von p hat. Sollten mehrere Kandidaten in Betracht kommen, so wird der mit dem minimalen x-Abstand genommen. Sollte es überhaupt keinen solchen Punkt in diesem Quadranten geben, so gibt es keinen Nachbarn (vgl. Abbildung 3.12).

Die Definition der vertikalen Nachbarn $p.ynbor[t]$ ist analog und ergibt sich durch vertauschen von x und y in der Definition.

Diese Nachbarn lassen sich in Zeit $O(n \log n)$ mittels mehrerer Sweep-Algorithmen berechnen. Für die Nachbarn vom Typ xnbor[2] und xnbor[3] reicht ein Sweep mit einer Geraden von links nach rechts. Bei jedem Ereignis mit einem Punkt p muss dessen y-Koordinate in der Menge der bisherigen Punkte gesucht werden. Der Punkt, der die

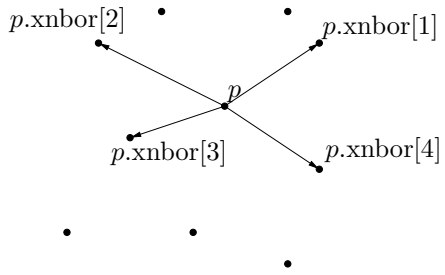


Abbildung 3.12: Horizontale Nachbarn von p

nächstgrößere y-Koordinate hat, ist dann p.xnbor[2], der Punkt, der die nächstkleinere y-Koordinate hat, ist dann p.xnbor[3]. Diese Lokalisierung kostet bei einer geeigneten Datenstruktur, wie beispielsweise einem AVL-Baum, nur Zeit $O(\log n)$, womit sich dann die Gesamtlaufzeit von $O(n \log n)$ ergibt. Analog lassen sich mittels einem Sweep von rechts nach links die Nachbarn vom Typ xnbor[1] und xnbor[4] berechnen. Die Nachbarn vom Typ ynbor lassen sich analog mittels vertikaler Sweeps berechnen.

Ein Punkt $p \in S$ heißt *vertikaler Vorgänger* von $q \in S$, wenn $(p, q) \in Z_{\text{ver}}$ gilt, oder vertikaler Nachfolger von $q \in S$, wenn $(q, p) \in Z_{\text{ver}}$ gilt. Ein Vorgänger oder Nachfolger heißt degeneriert, wenn er die gleiche x-Koordinate hat.

Nachdem die Nachbarn berechnet wurden, lassen sich daraus die erzeugenden Mengen $Z_{\text{ver}}, Z_{\text{hor}}$ und Z_{quad} berechnen. Z_{ver} lässt sich aus den vertikalen Nachbarn berechnen. Ein Punkt $p \in S$ kann entweder mit p.ynbor[1] oder p.ynbor[4] ein Rechteck aus Z_{ver} bilden, bei dem p auf der linken Seite sitzt. Wenn p.ynbor[1] die selbe x-Koordinate wie p hat, dann bilden p und p.ynbor[1] ein Punktpaar. Sollten p und p.ynbor[1] eine unterschiedliche x-Koordinate haben, so muss nur geschaut werden, ob und welche Punkte eine gleiche x-Koordinate wie p und p.ynbor[1] haben. Das lässt sich durch betrachten der Nachbarn p.ynbor[4], p.ynbor[1].ynbor[1] und p.ynbor[1].ynbor[4] feststellen. Dann lässt sich daran entscheiden, ob die Punkte p und p.ynbor[1] ein Paar bilden. Analoges gilt für p und p.ynbor[4].

Analog müssen für Z_{hor} die horizontalen Nachbarn betrachtet werden. Z_{quad} lässt sich berechnen, indem man die vier vertikalen Nachbarn betrachtet und schaut, ob der Punkt mit vertikalen oder horizontalen Nachbarn in diesem Quadranten ein Paar aus Z_{ver} oder Z_{hor} bildet. Dazu müssen jeweils maximal acht Paare betrachtet, und zwar p.xnbor[t] und p.ynbor[t] für $t \in \{1, 2, 3, 4\}$, da ein Punkt nur mit diesen Punkten ein Paar aus Z_{ver} oder Z_{hor} bilden kann. Damit ergibt sich dann folgendes Lemma:

Lemma 3.14 *Alle Zeiger vom Typ xnbor, ynbor und die erzeugende Menge Z lassen sich in Zeit $O(n \log n)$ berechnen.*

3.2.3 Minimale Cover

Normalerweise ist die Vereinigung eines minimalen, vertikalen Covers MVC und eines minimalen horizontalen Covers MHC kein Manhattan-Netzwerk für Paare aus $Z_{\text{ver}} \cup$

Z_{hor}. Es müssen meistens noch weitere Liniensegmente hinzugefügt werden. Um die Länge dieser Liniensegmente abschätzen zu können, benötigt man Cover mit besonderen Eigenschaften.

Definition 3.15 (Schönes Cover) *Ein minimales, horizontales oder vertikales Cover V zu einer Menge Z von Punktpaaren heißt schön, wenn alle Liniensegmente aus V mindestens einen Punkt aus Z enthalten:*

$$\forall k \in V : \exists\, (p,q) \in Z, p \in k \vee q \in k$$

Lemma 3.16 *Für ein schönes, minimales, vertikales Cover $\mathcal{V}$ und ein schönes, minimales, horizontales Cover $\mathcal{H}$ zu einer Menge S gibt es eine Menge L von Liniensegmenten, so dass $\mathcal{V} \cup \mathcal{H} \cup L$ einen Manhattan-Pfad für jedes Paar aus $Z_{ver} \cup Z_{hor}$ enthält und $|L| \leq W + H$, wobei W die Breite von $R(S)$ und H die Höhe von $R(S)$ ist. L lässt sich in linearer Zeit konstruieren, sofern in konstanter Zeit auf die Liniensegmente $k \in V$, die ein Rechteck $R \in R_{ver}$ schneiden, zugegriffen werden kann.*

Beweis Lemma 3.16 Es wird gezeigt, dass es eine Menge L_V von horizontalen Liniensegmenten gibt, mit $|L_V| \leq W$ und der Eigenschaft, dass $\mathcal{V} \cup L_V$ einen Manhattan-Pfad für jedes Punktpaar aus Z_{ver} enthält.

Sei $(p,q) \in Z_{\text{ver}}$. Wenn das umschließende Rechteck $R = R(\{p,q\})$ degeneriert ist, dann gibt es nach der Definition des Covers ein Liniensegment $l \in \mathcal{V}$ mit $R(\{p,q\}) \subseteq l$. Damit existiert auch ein Manhattan-Pfad in $\mathcal{V}$ für (p,q).

Sollte das Rechteck nicht degeneriert sein, gibt es einen offenen, vertikalen Streifen $\sigma(p,q)$, der von p und q begrenzt wird. Aufgrund der Definition von Z_{ver} ist R das einzige Rechteck von R_{ver}, das den Streifen $\sigma(p,q)$ schneidet. Damit gilt, dass sich die Breiten von $\sigma(p,q)$ über alle $(p,q) \in Z_{\text{ver}}$ maximal auf W aufsummieren.

Wenn es für jedes $\sigma(p,q)$ ein horizontales Liniensegment h gibt, mit $|h|$ entspricht der Breite von $\sigma(p,q)$, so dass $V \cup \{h\}$ einen Manhattan-Pfad für (p,q) enthält, dann ist die Behauptung gezeigt. Da $\mathcal{V}$ ein schönes MVC ist und $\sigma(p,q) \cap S = \emptyset$ gilt, schneidet kein Liniensegment aus $\mathcal{V}$ den Streifen (p,q). Die Segmente aus $\mathcal{V}$ schneiden das Rechteck $R(p,q)$ nur an den vertikalen Kanten des Rechtecks. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass $x_p < x_q$ und $y_p < y_q$ (Ansonsten kann man die Punktmenge umbenennen oder an der x-Achse spiegeln).

Aufgrund der Definition von Z_{ver} gibt es keinen Punkt vertikal über p. Wenn es ein Segment k_p in $\mathcal{V}$ gibt, das den linken Rand des Rechtecks R schneidet, dann muss k_p den Punkt p enthalten, da es sich um ein schönes MVC handelt. Analoges gilt für die rechte Seite von $R(\{p,q\})$: Wenn es dort ein Liniensegment k_q aus $\mathcal{V}$ gibt, muss dieses den Punkt q enthalten. Da $\mathcal{V}$ das Rechteck R abdeckt, muss k_p oder k_q existieren. Sei nun l die horizontale Gerade durch den obersten Punkt von k_p oder den untersten Punkt von k_q. Dann ist $h = l \cap R$ (vgl. Abbildung 3.13).

Damit ist gezeigt, dass es reicht, Liniensegmente L_v, die maximal die Gesamtlänge W haben, zu $\mathcal{V}$ hinzuzufügen, so dass $L_v \cup \mathcal{V}$ für jedes Paar aus Z_{ver} einen Manhattan-Pfad enthält. Analog kann man zeigen, dass es reicht, Liniensegmente L_h, die maximal die

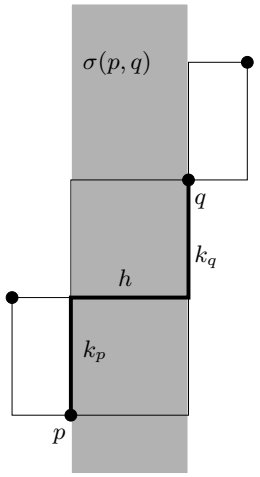


Abbildung 3.13: Es reicht genau ein Segment, und zwar das Segment h, zum Cover hinzuzufügen, um einen Manhattan-Pfad von p nach q zu erhalten.

Gesamtlänge H haben, zu $\mathcal{H}$ hinzuzufügen, so dass $L_h \cup \mathcal{H}$ für jedes Paar aus Z_{hor} einen Manhattan-Pfad enthält. Mit $L = L_v \cup L_h$ ist dann das Lemma bewiesen, denn die Aussage zur Laufzeit ist trivial.

□

Allerdings ist es nicht offensichtlich, ob es zu jeder Punktmenge S ein schönes, minimales Cover gibt. Im Folgenden wird dies für Z_{ver} gezeigt. Der Beweis geht für Z_{hor} analog.

Betrachte nun für eine beliebige, horizontale Gerade l den Graphen $G_l(V_l, E_l)$, der wie folgt definiert ist: Die Schnittpunkte der Geraden mit den Rechtecken aus R_{ver} bilden die Knoten V_l des Graphen. Eine Kante verbindet genau dann zwei Knoten des Graphen, wenn die geschnittenen Kanten zu demselben Rechteck gehören (vgl. Abbildung 3.14). Dann wird jede Zusammenhangskomponente getrennt betrachtet und jeder zweite Knoten in das Cover aufgenommen. Sollte eine Zusammenhangskomponente nur aus einem Knoten bestehen, wird dieser Knoten genommen. Diese Knoten werden als ungerade bezeichnet. Es wird dann gezeigt, dass das es auf jeder vertikalen Geraden, die den Rand eines oder mehrerer Rechtecke schneidet, maximal eine verbundene Menge von diesen ungeraden Punkten gibt, die dann das ungerade Segment bildet. Dieses ungerade Segment enthält, wenn es existiert, mindestens einen Punkt aus der Eingabemenge. Die Menge aller dieser ungeraden Liniensegmente ergibt dann ein schönes Cover. Durch dieses Abzählprinzip ist gewährleistet, dass von jedem Rechtecke eine der beide Seiten im schönen Cover enthalten ist, womit dieses schöne Cover auch tatsächlich ein Cover ist.

Weiterhin gibt es eine Abbildung $f : V_l \to \{\text{gerade}, \text{ungerade}\}$. Ein Knoten $v \in V_l$ ist ungerade, wenn er zu einem degenerierten Rechteck gehört, dass l schneidet. Sollte v zu einem nicht degenerierten Rechteck gehören, so ist v genau dann ungerade, wenn die

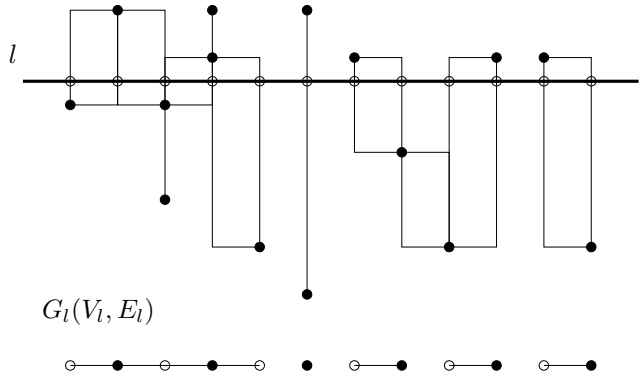


Abbildung 3.14: Oben sieht man den Schnitt der Geraden l mit den Rechtecken aus R_{ver}. Unten ist der entstehende Graph $G_l(V_l, E_l)$ dargestellt, wobei die Schnittpunkte den Knoten entsprechen und Kanten zwischen den Knoten genau dann vorhanden sind, wenn die Knoten zum selben Rechteck gehören. Die ausgefüllten Knoten des unteren Graphen sind dabei die ungeraden Knoten, die nicht gefüllten entsprechen den geraden Knoten.

Anzahl der Knoten links von v, die zu der selben Zusammenhangskomponente gehören, ungerade ist. Ansonsten ist v gerade.

Die folgende Idee ist nun, mit einer Sweep-Line von unten nach oben sich durch die Menge bewegt und dabei die ungeraden Segmente notiert. Das Ergebnis findet sich beispielsweise in Abbildung 3.18 auf Seite 48.

Für eine vertikale Gerade durch einen Punkt aus S ist ein *gerades Liniensegment* eine inklusions-maximale Menge von geraden, verbundenen Punkten. Die ungeraden Liniensegmente sind analog definiert. Die Punkte, an denen sich ein gerades und ein ungerades Liniensegment treffen, sind *Punkte von wechselnder Parität*. Die Menge der ungeraden Liniensegmente ergibt das schöne, minimale, vertikale Cover. Im folgenden Lemma werden die ungeraden Liniensegmente charakterisiert. Da abgeschlossene Liniensegmente die gleiche Länge haben wie offene Liniensegmente, können wir die ungeraden Liniensegmente als abgeschlossen definieren.

Lemma 3.17 *Sei g eine vertikale Gerade mit der x-Koordinate x_g durch einen Punkt $p \in S$.*

(i) Sei e eine vertikale Kante eines Rechtecks $R \in R_{ver}$. Dann sind entweder alle Punkte von e gerade oder es gibt nur inklusions-maximale Menge von verbundenen, ungeraden Punkten auf e und diese enthält einen Punkt aus S.

Einfacher formuliert: Dies bedeutet, wenn es gibt maximal ein ungerades Segment auf einer vertikalen Kante eines Rechtecks und wenn es dieses Segment gibt, enthält es einen Punkt aus S.

(ii) *Seien $R_1, \ldots, R_d$ die degenerierten Rechtecke und $R_{1'}, \ldots R_{d'}$ die nicht degenerierten Rechtecke von R_{ver}, die g schneiden. Dann ist $d = |g \cap S| - 1$ und $d' \leq 2$. Wenn $d = 0$ gilt, dann ist $d' > 0$ und jedes R'_i hat p als Ecke. Wenn $d > 0$ gilt, dann gibt es $p_1, p_2 \in S$, so dass $g \cap (R_1 \cup \cdots \cup R_d)$ dem Liniensegment zwischen p_1 und p_2 entspricht und jedes R'_i hat den Punkt p_1 oder p_2 als Eckpunkt.*

Einfacher formuliert: Die degenerierten Rechtecke auf einer vertikalen Geraden bilden ein zusammenhängendes Liniensegment mit Endpunkten in S. Die nicht degenierten Rechtecke können sich dann nur an die Endpunkte dieses Liniensegmentes anschließen.

(iii) *$A_{ver} \cap g$ ist ein zusammenhängendes Liniensegment. Dabei ist $b_g \in \mathbb{R}$ der untere und $t_g \in R$ der obere Endpunkt des Liniensegmentes.*

(iv) *Die Gerade g enthält höchstens zwei Punkte von wechselnder Parität und höchstens ein ungerades Segment. Für jeden Punkt c von wechselnder Parität gibt es einen Punkt aus S mit der gleichen y-Koordinate.*

Einfacher formuliert: Die vertikale Gerade enthält insgesamt nur maximal ein ungerades Liniensegment. Der Anfang und Ende dieses Liniensegmentes muss die gleiche y-Koordinate, wie ein Punkt aus der Eingabemenge haben.

(v) *Wenn g keinen Punkt wechselnder Parität hat, dann gibt es entweder kein ungerades Liniensegment auf g oder das ungerade Liniensegment ist $\{x_g\} \times [b_g, t_g]$.*

Wenn g einen Punkt c mit wechselnder Parität hat, dann ist entweder $\{x_g\} \times [b_g, y_c]$ oder $\{x_g\} \times [y_c, t_g]$ das ungerade Liniensegment.

Wenn g zwei Punkte c und c' mit wechselnder Parität hat, dann ist $\{x_g\} \times [y_c, y_{c'}]$ das ungerade Liniensegment.

Einfacher formuliert: Vor und/oder hinter dem ungeraden Liniensegment kann sich genau ein gerades Segment befinden.

Beweis Lemma 3.17 Für (i) können wir ohne Beschränkung der Allgemeinheit annehmen, dass e die rechte Kante eines Rechtecks $R = R(\{p, q\})$ ist, und dass q der obere Punkt von e ist. Die anderen Fälle sind symmetrisch. Wenn R ein degeneriertes Rechteck ist, dann sind nach der Definition von ungeraden Liniensegmenten alle Punkte auf e, inklusive p und q, ungerade. Deshalb können wir annehmen, dass $x_p < x_q$ gilt. Betrachte nun die Rechtecke aus R_{ver}, die für die Parität von e wichtig sind. Seien $p_0 = q, p_1 = p, p_2, \ldots, p_k$ die Punkte aus S mit absteigenden x-Koordinaten, die diese aufspannen. Ein Rechteck ist R' nur dann wichtig, wenn es eine horizontale Gerade gibt, die einen Punkt von auf der Kante e schneidet und die Punkte von R', die von dieser Geraden geschnitten werden, zu derselben Zusammenhangskomponente gehören. Für $2 \leq i \leq k$ ist $\overline{y_{p_i}}$ rekursiv definiert als:

$$\overline{y_{p_i}} := \begin{cases} \min\left\{y_{p_i}, \overline{y_{p_{i-2}}}\right\} & \text{für } i \text{ gerade} \\ \max\left\{y_{p_i}, \overline{y_{p_{i-2}}}\right\} & \text{für } i \text{ ungerade} \end{cases}$$

Darauf aufbauend ist $\overline{p_i}$ definiert als:

$$\overline{p_i} := (x_{p_i}, \overline{y_{p_i}})$$

Sei $\overline{\mathcal{L}}$ definiert als die polygonale Kette durch:

$$\overline{\mathcal{L}} := p_0, p_1, \overline{p_2}, \overline{p_3}, \ldots \overline{p_k}$$

Aufgrund der Definition von $\overline{p_i}$ verläuft $\overline{\mathcal{L}}$ in dem horizontalen Streifen zwischen p_0 und p_1. Ihre vertikale Ausdehnung kann dabei, von p_0 aus gesehen, nur abnehmen. Aufgrund der Tatsache, dass alle Rechtecke, die diese Punkte definieren, wichtig für die Parität von e sind, muss es auch immer abwechselnd ein aufsteigendes und ein absteigendes Segment geben. Falls zwei absteigende oder aufsteigende Segmente aufeinanderfolgen würden, wäre das linke der beiden Rechtecke kein wichtiges Rechteck (vgl. Abbildung 3.15).

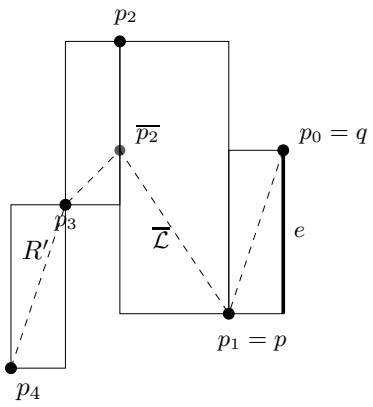


Abbildung 3.15: Unmöglicher Fall die Polygonale Kette $\overline{\mathcal{L}}$. Das Rechteck R' kann kein wichtiges Rechteck sein, da es keine horizontale Gerade gibt, so dass der Schnitt von R' mit der Geraden und der Schnitt der Geraden mit der Kante e zu derselben Zusammenhangskomponente gehören.

Die Parität eines Punkte v auf der Kante e ist bestimmt durch die Anzahl der Liniensegmente von $\overline{\mathcal{L}}$, die die horizontale Gerade h_v durch v schneidet. Wenn h_v unter $\overline{p_k}$ ist, dann schneidet die Gerade ein absteigendes Segment für jedes aufsteigende Segment von $\overline{\mathcal{L}}$. Dies ergibt sich daraus, dass das erste absteigende Segment auf jeden Fall geschnitten wird und das letzte geschnittene Segment ein aufsteigendes ist. Damit ist v gerade. Wenn h_v über $\overline{p_k}$ liegt, dann schneidet h_v ein aufsteigendes Segment für jedes absteigende Segment und das Segment zwischen p_1 und p_0. Damit ist v ungerade.

Damit sind alle Punkte von e gerade, wenn $\overline{y_{p_k}} = y_{p_0}$ gilt. Wenn $\overline{y_{p_k}} = y_{p_1}$ sind alle Punkte auf e ungerade und ansonsten ändert sich die Parität nur in $(x_{p_0}, \overline{y_{p_k}})$ und q ist ungerade. Dies beweist (i).

(ii) folgt direkt aus der Definition von Z_{ver}. Die Anzahl der degenerierten Rechtecke ist offensichtlich $|g \cap S| - 1$. Es kann außerdem maximal zwei nicht degenerierte Rechtecke geben, die an der Geraden g angrenzen, eins links und eins rechts. Da an jedem

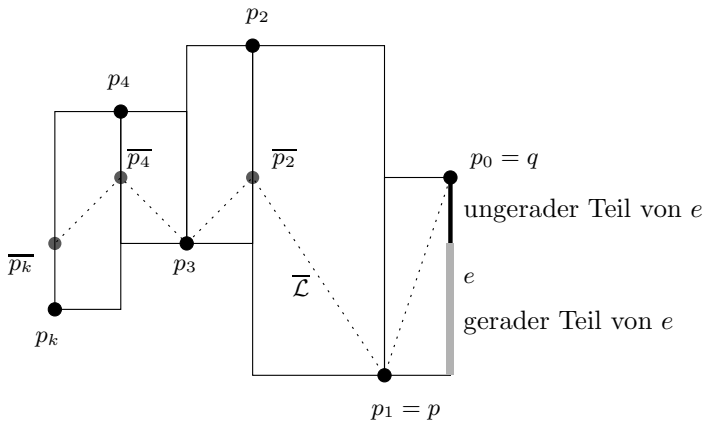


Abbildung 3.16: die Polygonale Kette $\overline{\mathcal{L}}$ zu der Kante e verläuft nur in dem horizontalen Streifen zwischen p_0 und p_1. Ihre vertikale Ausdehnung nimmt von rechts nach links gesehen nicht zu, sondern höchstens ab.

Punkt mindestens ein Rechteck angrenzen muss, folgt aus $d = 0$, dass $d' > 0$ gelten muss. Da jede vertikale Seite eines Rechtecks einen Punkt aus S enthalten muss, ist der Punkt p eine Ecke aller Rechtecke R'_i. Wenn es degenerierte Rechtecke gibt, bilden diese ein durchgehendes Liniensegment. Die Endpunkte des Liniensegments entsprechen dann p_1 und p_2 und somit muss jedes nicht degenerierte Rechteck entweder an p_1 oder p_2 angrenzen (vgl. Abbildung 3.17).

(iii) folgt direkt aus (ii).

Für (iv) nehmen wir erstmal an, dass $d = 0$ gilt. Dann folgt aus (ii), dass $d' \in \{1, 2\}$ und $g \cap P = \{p\}$ gelten muss. Aus (i) ergibt sich, dass das maximale, verbundene Liniensegment von ungeraden Punkten auf jeder vertikalen Seite eines Rechtecks einen Punkt aus S enthalten muss. Das heißt, dieser Punkt muss p sein. Damit gibt es maximal zwei Punkte von wechselnder Parität und maximal ein ungerades Liniensegment auf g. Aus dem Beweis von (i) ergibt sich, dass sich die Parität nur in Punkten vom Typ $(x_{p_0}, \overline{y_{p_k}})$ ändern kann und $\overline{y_{p_k}}$ die y-Koordinate eines Punktes in der Menge $\{p_0, \ldots, p_k\}$ ist.

Für den Fall $d > 0$ enthalten alle degenerierten Rechtecke nur ungerade Punkte. Aus (ii) ergibt sich, dass $g \cap (R_1 \cup \cdots R_d) = \text{Seg}[p_1, p_2]$ und dass jedes der maximal zwei nicht degenerierten Rechtecke eine Ecke entweder in p_1 oder p_2 hat. Damit ergibt sich ebenfalls die Behauptung.

Für den Beweis von (v) machen wir eine Fallunterscheidung abhängig von d'. Sei e wieder das Segment aus Fall (i). Falls $d' = 0$ ist, schneidet g nur degenerierte Rechtecke. Damit gibt es keinen Punkt mit wechselnder Parität und somit ist das ungerade Segment $\{x_g\} \times [b_g, t_g]$. Ansonsten kann man ohne Beschränkung der Allgemeinheit annehmen, dass e in g enthalten ist. Wenn $e = \{x_g\} \times [b_g, t_g]$ stimmt, sind wir fertig, denn aus (i) folgt, dass e entweder keinen Punkt von wechselnder Parität hat und damit alle Punkte

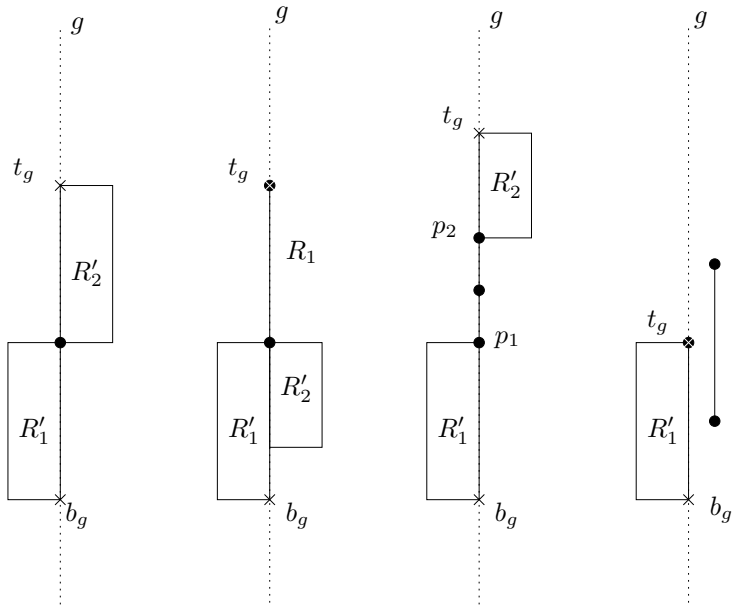


Abbildung 3.17: Die Kanten der Rechtecke, die von der Geraden g geschnitten werden, liegen alle auf dem Segment zwischen b_g und t_g. Dabei sind die nicht degenerierten Rechtecke R'_1 und R'_2, sofern sie existieren, immer am oberen oder unteren Punkt dieses Segmentes zu finden

auf e die gleiche Parität haben oder $c = c_1$ ist der einzige Punkt von wechselnder Parität und das ungerade Liniensegment ist $\{x_g\} \times [c_1, t_g]$, wobei $t_g = y_p$ gilt.

Wenn $e \neq \{x_g\} \times [b_g, t_g]$, dann gibt es ein weiteres Rechteck R_p in R_{ver} mit $R_p = R(\{p, r\})$ und $x_p \leq x_r, y_p < y_r$. Dieses Rechteck grenzt an R an und ist entweder degeneriert oder geht nach rechts von der Geraden weg. Wenn R_p nicht degeneriert ist, dann sind alle Punkte auf $\{x_g\} \times [b_g, t_g] \setminus e$ gerade, da es rechts keine relevanten Rechtecke gibt. In diesem Fall gibt es kein ungerades Segment auf g, wenn e komplett gerade ist. Sollte e komplett ungerade sein, ist das ungerade Segment $\{x_g\} \times [b_g, y_p]$, wobei $y_p = c_1$ gilt und falls c ein Punkt von wechselnder Parität ist, dann ist das ungerade Segment $\{x_g\} \times [c, y_p]$ mit $c = c_1$ und $y_p = c_2$. Falls R_p degeneriert ist, dann muss $\{x_g\} \times [p, r]$ zum ungeraden Segment hinzugefügt werden, wie vorher angegeben. Nun lässt sich das gleiche Argument für ein mögliches Rechteck R_r, das mit r verbunden ist, anwenden.

□

Lemma 3.18 *Die Menge $\mathcal{V}$ aller ungeraden Liniensegmente ist ein schönes, minimales vertikales Cover, das ungerade MVC (vgl. Abbildung 3.18).*

Beweis Lemma 3.18 $\mathcal{V}$ ist offensichtlich ein Cover für R_{ver}. Sei l eine horizontale Gerade, die A_{ver} schneidet. Betrachte eine verbundene Komponente C von G_l. Sei k

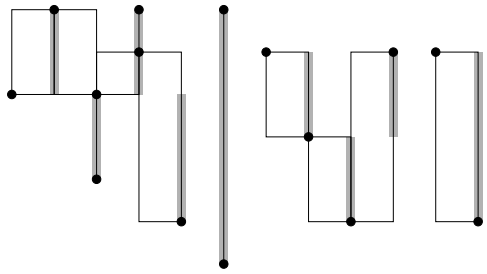


Abbildung 3.18: Das ungerade, minimale, vertikale Cover einer Punktmenge. Jedes Segment des Covers enthält einen Punkt der Punktmenge.

die Anzahl der Knoten in C. Wenn k gerade ist, dann muss jedes Cover mindestens $\frac{k}{2}$ Knoten von C enthalten und $\mathcal{V}$ enthält genau $\frac{k}{2}$ Knoten von C. Wenn jedoch $k > 1$ ungerade ist, dann muss jedes Cover mindestens $\frac{(k-1)}{2}$ Knoten von C enthalten und $\mathcal{V}$ enthält exakt $\frac{(k-1)}{2}$ Knoten. Wenn $k = 1$ gilt, muss jedes Cover diesen Knoten enthalten und $\mathcal{V}$ enthält diesen Knoten, da er zu einem degenerierten Rechteck gehört. Damit ist $\mathcal{V}$ ein minimales Cover. Aus Lemma 3.17 (i) folgt, dass $\mathcal{V}$ ein schönes Cover ist.

□

Lemma 3.19 *Die Berechnung des ungeraden, minimalen vertikalen Covers geht in Zeit* $O(n \log n)$ *und benötigt* $O(n)$ *Platz.*

Beweis Lemma 3.19 Das MVC wird mittels eines Sweep-Algorithmus berechnet. Dabei ist die Sweep-Line eine horizontale Gerade, die sich von unten nach oben bewegt. Seien $x_1, \ldots x_n$ die verschiedenen x-Koordinaten der Punktmenge, aufsteigend sortiert. Für jede vertikale Gerade $g_i : x = x_i$ werden in einem vorhergehenden Schritt die Punkte b_i und t_i berechnet, so dass $g_i \cap A_{\text{ver}} = [b_i, t_i]$ gilt. Dafür reicht es aus die Punktmenge in lexikographischer Reihenfolge durchzugehen. Für jedes g_i werden die Werte β_i und τ_i eingeführt, die am Anfang alle auf ∞ gesetzt werden. Nach dem Sweep-Algorithmus lässt sich aus diesen Werten das ungerade Liniensegment wie folgt bestimmen:

$\beta_i = \tau_i = \infty$ Es gibt kein ungerades Liniensegment auf g_i
sonst Das ungerade Liniensegment ist $x_i \times [\beta_i, \tau_i]$

Diese beiden Variablen reichen, da es nach Lemma 3.17 (iv) höchstens ein ungerades Segment auf g_i gibt.

Der Sweep-Algorithmus arbeitet wie folgt: Nach Lemma 3.17 (iv) gibt es für jeden Punkt c mit wechselnder Parität einen Punkt r in der Punktmenge S mit $y_c = y_r$. Diese Punkte müssen bestimmt werden, um die ungeraden Segmente zu berechnen.

Die Ereignis-Struktur ist eine sortierte Liste der y-Koordinaten der Punktmenge. Dabei können Koordinaten mehrmals auftreten, damit an jedem Punkt aus der Punktmenge S genau ein Ereignis stattfindet. Die Sweep-Status-Struktur ist ein balancierter

Binärbaum $\mathcal{T}$, zum Beispiel ein AVL-Baum. Jeder Knoten des Binärbaums entspricht einer Zusammenhangskomponente von G_l, wobei l der aktuellen Position der horizontalen Sweep-Line entspricht.

Am Anfang ist der Binärbaum $\mathcal{T}$ leer. Bei jedem Ereignis wird $\mathcal{T}$ aktualisiert. Für jede Komponente C von G_l werden zwei Indizes l_C und r_C gespeichert, so dass der Knoten von C, der ganz links ist, auf g_{l_C} liegt, und der Knoten von C, der ganz rechts ist, auf g_{r_C} liegt. Dann ist die Ordnung zwischen zwei Komponenten C und C' auf dem Binärbaum wie folgt definiert:

$$\begin{aligned} C' < C &\Leftrightarrow r'_C < l_C \\ C' > C &\Leftrightarrow r_C < l'_C \end{aligned}$$

Die Zusammenhangskomponenten sind disjunkt und können sich in dem Sinne nicht überlappen, dass die Knoten einer Zusammenhangskomponente C_1 entweder alle links von allen Knoten oder rechts von allen Knoten einer anderen Komponente C_2 sind. Damit ist die Ordnung wohldefiniert.

Folgende Änderungen an den Zusammenhangskomponenten (vgl. Abbildung 3.19) können bei einem Ereignis auftreten:

- Eine neue Komponente taucht auf
- Eine alte Komponente verschwindet
- Eine Komponente wird durch eine neue ersetzt
- Eine Komponente ändert ihre Größe
- Zwei oder drei Komponenten verschmelzen zu einer Komponente
- Eine Komponente wird in zwei oder drei Komponenten geteilt

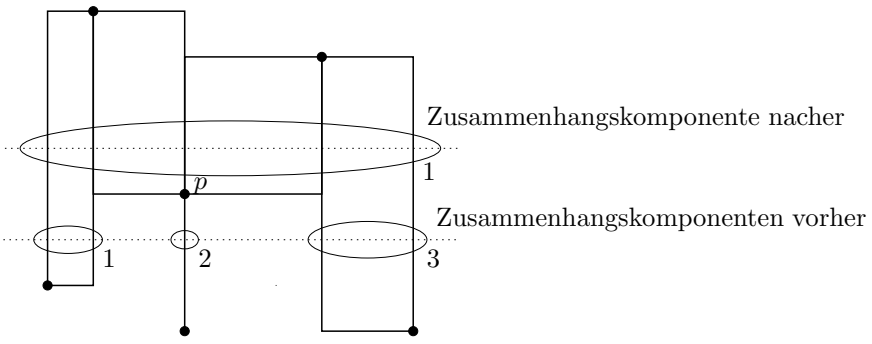


Abbildung 3.19: Drei einzelne Zusammenhangskomponenten verschmelzen am Ereignis von Punkt p zu einer Zusammenhangskomponente

Es lässt sich in konstanter Zeit entscheiden, welches der Ereignisse auftritt. Dazu werden die Werte b_i und t_i betrachtet, sowie die Wert $b_{i\pm1}$ und $t_{i\pm1}$, sofern diese existieren. i ist

dabei der Index der vertikalen Linie durch den Punkt, der für das Ereignis verantwortlich ist. Für jedes Ereignis müssen maximal die Einträge von drei Zusammenhangskomponenten geändert werden. Jede dieser Änderungen geht in $O(\log n)$.

Die Werte β_i und τ_i für jede Gerade g_i werden während des Sweeps berechnet. Zu jedem Zeitpunkt enthalten sie die Informationen über das bereits entdeckte ungerade Liniensegment von g_i. Wenn $\beta_i = \infty$ gilt, gibt es bis jetzt kein ungerades Liniensegment. Wenn $\beta_i \neq \infty$ und $\tau_i = \infty$ gilt, gibt es ein ungerades Liniensegment mit dem unteren Endpunkt (x_i, β_i). Sollte sowohl $\beta_i \neq \infty$ und $\tau_i \neq \infty$ gelten, so ist das ungerade Liniensegment auf g_i gegeben durch: $x_i \times [\beta_i, \tau_i]$. Damit muss bei jedem Ereignis des Sweep-Algorithmus überprüft werden, ob irgendwelche ungeraden Segmente an y_l starten oder aufhören, wobei y_l die y-Koordinate der Sweep-Line ist. Nach Lemma 3.17 (v) sind Punkte von wechselnder Parität immer Endpunkte von ungeraden Liniensegmenten. Punkte ganz oben oder ganz unten von $A_{\text{ver}} \cap g_i$ können Endpunkte sein. Um alle Endpunkte zu finden, müssen wir die alten und neuen Einträge der geänderten Komponenten von $\mathcal{T}$ anschauen. Unterste Punkte von $A_{\text{ver}} \cap g$ treten auf, wenn eine Komponente auftaucht, wächst oder wenn Komponenten verschmolzen werden. Obere Punkte von $A_{\text{ver}} \cap g$ treten auf, wenn eine Komponente verschwindet, schrumpft oder wenn Komponenten geteilt werden. Punkte mit wechselnder Parität können auftreten, wenn die Größe einer Komponente sich ändert, Komponenten verschmolzen oder geteilt werden oder wenn eine Komponente durch eine andere ersetzt wird. Wenn ein unterster Punkt b_i gefunden wird, müssen wir überprüfen, ob b_i ungerade ist. Sollte dies der Fall sein, so ist b_i der Startpunkt eines ungeraden Segmentes. Dies lässt sich feststellen, indem wir uns l_C, r_C und i ansehen, wobei C die Komponente ist, die b_i enthält. Wenn $l_C = r_C$ gilt, handelt es sich um ein degeneriertes Rechteck und damit ist b_i ungerade. Ebenfalls ist b_i ungerade, wenn die Parität von l_C und i unterschiedlich ist. In diesen Fällen setzen wir $\beta_i = b_i$. Wenn wir einen Punkt von wechselnder Parität entdecken, wird geschaut, ob es bereits ein ungerades Segment auf g_i gibt. Wenn $\beta_i = \infty$ gilt, gibt es bis jetzt kein ungerades Liniensegment und damit wird $\beta_i = y_l$ gesetzt. Sollte es bereits ein ungerades Liniensegment geben, so muss es sich hierbei um den Endpunkt handeln und es wird $\tau_i = y_l$ gesetzt. Sollten wir einen oberen Punkt t_i entdecken, so muss geschaut werden, ob es ein ungerades, nicht fertiges Liniensegment auf g_i gibt. Dies ist der Fall, wenn $\beta_i \neq \infty$ und $\tau_i = \infty$ gilt. In diesem Fall ist t_i der Endpunkt des ungeraden Liniensegmentes und es wird $\tau_i = t_i$ gesetzt.

Da der Sweep-Algorithmus n Ereignisse hat, wo jeweils maximal drei Komponenten geändert werden, gibt es insgesamt nur $O(n)$ Änderungen. Jede Änderung kostet $O(\log n)$, so dass sich als Gesamtlaufzeit $O(n \log n)$ ergibt.

□

3.2.4 Der Approximations-Algorithmus

Der Algorithmus ApproxMMN (vgl. Algorithmus 3.2) besteht aus vier Phasen. In Phase 0 werden alle Punkte vom Typ xnbor und ynbor, sowie die Menge Z berechnet. In Phase 1 wird ein Netzwerk für alle Paare in Z_{ver} und Z_{hor} berechnet, das Netzwerk N_1. Dieses

besteht aus der Vereinigung eines schönen, minimalen, vertikalen Covers C_{ver} und eines schönen, minimalen, horizontalen Covers C_{hor}, sowie maximal einem Liniensegment für jedes Rechteck aus $R_{\text{ver}} \cup R_{\text{hor}}$. In Phase 2 werden Treppenpolygone berechnet. Die Vereinigung der Flächen dieser Polygone ist das Gebiet A_3. Das Netzwerk N_2 ist der Rand dieser Polygone und Segmente, um diese Ränder mit N_1 zu verbinden. In Phase 3 wird das Netzwerk N_3 berechnet, das aus Segmenten besteht, die in A_3 liegen. Das entstehende Netzwerk $N_1 \cup N_2 \cup N_3$ ist dann ein Manhattan-Netzwerk für alle Paare aus Z. Da Z eine erzeugende Menge ist, ist dieses Netzwerk damit ein Manhattan-Netzwerk für die gesamte Punktmenge S.

Phase 0

In dieser Phase werden alle Punkte vom Typ xnbor und ynbor, sowie daraus die erzeugende Menge Z berechnet. Die Mengen werden so gespeichert, dass von einem Punkt p der Zugriff auf xnbor, ynbor und die horizontalen und vertikalen Vorgänger/Nachfolger in konstanter Zeit möglich ist.

Phase 1

Als erstes wird das schöne, ungerade MVC C_{ver} und das schöne, ungerade MHC C_{hor} berechnet. Danach werden zusätzliche Liniensegmente L nach Lemma 3.16 berechnet. $C_{\text{ver}}, C_{\text{hor}}$ und L lassen sich so berechnen, dass von jedem Punkt $p \in S$ in konstanter Zeit Zugriff auf die maximal zwei Cover-Segmente aus $C_{\text{ver}} \cup C_{\text{hor}}$, die p enthalten, möglich ist, sowie auf die zusätzlichen Segmente der maximal vier an p angrenzenden Rechtecke.

Aus Lemma 3.12, Lemma 3.16 und Lemma 3.19 ergibt sich, dass $N_1 = C_{\text{ver}} \cup C_{\text{hor}} \cup L$ in Zeit $O(n \log n)$ berechnet werden kann und dass $|N_1| \leq |N_{\text{opt}}| + H + W$ gilt.

Phase 2

In den meisten Fällen ist N_1 kein Manhattan-Netzwerk für Z_{quad}, sondern es werden weitere Segmente dafür benötigt. Um die Länge dieser Segmente zu beschränken, wird die Ebene in die zwei Regionen A_{12} und A_3 aufgeteilt, wie bereits in Abschnitt 3.2.1 erwähnt. Wir wollen A_3 so definieren, dass $|N_{\text{opt}} \cap A_3|$ so groß ist, dass wir die Länge der Segmente, die für A_3 gebraucht werden, beschränken können. Allerdings lässt sich A_3 nicht so definieren, dass sich durch Hinzufügen von Segmenten, die nur in A_3 liegen, ein Manhattan-Netzwerk für Z_{quad} ergibt, und gleichzeitig die Länge der Segmente beschränkt ist. Daher werden die Segmente in zwei disjunkte Mengen N_2 und N_3 eingefügt, wobei $N_1 \cup N_2 \subseteq A_{12}$ und $N_3 \subseteq A_3$ gilt. Dies erlaubt es $|N_1 \cup N_2|$ durch $3|N_{\text{opt}} \cap A_{12}|$ und $|N_3|$ durch $2|N_{\text{opt}} \cap A_3|$ zu beschränken.

Für die Definition von A_3 werden folgende Punktmengen gebraucht:

$$P(q,t) := \{p \in S \cap Q(q,t) | (p,q) \in Z_{\text{quad}}\}, 1 \leq t \leq 4$$

$P(q,t)$ enthält die Punkte p, die im Quadranten t bezüglich q liegen und bei denen sich das Punktpaar (p,q) in Z_{quad} befindet. In Abbildung 3.20 ist $P(q,1) = \{p_1, \ldots, p_5\}$.

Algorithmus 3.2 Faktor drei Approximations-Algorithmus

Faktor drei Approximation eines minimalen Manhattan-Netzwerkes

Eingabe: Eine Punktmenge S

Ausgabe: Ein Manhattan-Netzwerk M, das maximal um den Faktor drei schlechter ist als ein minimales Manhattan-Netzwerk.

procedure ApproxMMN(S)

Phase 0: ▷ Nachbarn und erzeugende Mengen

for $p \in S$ und $t \in [1, 2, 3, 4]$ **do**

Berechne p.xnbor[t] und p.ynbor[t] ▷ Siehe Seite 39

end for

Berechne $Z = Z_{\text{ver}} \cup Z_{\text{hor}} \cup Z_{\text{quad}}$ ▷ Siehe Seite 40

Phase 1: ▷ Berechne N_1

Berechne das ungerade MVC C_{ver} und das ungerade MHC C_{hor} ▷ Siehe Lemma 3.19 auf Seite 48

Berechne die zusätzlichen Liniensegmente L

$N_1 = C_{\text{ver}} \cup C_{\text{hor}} \cup L$

$N_2 = \emptyset$

$N_3 = \emptyset$

Phase 2: ▷ Berechne N_2

for Verbundene Komponente A von A_3 **do** ▷ Definition von A_3: Seite 57

$N_2 = N_2 \cup (\partial A \setminus \bigcup N_1)$ ▷ Füge den Rand von A ein

if $q_a \notin \bigcup N_1$ **then** ▷ vgl. Abbildung 3.20 auf der nächsten Seite

$N_2 = N_2 \cup \{s_A\}$ ▷ vgl. Abbildung 3.20

end if

end for

Phase 3: ▷ Berechne N_3

for Verbundene Komponente A von A_3 **do**

$N_3 = N_3 \cup \text{BRIDGE}(A)$ ▷ Siehe Algorithmus 3.3 auf Seite 60

end for

return $N = N_1 \cup N_2 \cup N_3$

end procedure

Aufgrund der Definition von Z_{quad} ergibt sich, dass $Q(p,t) \cap P(q,t) = \{p\}$ für jedes $p \in P(q,t)$ gilt. Damit ist das Gebiet

$$A_{\text{quad}}(q,t) := \bigcup_{p \in P(q,t)} R(\{p,q\})$$

ein Treppenpolygon. Die Punkte in $P(q,t)$ sind die Treppen des Polygons und q ist die gegenüberliegende Ecke der Treppenstufen. In Abbildung 3.20 ist $A_{\text{quad}}(q,t)$ die Vereinigung aller schattierten Gebiete. Um mit Hilfe der Flächen $A_{\text{quad}}(p,t)$ die Menge A_3 zu definieren, ist die Idee die Gebiete zu entfernen, die Segmente aus N_1 enthalten, sowie die Gebiete, die zur Konstruktion der Manhattan-Netzwerkes für Z_{quad} nicht nötig sind.

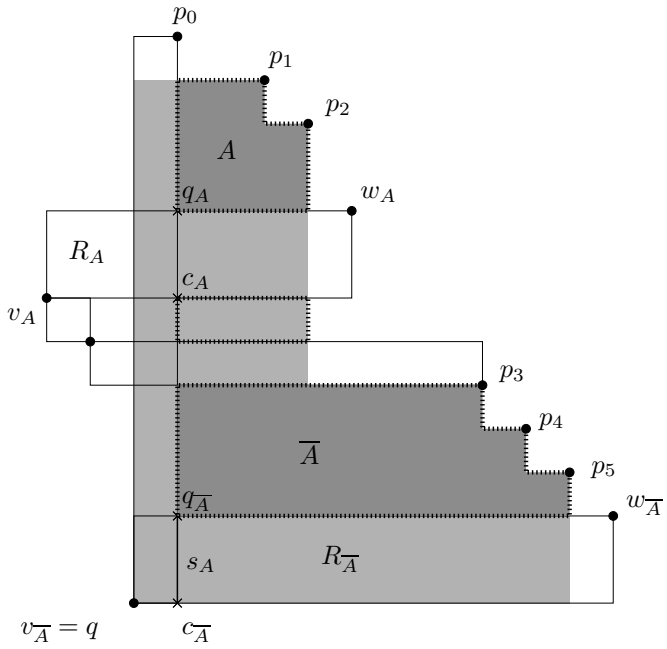


Abbildung 3.20: Konstruktion der Gebiete für die Region A_3. Alle schattierten Gebiete zusammen sind $A_{\text{quad}}(q,1)$. Die gepunkteten Gebiete bilden $\Delta(q,1)$, dies sind die Gebiete aus $A_{\text{quad}}(q,1)$, die nicht in $A_{\text{hor}} \cup A_{\text{ver}}$ enthalten sind. Die dunkel schattierten Gebiete sind die Gebiete von $\Delta(q,1)$, die mindestens einen Punkt aus S enthalten, weswegen das mittlere, gepunktete Gebiet nicht darin enthalten ist. Sie bilden $\delta(q,1)$. Die Menge $P(q,1)$ entspricht den Punkten $\{p_1,\ldots,p_5\}$. Der Punkt p_0 ist der vertikale Nachfolger von q. Das Segment s_A ist das verbindende Segment, das, sofern nötig, in N_2 eingefügt wird.

Sei $\Delta(q,t)$ definiert als:

$$\Delta(q,t) := \text{int}\left(A_{\text{quad}}(q,t) \setminus (A_{\text{hor}} \cup A_{\text{ver}})\right)$$

Dabei ist $\text{int}(M)$ das Innere einer Menge $M \subseteq \mathbb{R}^2$. In Abbildung 3.20 ist $\Delta(q,1)$ die Vereinigung der schattierten Gebiete mit gepunkteter Umrandung. Sei $\delta(q,t)$ die Vereinigung der Zusammenhangskomponenten A aus $\Delta(q,t)$, für die $\partial A \cap P(q,t) \neq \emptyset$ gilt. Im Gegensatz zu $\Delta(q,t)$ enthält jede Komponente aus $\delta(q,t)$ auf dem Rand mindestens einen Punkt p mit $(p,q) \in Z_{\text{quad}}$ und $p \in Q(q,t)$. In Abbildung 3.20 ist $\delta(q,1)$ die Vereinigung der beiden dunkel schattierten Flächen A und $\overline{A}$.

Aufgrund der Konstruktion von $\delta(q,t)$ muss jede Zusammenhangskomponente $A \in \delta(q,t)$ ebenfalls ein Treppenpolygon sein. Die Treppen von A entsprechen jeweils den Eingabepunkten auf ∂A. Sei nun q_A der Punkt auf ∂A, der q am nächsten ist (vgl. Abbildung 3.20). Diese Ecke liegt den Treppenstufen gegenüber.

Lemma 3.20 *Das Innere der Gebiete vom Typ $\delta(q,t)$ ist paarweise disjunkt.*

Beweis Lemma 3.20 Für jedes Paar $(p,q) \in Z_{\text{quad}}$ definieren wir das verbotene Gebiet F_{pq} (vgl. Abbildung 3.21) als die Vereinigung aus:

- dem Schnitt der Halbebene, die durch die horizontale Gerade durch q definiert ist und p nicht enthält mit dem offenen Streifen zwischen den vertikalen Geraden durch p und q
- und dem umschließenden Rechteck $R(\{p,q\})$.

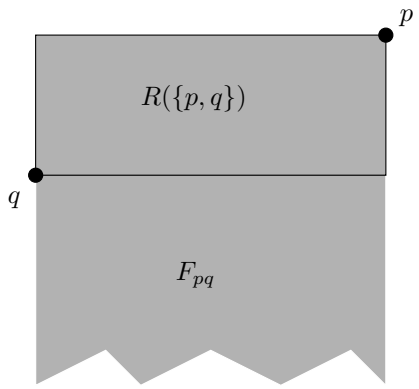


Abbildung 3.21: Die schattierte Fläche entspricht dem verbotenden Gebiet F_{pq} zu p und q. Außer p und q kann kein weiterer Punkt der Punktmenge in diesem Gebiet liegen.

Außer p und q kann F_{pq} keinen weiten Punkt aus S enthalten, da sonst (p,q) nicht in Z_{quad} liegen würde:

$$F_{pq} \cap (S \setminus \{p,q\}) = \emptyset$$

Sollte F_{pq} einen weiteren Punkt r enthalten, würde dies der Definition von Z_{quad} widersprechen, da dann (p,q) nicht in Z_{quad} liegen würde, sondern (p,r).

Angenommen, es gibt einen Punkt s, der im Innern von zwei Regionen liegt:

$$\begin{aligned} & s \in \text{int}(delta(q,t)) \cap \text{int}(\delta(q',t')) \\ \wedge \quad & \delta(q,t) \neq \delta(q',t') \end{aligned}$$

Es ist offensichtlich, dass $q \neq q'$ gilt. Sollte $q = q'$ gelten, muss auch $t = t'$ gelten, da $\text{int}(\delta(q,t)) \cap \text{int}(\delta(q,t') = \emptyset$ für $t \neq t'$.

Da $\delta(q,t), \delta(q',t') \subseteq A_{\text{quad}}$ gilt, muss es Punkte p und p' geben, so dass s in den umschließenden Rechtecken liegt:

$$s \in R(\{p,q\}) \cap R(\{p',q'\})$$

Sei $R = R(\{p,q\})$ und $R' = R(\{p',q'\})$. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass p rechts oben von q liegt. Alle anderen Fälle sind symmetrisch. p' und q' können nicht in R liegen, da $R \in F_{pq}$ gilt und dieses Gebiet keine weiteren Punkte enthält. Aus dem gleichen Grund folgt, dass p und q nicht in R' liegen können. Seien l und r die anderen beiden Ecken von R (vgl. Abbildung 3.22). Dann gibt es drei Fälle:

Fall 1: $R' \cap \{l,r\} = \emptyset$

Da in diesem Fall $R' \cap \{l,r\} = \emptyset, R' \cap \{p,q\} = \emptyset$ und $R \cap R' \neq \emptyset$ gilt, liegt R' entweder im unbeschränkten vertikalen Streifen $S_1 = (x_q, x_p) \times (-\infty, +\infty)$ oder im unbeschränkten vertikalen Streifen $S_2 = (-\infty, +\infty) \times (y_q, y_p)$ (vgl. Abbildung 3.22). p' und q' können nicht auf dem Rand von S_1 bzw. S_2 liegen, da

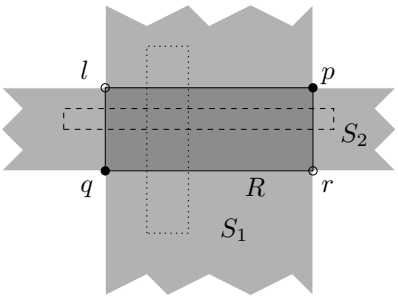


Abbildung 3.22: Fall 1 des Beweises, dass die Regionen $\delta(q,t)$ disjunkt sind. Das Rechteck R' muss entweder im Streifen S_1 (gepunktet skizziert) oder im Streifen S_2 (gestrichelt skizziert) liegen.

sonst (p,q) oder (p',q') nicht in Z_{quad} liegen könnten. Wenn $R' \subset S_1$ gilt, dann muss p' oder q' in F_{pq} liegen. Dies ist aber durch die Konstruktion des verbotenen Gebietes ausgeschlossen. Analog muss p oder q' in $F_{p'q'}$ liegen, wenn $R' \subset S_2$ gilt. Das kann ebenfalls nicht sein, womit dieser Fall nicht auftreten kann.

Fall 2: $R' \cap \{l,r\} = \{r\}$

Da R' den Punkt r aus R enthält, liegt die linke, obere Ecke von R' in R. Die Ecke kann nicht außerhalb von R liegen, da sonst der Punkt q, wenn die Ecke links von R, oder der Punkt p, wenn die Ecke oberhalb von R läge, enthalten wäre. Damit kann diese Ecke weder dem Punkt p' noch dem Punkt q' entsprechen. Daraus folgt jedoch, dass die linke untere Ecke von R' entweder dem Punkt p' oder dem Punkt q' entsprechen muss, diese Ecke jedoch in F_{pq} liegt. Da in F_{pq} kein Punkt außer p und q liegen kann, kann auch dieser Fall nicht auftreten.

Fall 3: $R' \cap \{l, r\} = \{l\}$

Betrachte Abbildung 3.23. Dann liegt die rechte, untere Ecke von R' in R und die obere, rechte Ecke oberhalb von R. Da die untere, rechte Ecke in R liegt, muss die obere, rechte Ecke dem Punkt p' oder q' entsprechen. Sollte sie dem Punkt p' entsprechen, würde der Punkt q von R in $F_{p'q'}$ liegen, was durch die Konstruktion von $F_{p'q'}$ ausgeschlossen ist. Also entspricht die rechte, obere Ecke von R' dem Punkt q' und die linke, untere Ecke dem Punkt p'. Diese Konstellation

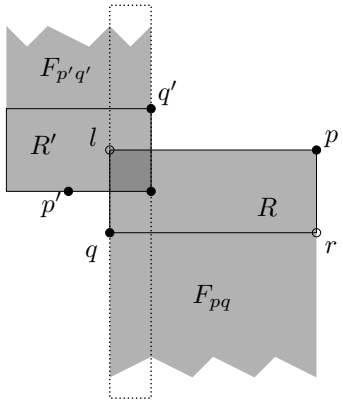


Abbildung 3.23: Fall 3 des Beweises, dass die Regionen $\delta(q,t)$ disjunkt sind. R' enthält l und die Ecke p' liegt links von R, sowie q' oberhalb von R. Der gestrichelt markierte Streifen zwischen q und q' ist komplett in $F_{pq} \cup F_{p'q'}$ enthalten, womit dort kein Punkt liegen kann und somit das Rechteck $R(\{q, q'\})$ in R_{ver} liegt und diese Fläche nicht in $\delta(q,t)$ sein kann.

ist denkbar. Allerdings gilt dann $R \cap R' \subset R(\{q, q'\})$. Da der offene, vertikale Streifen zwischen q und q' komplett in $F_{pq} \cup F_{p'q'}$ enthalten ist, sind in diesem Streifen aber keine Punkte enthalten. Damit ist das Paar (q, q') jedoch in Z_{ver} enthalten. Damit gilt aber:

$$s \in R \cap R' \subset R(\{q, q'\}) \subset A_{\text{ver}}$$

Das widerspricht jedoch der Annahme, dass $s \in \delta(q,t)$ gilt, da $\delta(q,t)$ im Komplement von A_{ver} enthalten ist. Damit ist auch dieser Fall nicht möglich.

□

Damit lassen sich nun die Zusammenhangskomponenten $A \in \delta(q,t)$ unabhängig behandeln. Schließlich definieren wir A_3 und A_{12}:

$$\begin{aligned} A_3 &:= \bigcup_{t\in\{1,2,3,4\}} \bigcup_{q\in S} \delta(q,t) \\ A_{12} &:= \mathbb{R}^2 \setminus A_3 \end{aligned}$$

Damit gilt $N_1 \subset A_{12}$, da $\delta(q,t)$ so konstruiert wurde, dass kein Pfad aus N_1 darin liegen kann. Nun wird das Netz N_2 konstruiert. Für jede Zusammenhangskomponente $A \in A_3$ wird $\partial A \setminus \bigcup N_1$ in N_2 eingefügt und getestet, ob es einen Pfad von q nach q_A in N_1 gibt. q_A bezeichnet dabei den Punkt, der den Treppenstufen von A gegenüber liegt (vgl. Abbildung 3.20 auf Seite 53). Wenn nicht, wird ein weiteres Segment in N_2 eingefügt. Dieses Segment wird in A_{hor} liegen und später definiert. Da sowohl A_{hor} als auch ∂A in A_{12} liegen, gilt damit: $N_2 \subset A_{12}$. Das Netzwerk N_3 wird in Phase 3 des Algorithmus so konstruiert, dass $N_3 \subset A_3$ gilt.

Nun wird beschrieben, wie sich die Regionen $\delta(q,t)$ berechnen lassen. Dazu werden als erstes die Mengen $P(q,t) = \{p_1, \ldots, p_m\}$ konstruiert. Diese enthalten alle Punkte p aus S, für die gilt, dass (p,q) in Z_{quad} liegt, und dass p in $Q(q,t)$ liegt. Da von jedem Punkt q Zugriff in konstanter Zeit auf seine Partner in Z_{quad} möglich ist, reicht es dazu aus, alle Punkte durchzugehen und die Partner in Z_{quad} jeweils nach $P(q,t)$ einsortieren. Das Ganze geht in $O(n)$, da $|Z_{\text{quad}}| \leq 4n$ gilt. Die Punkte in $P(q,t)$ werden danach entsprechend ihres x-Abstandes von q sortiert. Dies geht in Zeit $O(n \log n)$. Da diese Punkte ein Treppenpolygon bilden, sind sie damit auch gleichzeitig nach ihrem y-Abstand von q absteigend sortiert. Nun muss nur noch entschieden werden, welche Punkte von $P(q,t)$ zur gleichen Zusammenhangskomponente von $\delta(q,t)$ gehören. Beispielsweise ist in Abbildung 3.20 auf Seite 53 die Menge $\{p_1, p_2\} \subset \partial A$ und $\{p_3, p_4, p_5\} \subset \partial\overline{A}$. Für die Beschreibung des Verfahrens nehmen wir an, dass $t = 1$ und $P(q,1) = \{p_1, \ldots p_m\}$ gilt. Für die anderen Quadranten ist das Verfahren symmetrisch. Alle Zusammenhangskomponenten A von $\delta(q,1)$ entsprechen einer Menge von aufeinanderfolgenden Punkten in $P(q,1)$. Aufgrund der Definition der Komponenten gilt für alle $p_i, p_j \in A : p_i.\text{ynbor}[3] = p_j.\text{ynbor}[3]$, da $p_i.\text{ynbor}[3]$ dem Punkt entspricht, der das Rechteck definiert, dass zwei Regionen trennt.

Diese Folgen von Punkten lassen sich finden, in dem man die Punkte $p_1, \ldots p_m$ nacheinander betrachtet. Sei p_i der aktuelle Punkt und A die aktuelle Zusammenhangskomponente. Genau dann, wenn $p_i.\text{ynbor}[3] \neq p_{i+1}.\text{ynbor}[3]$ gilt, gibt es ein Rechteck $R_A \in \mathbb{R}_{\text{hor}}$, das A von der nächsten Zusammenhangskomponente trennt. Dieses Rechteck R_A ist durch den Punkt $v_A = p_i.\text{ynbor}[3]$ und dem horizontalen Nachfolger w_A definiert. Aufgrund der Lage von v_A muss dieser eindeutig sein. Sei p_0 der vertikale Nachfolger von q. Dann gilt $x_{q_A} = x_{p_0} \wedge y_{q_A} = y_{w_A}$ (vgl. Abbildung 3.20 auf Seite 53).

Als letztes muss nun sichergestellt werden, dass $N_1 \cup N_2$ einen Manhattan-Pfad von q nach q_A enthält. In Phase 3 werden nachher Manhattan-Pfade von jedem Punkt $p_i \in \partial A$ nach q_A konstruiert. Wenn man diese Pfade mit dem Pfad von q_A nach q kombiniert, erhält man einen Manhattan-Pfad von jedem $p_i \in \partial A$ nach q, da q_A in $R(\{q, p_i\})$ liegt.

Da das Netzwerk N_3 komplett in A_3 liegt und das umschließende Rechteck $R(\{q, q_A\})$ in A_{12} liegt, können die Segmente aus N_3 nichts zu dem Pfad von q_A nach q beitragen.

Das Netzwerk N_1 enthält die Manhattan-Pfade P_{ver} von q nach p_0 und P_{hor} von v_A nach w_A, da $(q, p_0) \in Z_{\text{ver}}$ und $(v_A, w_A) \in Z_{\text{hor}}$. Wenn q_A im Pfad P_{ver} enthalten ist, dann enthält N_1 automatisch einen Pfad von q nach q_A. Sollte q_A in P_{hor} enthalten sein, dann enthält N_1 ebenfalls einen Pfad von q nach q_A, da der Schnittpunkt von P_{hor} und P_{ver} links von q_A liegen muss. Sollte jedoch $q_A \notin (P_{\text{ver}} \cup P_{\text{hor}})$ gelten, dann muss P_{hor} den Punkt $c_A = (x_{q_A}, y_{v_A})$ enthalten. Dieser Punkt entspricht dem Schnitt der vertikalen Geraden durch q_A mit der q_A entgegengesetzten Seite des Rechtecks R_A (vgl. Abbildung 3.20 auf Seite 53). Um einen Manhattan-Pfad von q nach q_A zu erhalten, reicht es, das Segment $l_A = [q_A, c_A]$ zu N_2 hinzuzufügen. Diese Segmente heißen *verbindende Segmente*.

Der Algorithmus ApproxMMN berechnet die beiden Pfade P_{ver} und P_{hor} nicht explizit, sondern testet nur, ob $q_A \notin \bigcup N_1$ gilt. Das ist äquivalent zu $q_A \notin (P_{\text{ver}} \cup P_{\text{hor}})$, da die Cover, aus denen N_1 gebildet wird, minimal sind und die umschließenden Rechtecke zu P_{ver} und P_{hor} die einzigen Rechtecke in $R_{\text{ver}} \cup R_{\text{hor}}$ sind, die l_A enthalten. Mit dem gleichen Argument und der Tatsache, dass die Kanten des Covers in $R_{\text{hor}} \cup R_{\text{ver}}$ sind, gilt $l_A \cap \bigcup N_1 = c_A$. Daraus ergibt sich, dass die verbindenden Segmente die Segmente aus N_1 maximal an Endpunkten schneiden. Diese Eigenschaft wird in einem späteren Beweis wichtig. Nun lässt sich folgendes Lemma zeigen:

Lemma 3.21 *Das Netzwerk N_2 lässt sich mit folgenden Eigenschaften in Zeit $O(n \log n)$ berechnen:*

(i) $N_2 \subset A_{12}$

(ii) Ein Segment aus N_1 und ein Segment aus N_2 schneiden sich maximal in den Endpunkten

(iii) Für jede Region $\delta(q, t)$ und jede Zusammenhangskomponente $A \in \delta(q, t)$ gilt: $\partial A \in N_1 \cup N_2$ und $N_1 \cup N_2$ enthält einen Manhattan-Pfad von q nach q_A.

Beweis Lemma 3.21 Die Eigenschaften (i) bis (iii) ergeben sich unmittelbar aus der oben angegebenen Beschreibung. Die Laufzeit für die Berechnung der Zusammenhangskomponenten ergibt sich ebenfalls aus der Beschreibung. Für jede Zusammenhangskomponente A aus A_3 kann das verbindende Segment l_A und die Menge $\partial A \setminus \bigcup N_1$ in Zeit $O(m)$ berechnet werden, wobei $m = |S \cap \partial A|$ gilt. Dies geht, da der Zugriff auf die m Rechtecke von $R_{\text{hor}} \cup R_{\text{ver}}$, die ∂A schneiden, sowie auf die $O(m)$ Segmente von N_1, die in diesen Rechtecken liegen, in konstanter Zeit möglich ist. Weiterhin lassen sich die Punkte q_A und w_A wie in der Beschreibung erwähnt sofort aus den vertikalen Nachbarn, bzw. horizontalen Nachfolgern der Punkte bestimmen.

□

Phase 3

In dieser letzten Phase wird ein Netzwerk N_3 konstruiert, so dass $N_1 \cup N_2 \cup N_3$ Pfade für alle Paare aus Z_{quad} enthält. Aufgrund von Lemma 3.21 (iii) reicht es aus, eine Menge $B(A)$ von Segmenten zu berechnen, so dass die Vereinigung von ∂A und $B(A)$ einen Manhattan-Pfad von jedem Punkt aus S auf ∂A zu q_A enthält. Diese Menge $B(A)$ „überbrückt“ A. Die Menge N_3 ist dann die Vereinigung aller Mengen vom Typ $B(A)$. Der Algorithmus BRIDGE(A), der die Menge $B(A)$ berechnet, ist ähnlich zu dem „thickes-first“ Algorithmus von Levocopulous und Östlin [LPRS82], der auch in Theorem 3.3 auf Seite 28 von Gudmundsson et al. [GLN01] verwendet wird. Allerdings lässt er sich nicht direkt verwenden, da dieser Segmente erzeugt, die nicht in A_3 liegen würden. Hier liese sich zwar, analog zum Algorithmus von Gudmundsson et al. [GLN01] eine exakte Lösung berechnen, allerdings würde diese die Laufzeit auf $O(n^3)$ erhöhen, ohne den Approximationsfaktor zu verbessern, die schon für die Netzwerke N_1 und N_2 jeweils nur ein Approximationfaktor von zwei erreicht wird.

Für die Beschreibung des Algorithmus BRIDGE (siehe Algorithmus 3.3) nehmen wir an, dass A in einer Region vom Typ $\delta(q, 1)$ liegt. Sei $\{p_1, \ldots, p_m\}$ wieder die Menge der Punkte auf ∂A, sortiert nach dem x-Abstand zu q. Da ∂A in $N_1 \cup N_2$ enthalten ist und ∂A bereits einen Manhattan-Pfad von q_A zu p_1 und p_m enthält, müssen nur noch Pfade von q_a zu der Menge $\{p_2, \ldots p_{m-1}\}$ berechnet werden. Sollte $m \leq 2$ gelten, so braucht nichts gemacht zu werden. Ansonsten gelten folgende Definitionen für $1 \leq j \leq m-1$ (vgl. Abbildung 3.24):

$$\begin{aligned} p'_j &:= \left(x_{p_j}, y_{p_{j+1}}\right) \\ a_j &:= \text{Seg}\left[\left(x_{q_A}, y_{p'_j}\right), p'_j\right] \\ b_j &:= \text{Seg}\left[\left(x_{p'_j}, y_{q_A}\right), p'_j\right] \end{aligned}$$

Die Länge von a_j wird mit $\alpha_j = |a_j|$ bezeichnet, analog gilt $\beta_j = |b_j|$. Im Folgenden wird das Treppenpolygon durch das Tupel $(q_A, p_1, \ldots, p_m)$ identifiziert. Sei B die Menge der Segmente, die der Algorithmus berechnet. Am Anfang ist $B = \emptyset$. Der Algorithmus wählt ein $i \in \{1, \ldots m-1\}$ und fügt, falls sie existieren, die Segmente a_{i-1} und b_{i+1} zu B hinzu. Damit sind in $B \cup N_1 \cup N_2$ die Pfade von q_A nach p_i, sowie von q_A nach p_{i+1} enthalten. Für die Menge der Pfade von q_A nach $\{p_2, \ldots, p_{i-1}\}$ und von q_A nach $\{p_{i+1}, p_{m-1}\}$ wird das Problem rekursiv für die Treppenpolygone $((x_{q_A}, y_{p_i}), p_1, \ldots, p_{i-1})$ und $\left((x_{p_{i+1}}, y_{q_A}), p_{i+2}, \ldots, p_m\right)$ gelöst.

Die Wahl von i wird nun definiert. Es gilt:

$$\alpha_1 < \cdots < \alpha_{m-1} \quad \wedge \quad \beta_1 > \cdots > \beta_{m-1}$$

Dann wird Λ definiert als:

$$\Lambda = \{j \in \{1, \ldots, m-1\} | \alpha_j \leq \beta_j\}$$

Wenn $\Lambda = \emptyset$ gilt, dann folgt daraus $\alpha_1 > \beta_1$, womit A flach und lang ist. In diesem Fall wählen wir $i = 1$, wodurch nur b_2 in B eingefügt wird. Ansonsten sei nun $i' = \max \Lambda$.

Algorithmus 3.3 Prozedur BRIDGE des Faktor drei Approximations-Algorithmus

procedure BRIDGE($A = (q_a, p_1, \ldots, p_m)$)
  for $i = 1$ to $m - 1$ **do**
    Berechne α_i und β_i
  end for
  return SUBBRIDGE$(1, m, 0, 0)$
end procedure

procedure SUBBRIDGE$(k, l, x_{\text{off}}, y_{\text{off}})$
  $A_{\text{curr}} = (q_a + (x_{\text{off}}, y_{\text{off}}), p_k, \ldots, p_l)$
  if $l - k < 2$ **then**
    return $\emptyset$
  end if
  $\Lambda = \{j \in \{k, \ldots, l-1\} : \alpha_j - x_{\text{off}} \leq \beta_j - y_{\text{off}}\}$
  $i = \max\{\Lambda \cup \{k\}\}$
  if $i < l - 1 \wedge \alpha_i - x_{\text{off}} \leq \beta_{i+1} - y_{\text{off}}$ **then**
    $i = i + 1$
  end if
  $B = \emptyset$
  if $i > 1$ **then**
    $B = B \cup \{a_{i-1} \cap A_{\text{curr}}\}$
  end if
  if $i < l - 1$ **then**
    $B = B \cup \{b_{i+1} \cap A_{\text{curr}}\}$
  end if
  $x_{\text{new}} = x_{p_{i+1}} - x_{q_a}$
  $y_{\text{new}} = y_{p_i} - y_{q_a}$
  return $B \cup$ SUBBRIDGE$(l, i-1, x_{\text{off}}, y_{\text{new}}) \cup$ SUBBRIDGE$(i+2, l, x_{\text{new}}, y_{\text{off}})$
end procedure

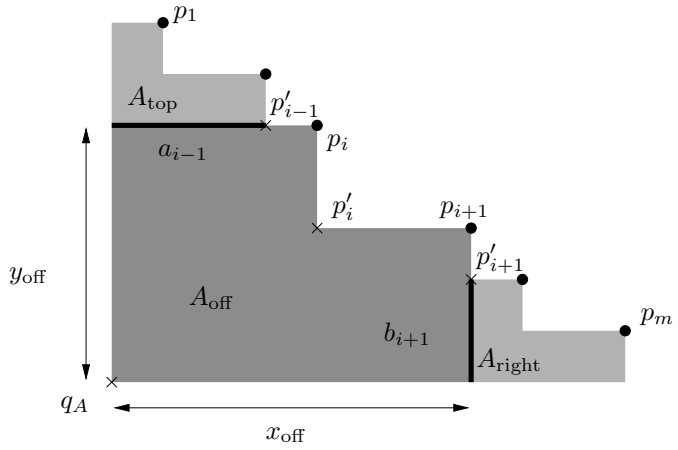


Abbildung 3.24: Die Bezeichnungen der Segmente und Punkte für den Algorithmus BRIDGE. Das Gebiet A_{off} ist der Bereich, der durch die Wahl der Segmente a_{i-1} und b_{i+1} bereits abgedeckt wurde und nun als Offset für die Länge in der Rekursion funktioniert. Dieser Offset wird durch x_{off} und y_{off} angegeben. Für die beiden Gebiete A_{right} und A_{top} werden die Kanten rekursiv berechnet.

Wenn $i' < m-1$ und $\alpha_{i'} \leq \beta_{i'+1}$ gilt, setze $i = i'+1$. In allen anderen Fällen setze $i = i'$. Diese Wahl soll gewährleisten, dass die Werte von α_{i-1} und β_{i+1} balanciert sind. Das erlaubt es $\alpha_{i-1} + \beta_{i+1}$ gegen $\min\{\alpha_i, \beta_i, \alpha_{i-1} + \beta_{i+1}\}$ abzuschätzen, was der Länge der Segmente, die auf jeden Fall für ein minimales Manhattan-Netzwerk gebraucht werden, entspricht.

Um in der Rekursion zu verhindern, dass die α und β Werte aufwendig aktualisiert werden müssen, werden die Werte x_{off} und y_{off} eingeführt, die den jeweiligen Offset angeben, um den die Ecke des gerade betrachteten Treppenpolygons gegenüber q_A verschoben ist. Damit wird dann jeweils $\alpha_j - x_{\text{off}}$ mit $\beta_j - y_{\text{off}}$ verglichen anstelle von α_j mit β_j in der Definition von Λ (vgl. Abbildung 3.24). Der Algorithmus 3.3 ist der Pseudo-Code für ein Treppenpolygon des Typs $\delta(q, 1)$.

Theorem 3.22 *Für eine beliebige Zusammenhangskomponente $A \in A_3$ mit $|S \cap \partial A| = m$ berechnet der Algorithmus* BRIDGE *in Zeit $O(m \log m)$ eine Menge B von Liniensegmenten mit den Eigenschaften:*

(i) $|B| \leq 2|N_{opt} \cap A|$

(ii) $\bigcup B \subset A$ *„überbrückt“* A

Beweis Theorem 3.22 Da $\alpha_1, \ldots, \alpha_{m-1}$ monoton steigend sind, lässt sich der Wert für i mittels einer binären Suche in Zeit $O(\log m)$ berechnen. Da wir nach maximal m Rekursionsschritten fertig sind, ergibt sich als Gesamtlaufzeit $O(m \log m)$. Nach Lemma 3.20

sind die Regionen A disjunkt und können deshalb unabhängig voneinander betrachtet werden.

Sei i der Index, der beim ersten Aufruf von SubBridge (siehe Algorithmus 3.3 auf Seite 60) bestimmt wird. Wenn $i > 1$ ist, dann sei A_{top} der Bereich von A, der oberhalb von a_{i-1} liegt, ansonsten sei $A_{\text{top}} = \emptyset$. Wenn $i < m - 1$ ist, dann sei A_{right} der Bereich von A, der rechts von b_{i+1} ist, ansonsten sei $A_{\text{right}} = \emptyset$. Sei nun A_{off} definiert als $A \setminus (A_{\text{top}} \cup A_{\text{right}})$. Damit gilt (vgl. Abbildung 3.24):

$$(a_{i-1} \cup b_{i+1}) \subset A_{\text{off}}$$

Per Induktion können wir annehmen, dass folgendes gilt:

$$\begin{aligned} |B(A_{\text{top}})| &\leq 2|N_{\text{opt}} \cap A_{\text{top}}| \\ |B(A_{\text{right}})| &\leq 2|N_{\text{opt}} \cap A_{\text{right}}| \end{aligned}$$

Damit reicht es zu zeigen, dass folgende Gleichung gilt:

$$\alpha_{i-1} + \beta_{i+1} \leq 2|N_{\text{opt}} \cap A_{\text{off}}| \tag{3.1}$$

Das Netzwerk N_{opt} enthält Segmente in A_{off} für einen Pfad von q nach p_i und einen von q nach p_{i+1}. Genauer gilt damit:

$$|N_{\text{opt}} \cap A_{\text{off}}| \geq \min\{\alpha_i, \beta_i, \alpha_{i-1} + \beta_{i+1}\}$$

Wenn N_{opt} ein Segment mit mindestens der Länge $\alpha_{i-1} + \beta_{i+1}$ in A_{off} enthält, ist die Gleichung 3.1 offensichtlich wahr. Damit bleibt nur noch übrig zu zeigen, dass $\alpha_{i-1} + \beta_{i+1} \leq 2\min\{\alpha_i, \beta_i\}$ gilt. Wir machen nun eine Fallunterscheidung über die Art, wie i bestimmt wurde.

$\Lambda = \emptyset, i = 1$
: In diesem Fall ist $A_{\text{top}} = \emptyset$ und $\alpha_1 > \beta_1$ und es wird nur das Segment b_2 hinzugefügt. Damit gilt:
$$\beta_2 < \beta_1 = \min\{\alpha_1, \beta_1\}$$

$i' = \max \Lambda = m - 1$
: In diesem Fall gilt ein analoges Argument wie für $\Lambda = \emptyset$.

$i' < m - 1$ **und** $\alpha_{i'} > \beta_{i'+1}$
: In diesem Fall gilt $i = i'$. Dies bedeutet, dass $\beta_{i+1} < \alpha_i$ gilt und damit auch $\alpha_{i-1} + \beta_{i+1} < 2\alpha_i$. Andererseits gilt aufgrund der Definition von Λ, auch: $\alpha_i \leq \beta_i$. Daraus folgt:
$$\alpha_{i-1} + \beta_{i+1} < 2\alpha_i = 2\min\{\alpha_i, \beta_i\}$$

$i' < m - 1$ **und** $\alpha_{i'} \leq \beta_{i'+1}$
: In diesem Fall ist $i = i' + 1$. Daraus ergibt sich $\alpha_{i-1} \leq \beta_i$ und damit auch $\alpha_{i-1} + \beta_{i+1} < 2\beta_i$. Andererseits ergibt sich aus der Definition von Λ, dass $\alpha_i > \beta_i$ gilt. Daraus folgt:
$$\alpha_{i-1} + \beta_{i+1} < 2\beta_i = 2\min\{\alpha_i, \beta_i\}$$

□

Theorem 3.23 *Der Algorithmus* ApproxMMN *läuft in Zeit* $O(n \log n)$ *und nutzt* $O(n)$ *Speicherplatz.*

Beweis Theorem 3.23 Jede der vier Phasen läuft in Zeit $O(n \log n)$.

Phase 0: Gezeigt durch Lemma 3.14 auf Seite 40.

Phase 1: Gezeigt durch Lemma 3.16 auf Seite 41 und Lemma 3.19 auf Seite 48.

Phase 2: Gezeigt durch Lemma 3.21 auf Seite 58.

Phase 3: Gezeigt durch Theorem 3.22 auf Seite 61.

Als Ausgabe hat der Algorithmus $O(n)$ Liniensegmente, da Z nur $O(n)$ Paare enthält. Für jedes Paar muss nur eine konstante Anzahl an zugehörigen Attributen gespeichert. Diese sind die Nachbarn und die dazugehörigen Rechtecke.

□

3.2.5 Der Approximationsfaktor

Um den Approximationsfaktor von drei zu beweisen, betrachten wir die Länge von N in A_{12} und A_3 separat. Aus Theorem 3.22 und Lemma 3.20 ergibt sich sofort:

$$|N \cap A_3| = |N_3| \leq 2|N_{\text{opt}} \cap A_3|$$

Damit muss nur noch gezeigt werden, dass $|N \cap A_{12}| = |N_1 \cup N_2|$ durch $3|N_{\text{opt}} \cap A_{12}|$ beschränkt ist. Aus Lemma 3.12 und Lemma 3.16 folgt $|N_1| \leq |N_{\text{opt}}| + H + W$. Die Segmente von N_{opt}, die benutzt worden sind, um die Abschätzung für Lemma 3.16 zu erreichen, liegen alle in dem Gebiet von $A_{\text{ver}} \cup A_{\text{hor}}$. Deshalb gilt sogar die schärfere Schranke $|N_1| \leq |N_{\text{opt}} \cap A_{12}| + H + W$. Nun muss noch die Länge der Segmente für N_2 abgeschätzt werden. Seien N_2^{ver} die vertikalen Segmente von N_2 und N_2^{hor} die horizontalen Segmente von N_2. Wir werden nun N_2^{ver} mit dem schönen Cover C_{ver} und N_2^{hor} mit dem schönen Cover C_{hor} vergleichen. Daraus ergibt sich dann die gewünschte Schranke. Im Folgenden werden wir dies nur für N_2^{ver} durchführen, da dies der kompliziertere Fall ist. Aufgrund der Tatsache, dass die verbindenden Segmente vertikal sind, enthält N_2^{ver} auch die verbindenden Segmente. Als erstes trennen wir die verbindenden Segmente von den anderen Segmenten in N_2^{ver}. Die nicht verbindenden Segmente von N_2^{ver} heißen *begrenzende Segmente*, da sie auf ∂A_3 liegen. Nach Lemma 3.21 schneiden sich die Segmente von N_2^{ver} und C_{ver} maximal an den Endpunkten. Damit kann eine horizontale Gerade l mit $l \cap S = \emptyset$ keinen Punkt enthalten, der sowohl in $\bigcup N_2^{\text{ver}}$, als auch in $\bigcup C_{\text{ver}}$ enthalten ist. Wir betrachten im Folgenden nur diese Geraden. Dies ändert nichts an der Längenabschätzung, da damit nur eine endliche Zahl von Punkten nicht betrachtet wird. Um die gewünschte Abschätzung zu erreichen, charakterisieren wir die Sequenz, die durch Schnitt so einer Geraden l mit Segmenten des Covers und begrenzenden Segmenten entsteht, sowie die Sequenz, die durch den Schnitt so einer Geraden l mit Segmenten des Cover und verbindenden Segmenten entsteht.

Lemma 3.24 *Sei l eine horizontale Gerade mit $l \cap S = \emptyset$ und $l \cap R(S) \neq \emptyset$. Betrachte die Sequenz von* Cover-Segmenten *und begrenzenden Segmenten, die von l geschnitten werden. Dann gilt:*

(i) Es folgen nie mehr als zwei begrenzende Segmente aufeinander.

(ii) Das Segment ganz links und das Segment ganz rechts sind Cover-Segmente.

Beweis Lemma 3.24 Wir zeigen, dass jedes begrenzende Segment s ein Cover-Segment als direkten Nachfolger oder Vorgänger hat. Daraus folgt dann sofort (i). Aus der Art des Cover-Segments, das der Vorgänger oder Nachfolger ist, wird sich folgern lassen, dass es kein begrenzendes Segment ganz links oder ganz rechts geben kann. Damit ergibt sich dann sofort (ii).

Die begrenzenden Segmente liegen jeweils auf dem Rand einer Zusammenhangskomponente von A. Ohne Beschränkung der Allgemeinheit liege A in einer Region vom Typ $\delta(q, 1)$. Die anderen Fälle sind bis auf Symmetrie äquivalent. Seien $p_1, \ldots, p_m$ die Punkte von S, die auf dem Rand von A liegen, sortiert nach dem x-Abstand von q. Wie vorher sei $v_A = p_1.\text{xnbor}[3]$. Sei nun R das Rechteck in R_{ver}, das durch q und den vertikalen Nachfolger p_0 von q definiert wird. Weiterhin sei p_l der Schnittpunkt des Segmentes s mit der horizontalen Geraden l, $p_l = s \cap l$. Damit liegt p_l in $\bigcup N_2^{\text{ver}}$, jedoch nicht in $\bigcup C_{\text{ver}}$. Nun gibt es zwei Fälle für den Typ von s:

s ist begrenzendes Segment auf der linken Seite von A: In diesem Fall muss das Segment s auf dem rechten Rand des Rechtecks R liegen. Sei $q_l = (x_q, y_l)$ der Punkt auf der linken Seite von R, der p_l direkt gegenüber liegt. Dann muss q_l in $\bigcup C_{\text{ver}}$ liegen, da p_l nicht in $\bigcup C_{\text{ver}}$ liegt, das Rechteck aber vom Cover abgedeckt werden muss. Und da es sich um ein schönes Cover handelt, muss somit q_l im Cover enthalten sein. Da $\text{int}(R) \subset \text{int}(A_{12})$ gilt, kann kein weiteres begrenzendes Segment das Innere des Segments zwischen q_l und p_l schneiden. Damit ist $q_l \in \bigcup C_{\text{ver}}$ der Vorgänger von p_l auf l.

s ist ein vertikales Segment auf der rechten Seite von A: In diesem Fall ist s ein Segment der Treppe von A und wir zeigen, dass s ein Segment aus C_{ver} als Nachfolger hat. Dann lässt sich dieser Fall in zwei weitere Fälle unterteilen: Entweder ist s die linke Seite eines umschließenden Rechtecks $R(\{p_i, p_{i+1}\})$ für ein $i \in \{1, \ldots, m-1\}$ oder s ist die linke Seite des umschließenden Rechtecks $R(\{p_m, w_A\})$ (vgl. Abbildung 3.25). Wir machen deshalb nun eine witere Fallunterscheidung.

$s \in R(\{p_i, p_{i+1}\})$

Sei $\beta = R(\{p_i, p_{i+1}\})$ (vgl. Abbildung 3.25). Wir zeigen nun, dass l ein vertikales Segment in C_{ver} schneidet und dass $\beta \cap A_3 = \emptyset$ gilt. Somit kann es kein begrenzendes Segment im Innern von β geben.

Dies geschieht, indem wir die Punktpaare (p', q') charakterisieren, die die Eigenschaft $R(\{p', q'\}) \cap \text{int}(\beta) \neq \emptyset$ haben und zeigen, dass jede Zusammenhangskomponente aus A_3, die zu p' gehört, β nicht schneidet. Seien σ und τ

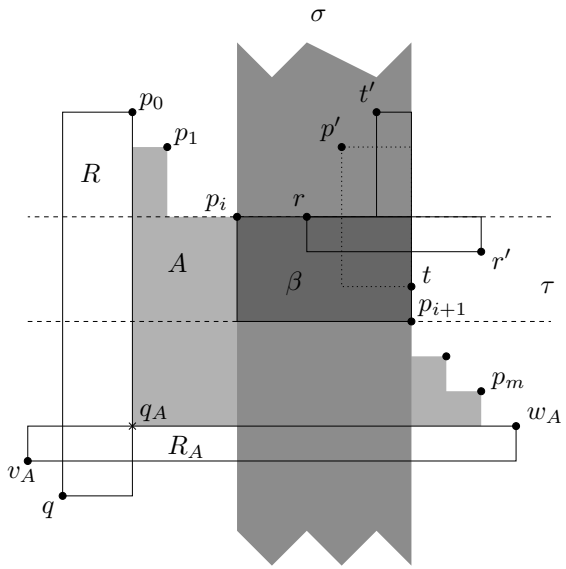


Abbildung 3.25: Notation für den Beweis des Approximations-Faktor. Der rechte Rand von β ist das begrenzende Segment, das von der horizontalen Geraden l geschnitten wurde. Der Streifen τ kann keinen Punkt aus S links von β enthalten und der Streifen σ keinen unterhalb von β. Die Punkte r und t auf dem Rand von β sind die, die am weitesten entfernt zu p_i und p_{i+1} liegen. Das Innere von β schneidet A_3 nicht, somit muss die Gerade l entweder den Rand von $R(\{t, t'\})$ oder $R(\{t, p_{i+1}\})$ schneiden.

die vertikalen und horizontalen Streifen durch β. Der Streifen τ kann keinen Punkt aus S links von β enthalten, da sonst p_i und p_{i+1} nicht in der selben Zusammenhangskomponente liegen können. Der Streifen σ kann keinen Punkt aus S unterhalb von β enthalten, da sonst das Paar (q, p_{i+1}) nicht in Z_{quad} liegen würde. Sei β' definiert als β ohne den rechten und oberen Rand. Dann kann es keinen Punkt aus S in β' geben, da der Punkt $p \in \beta'$ der Bedingung, dass p_i und p_{i+1} zwei aufeinanderfolgende Punkte sind, widersprechen würde.

Sei nun r der Punkt aus S am weitesten rechts auf der oberen Kante von β und t der oberste Punkt von S auf der rechten Kante von β (vgl. Abbildung 3.25). Es ist möglich, dass $p_i = r$ und $t = p_{i+1}$ gilt. Es muss jedoch einen Punkt $r' \in Q(r, 4)$ mit $(r', r) \in Z_{\text{hor}}$ und einen Punkt $t' \in Q(t, 2)$ mit $(t', t) \in Z_{\text{ver}}$ geben. Damit muss jedoch $q' = t$ und $p' \in Q(q', 2)$ gelten, da sonst das umschließende Rechteck $R(\{p', q'\})$ nicht $\text{int}(\beta)$ schneiden würde. Nun teilt das Rechteck $R(\{r, r'\}) \in R_{\text{ver}}$ das Rechteck $R(\{p', q'\})$ mit $q' =$ in zwei Komponenten. Damit können die Zusammenhangskomponenten aus A_3, die an p' angrenzen, β jedoch nicht schneiden und somit gilt $\beta \cap A_3 = \emptyset$.

Da das umschließende Rechteck $R(\{t, t'\})$ in R_{ver} liegt und $y_{t'} \geq y_{p_i}$ gilt, muss

l ein vertikales Cover-Segment in β schneiden. Entweder das Segment, das durch das nicht-degenerierte Rechteck $R(\{t, t'\})$ erzeugt wird, wenn $y_l \geq y_t$ gilt oder das degenerierte Rechteck $R(\{p_{i+1}, t\})$, wenn $y_l = y_t$ gilt.

$s \in R(\{p_m, w_A\})$

Sei $\beta = R(\{p_m, w_A\})$. In diesem Fall können wir nicht annehmen, dass es

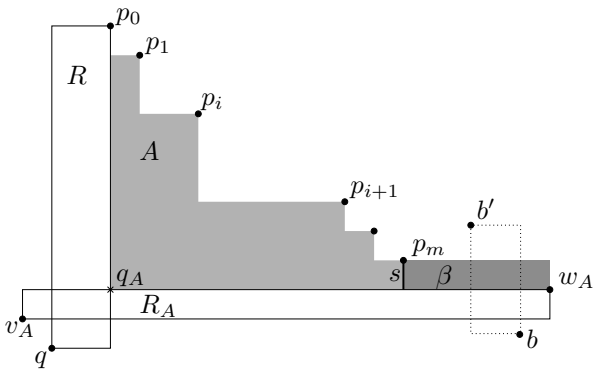


Abbildung 3.26: Das begrenzende Segment s entspricht der untersten Treppenstufe. b ist p_m.xnbor[4]. Wenn b unterhalb von w_A liegt, so schneidet das Rechteck $R(\{b, b'\})$ das Rechteck β rechts von s.

einen Punkt aus s unterhalb von β gibt. Sei $b = p_m$.xnbor[4], dabei ist $b = w_A$ erlaubt. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass $y_{p_m} > y_b$ gilt. Sollte dies nicht der Fall sein, so sei $b = b$.xnbor[4], bis diese Bedingung erfüllt ist. Nun kann b nur in $\text{int}(\beta)$ liegen, wenn es einen Punkt b' mit $x_b = x_{b'}$ und $y_{b'} < y_{w_A}$ gibt. Sonst würde es einen Punkt $p \in \text{int}(\beta)$ geben mit $(p, q) \in Z_{\text{quad}}$, was nicht sein kann. Wir betrachten nun aber zuerst den Fall, dass b nicht in $\text{int}(\beta)$ liegt und somit $y_b < y_{w_a}$ gilt (vgl. Abbildung 3.26). Nun gibt es einen Punkt $p' \in Q(b, 2)$ mit $(p', b) \in Z_{\text{ver}}$. Aufgrund der Konstruktion muss $y_{p'} \geq y_{p_m}$ gelten und somit teilt die vertikale Gerade durch b das Rechteck β in zwei Zusammenhangskomponenten. Für die Komponente β', die an p_m angrenzt, können wir mit dem gleichen Argument wie oben zeigen, dass $\beta' \cap A_3 = \emptyset$ gilt, da der vertikale Streifen durch β' durch diese Konstruktion keinen Punkt aus S enthält. Damit hat s ein vertikales Cover-Segment des Rechtecks $R(\{p', b\})$ als Nachfolger. Nun betrachten wir den Fall, dass b im Innern von β liegt. Wenn $y_l < y_b$ gilt, dann hat s als Nachfolger das degenerierte Rechteck $R(\{b, b'\})$, ansonsten gilt das gleiche Argument mit $p' \in Q(b, 2)$ und $(p', b) \in Z_{\text{ver}}$.

□

Als nächstes werden die verbindenden Segmente mit C_{ver} verglichen. Aufgrund der Konstruktion können diese nur im Innern von nicht degenerierten Rechtecken aus R_{hor} liegen.

Lemma 3.25 *Sei l eine horizontale Gerade, die das Innere eines Rechtecks $R_l \in R_{hor}$ schneidet. Betrachte die Sequenz von verbindenden und Cover-Segmenten in R_l. Dann gelten folgende Aussagen:*

(i) Kein verbindendes Segment liegt auf dem vertikalen Rand von R_l.

(ii) Maximal zwei verbindende Segmente können aufeinanderfolgen.

(iii) Mindestens eins der beiden Segmente ganz links ist ein Cover-Segment.

(iv) Mindestens eins der beiden Segmente ganz rechts ist ein Cover-Segment.

(v) Das Segment ganz links oder das Segment ganz rechts ist ein Cover-Segment.

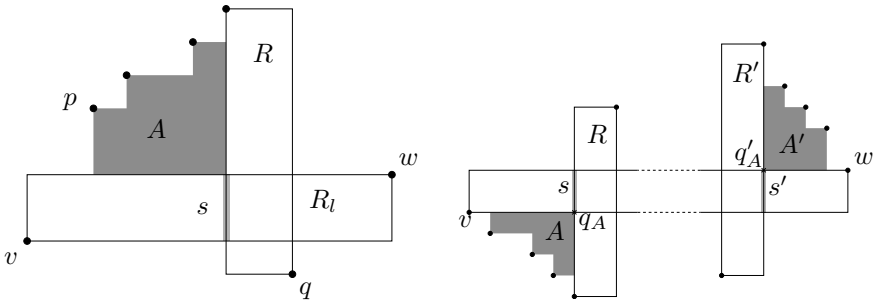
Beweis Lemma 3.25 Um (i) zu beweisen, müssen wir nur zeigen, dass kein verbindendes Segment an einen Punkt aus S angrenzen kann. Dies reicht, da die verbindenden Segmente über die komplette Höhe des Rechtecks gehen. Sollte ein verbindendes Segment auf dem Rand liegen, muss es auch an einen Punkt aus S angrenzen. Jedes verbindende Segment $s_A = \text{Seg}[q_A, c_A]$ liegt aufgrund der Konstruktion auf einer vertikalen Kante eines Rechtecks $R = R(\{q, p_0\}) \in R_{\text{ver}}$ und in einem Rechteck $R_A = R(\{v_A, w_A\}) \in R_{\text{hor}}$. Das Rechteck R kann nicht degeneriert sein, ansonsten wäre $s_A \in \bigcup C_{\text{ver}}$. Damit gilt $c_A \neq q$. Ebenfalls gilt offensichtlich $q_A \neq q$. Würde nun jedoch $\{c_A, q_A\} \cap (S \setminus \{q\}) \neq \emptyset$ gelten, würde dies der Tatsache widersprechen, dass alle Punktpaare (p, q) mit $p \in \partial A \cap S$ in Z_{quad} liegen. Also grenzt s_A an keinen Punkt aus S an.

Da das verbindende Segment s_A nicht in C_{ver}, jedoch auf dem vertikalen Rand eines Rechtecks $R \in R_{\text{ver}}$ liegt, muss es als Nachfolger oder Vorgänger das Cover-Segment auf der gegenüberliegenden Seite von R haben. Daraus folgt dann sofort (ii), (iii) und (iv).

Der Beweis für (v) geht über einen Widerspruch. Wir nehmen an, dass das Segment s ganz links und das Segment s' ganz rechts in R_l verbindende Segmente sind. Sei R_l durch die Punkte v und w definiert: $R_l = R(\{v, w\})$. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass v die linke untere Ecke und w die rechte obere Ecke des Rechtecks ist. Die anderen Fällen sind symmetrisch. Seien A und A' die Zusammenhangskomponenten von A_3 mit $s = s_A$ und $s' = s'_A$. Es muss $R_A = R_{A'} = R_l$ gelten, da in diesem Rechteck die beiden verbindenden Segmente liegen. Seien R und R' die Rechtecke in R_{ver}, deren vertikale Kanten s und s' enthalten. Dann muss s auf der linken Seite von R und s' auf der rechten Seite von R' liegen. Damit muss A eine Region vom Typ $\delta(q, 2)$ oder $\delta(q, 3)$ sein.

Als Erstes nehmen wir an, dass $A \subseteq \delta(q, 2)$ für ein $q \in S$ gilt. Dann würde A oberhalb von R liegen und q unterhalb von R (vgl. Abbildung 3.27(a)). Dies ist jedoch unmöglich. Sei p der Punkt ganz links von $P(q, t) \cap \partial A$. Dann muss p einen Z_{hor} Partner in $Q(p, 4)$ haben. Dies widerspricht jedoch der Tatsache, dass (p, q) in Z_{quad} liegt.

Also bleibt noch der Fall, dass $A \subseteq \delta(q, 3)$ gilt und damit $A' \subseteq \delta(q', 1)$ für $q, q' \in S$. Nun hat jedoch der Manhattan-Pfad von v nach w mindestens einen Eckpunkt in q_A oder q'_A. Damit können jedoch nicht sowohl s als auch s' verbindende Segmente sein, weil mindestens ein Segment in N_1 liegen muss (vgl. Abbildung 3.27(b)).



(a) Das Rechteck $R(\{w, p\}) \in Z_{\text{hor}}$ verhindert, dass $R(\{p, q\})$ in R_{quad} liegt, wo es aber nach Voraussetzung des Beweises von Lemma 3.25 liegt.

(b) Die Segmente s und s' können nicht beide in der Menge N_2 liegen, da der Manhattan-Pfad von v nach w entweder durch q_A oder durch q'_A geht. Ein verbindendes Segment zu q_A bzw. q'_A wird jedoch nur eingefügt, wenn kein Pfad aus N_1 über q_A bzw. q'_A geht.

Abbildung 3.27: Unmögliche Fälle beim Beweis von Lemma 3.25

□

Mit Hilfe von Lemma 3.24 und Lemma 3.25 lässt sich nun folgendes Lemma beweisen:

Lemma 3.26 *Sei N_2^{ver} die Menge der vertikalen Kanten in N_2 und N_2^{hor} die horizontalen. Seien C_{ver} und C_{hor} minimale, schöne Cover und H die Höhe von $R(S)$ und W die Breite von $R(S)$. Dann gilt:*

$$\begin{aligned} |N_2^{ver}| &\leq 2|C_{ver}| - H \\ |N_2^{hor}| &\leq 2|C_{hor}| - W \end{aligned}$$

Beweis Lemma 3.26 Für eine horizontale Gerade l mit $l \cap S = \emptyset$ werden wir die Anzahl der Segmente in N_2^{ver} mit der Anzahl der Segmente in C_{ver} vergleichen, die von l geschnitten werden. Bezeichne im Folgenden $\#N$ die Anzahl der Segmente in der Menge N. Wenn es gelingt zu zeigen, dass $\#N_2^{\text{ver}} \leq 2\#C_{\text{ver}} - 1$ gilt, so folgt daraus, dass es über die Höhe des gesamten Rechtecks $R(S)$ gesehen maximal ein Segment in N_2^{ver} mehr gibt und somit folgt daraus dann die zu zeigende Abschätzung $|N_2^{\text{ver}}| \leq 2|C_{\text{ver}}| - H$. Die endliche Anzahl an horizontalen Geraden, für die $l \cap S \neq \emptyset$ gilt, ändert an der Länge nichts, da es sich nur um eine endliche Zahl von Geraden handelt. Nach Lemma 3.25 (i) schneidet l verbindende Segmente nur im Innern von Rechtecken aus R_{hor}. Andererseits kann l aufgrund der Definition von A_3 keine begrenzenden Segmente im Innern von solchen Rechtecken schneiden. Wir betrachten nun drei Fälle:

l **schneidet kein verbindendes Segment:** In diesem Fall werden nur begrenzende und Cover-Segmente geschnitten. Nach Lemma 3.24 folgen maximal zwei begrenzende Segmente aufeinander und sowohl das Segment ganz links, als auch das Segment

ganz rechts sind Cover-Segmente. Durch einfaches Aufsummieren ergibt sich für diesen Fall sogar:

$$\#N_2^{\text{ver}} \leq 2\#C_{\text{ver}} - 2$$

l **schneidet kein begrenzendes Segment:** Nach Lemma 3.25 (ii) und (v) folgen maximal zwei begrenzende Segmente aufeinander und das Segment ganz links oder ganz rechts muss ein Cover-Segment sein. Da nach 3.25 (iii) und (iv) sowohl von den beiden Segmenten ganz links, als auch von den beiden Segmenten ganz rechts jeweils eins ein Cover-Segment sein muss, ergibt sich durch Aufsummieren das gewünschte:

$$\#N_2^{\text{ver}} \leq 2\#C_{\text{ver}} - 1$$

l **schneidet begrenzende und verbindende Segmente:** Aufgrund von Lemma 3.24 (ii) und Lemma 3.25 (v) muss entweder das Segment ganz links oder das Segment ganz rechts ein Cover-Segment sein. Sollten nun in der Sequenz der von l geschnittenen Segmente maximal zwei Segmente aus N_2^{ver} aufeinanderfolgen, ergibt sich die gleiche Situation wie im vorhergehenden Fall und als Summe:

$$\#N_2^{\text{ver}} \leq 2\#C_{\text{ver}} - 1$$

Allerdings können auch mehr als zwei Segmente aus N_2^{ver} aufeinanderfolgen. Dies ist der Fall, wenn zwei begrenzende Segmente als Nachfolger oder Vorgänger ein Rechteck aus $R \in R_{\text{hor}}$ haben. Aufgrund von Lemma 3.25 (iii) und (iv) kann maximal eins der beiden folgenden Segmente in R ein verbindendes Segment sein. Damit können jedoch nicht mehr als drei Segmente aus N_2^{ver} aufeinanderfolgen. Sollten drei Segmente aus N_2^{ver} aufeinanderfolgen, dann muss entweder das Segment ganz links oder das Segment ganz rechts ein verbindendes Segment sein. Ohne Beschränkung der Allgemeinheit sei dies das Segment ganz links. Nach Lemma 3.25 (v) muss dann das Segment ganz rechts von R ein Cover-Segment sein. Daraus lassen sich zwei Eigenschaften ableiten:

(a) Da l maximal ein Rechteck in R_{hor} schneidet, können nur einmal drei Segmente aus N_2^{ver} aufeinanderfolgen.

(b) Wenn drei solche Segmente aufeinanderfolgen, dann muss nach Lemma 3.24 (ii) sowohl das Segment ganz links, als auch das Segment ganz rechts ein Cover-Segment sein.

Diese beiden Eigenschaften kombiniert ergeben dann wieder die gewünscht Schranke:

$$\#N_2^{\text{ver}} \leq 2\#C_{\text{ver}} - 1$$

Mit den gleichen Argumenten lässt sich die Länge von N_2^{hor} beschränken, was sogar noch einfacher ist, da N_2^{hor} keine verbindenden Segmente enthält.

□

Mit Hilfe dieses Lemmas lässt sich nun der Approximationsfaktor des Algorithmus ApproxMMN zeigen:

Theorem 3.27 *Der Algorithmus* ApproxMMN *berechnet ein Netzwerk N, für das gilt:*

$$|N| \leq 3|N_{opt}|$$

Beweis Theorem 3.27 Nach Lemma 3.26 gilt:

$$|N_2| \leq 2|C_{\text{ver}} \cup C_{\text{hor}}| - H - W \tag{3.2}$$

Wie in Lemma 3.12 auf Seite 38 gezeigt wurde, muss ein minimales Manhattan-Netzwerk mindestens so groß wie das Cover sein. Außerdem muss der Anteil von N_{opt}, der für diese Abschätzung genutzt wurde, in A_{12} liegen. Damit gilt:

$$|C_{\text{ver}} \cup C_{\text{hor}}| \leq |N_{\text{opt}} \cap A_{12}| \tag{3.3}$$

Aus den Gleichungen 3.2 und 3.3 folgt als Abschätzung für N_2:

$$|N_2| \leq 2|N_{\text{opt}} \cap A_{12}| - H - W \tag{3.4}$$

Für N_1 gilt die Abschätzung:

$$|N_1| \leq |N_{\text{opt}} \cap A_{12}| + H + W \tag{3.5}$$

Aus den Gleichungen 3.4 und 3.5 ergibt sich als Abschätzung für $N_1 \cup N_1$:

$$|N_1 \cup N_2| \leq 3|N_{\text{opt}} \cap A_{12}| \tag{3.6}$$

Aus Theorem 3.22 und Lemma 3.20 ergibt sich:

$$|N_3| \leq 2|N_{\text{opt}} \cap A_3| \tag{3.7}$$

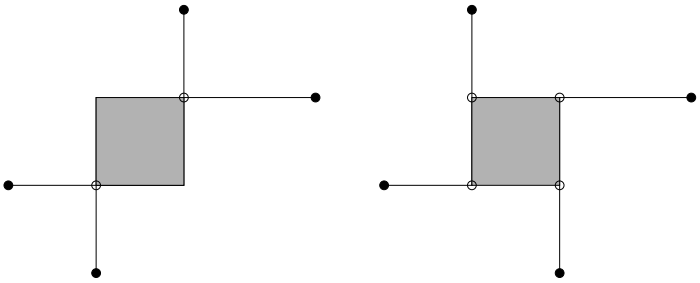
Da die Gebiete A_{12} und A_3 disjunkt sind, ergibt sich aus den Gleichungen 3.6 und 3.7:

$$|N| \leq 3|N_{\text{opt}}| \tag{3.8}$$

□

3.3 Ein Rundungs-Algorithmus für minimale Manhattan-Netzwerke

Hier wird der Algorithmus von Chepoi, Nouioua und Vaxès aus „A rounding algorithm for approximating minimum Manhattan networks“ [CNV05] vorgestellt. Der Algorithmus basiert auf Formulierung des minimalen Manhattan-Netzwerk Problems als Lineares Programm. Er erreicht einen Approximationsfaktor von zwei bei einer polynomiellen Laufzeit.



(a) Wenn die vier Punkte so liegen, hat die Pareto-Hülle zwei verschiedene Trennpunkte. Bei einer Kontur wäre die gefüllte Fläche nicht enthalten, sondern es wäre stattdessen ein Manhattan-Pfad enthalten, der die beiden Trennpunkte verbindet.

(b) Wenn die vier Punkte so liegen, hat die Pareto-Hülle vier verschiedene Trennpunkte. Bei dieser Lage der Punkte entspricht die Pareto-Hülle auch einer Kontur.

Abbildung 3.28: Die Pareto-Hülle von vier Punkten. Die gefüllten Punkte sind dabei die Punkte aus S, während die leeren Punkte die Trennpunkte des schattierten Blocks sind.

3.3.1 Eigenschaften und die LP-Formulierung

Ein Punkt $t \in \mathbb{R}^2$ heißt *effizienter Punkt* zu S, wenn es keinen anderen Punkt $t' \in \mathbb{R}^2$ gibt, mit folgenden Eigenschaften:

$$\forall p_i \in S : \exists p_j \in S : d(t', p_i) \leq d(t, p_i) \wedge d(t', p_j) < d(t, p_j)$$

Dies bedeutet, ein Punkt ist genau dann effizient, wenn es keinen anderen Punkt in der Ebene gibt, der zu allen Punkten maximal so weit entfernt ist, wie der effiziente Punkt und zu mindestens einem Punkt sogar näher dran liegt. Die Menge aller effizienten Punkte ist $\mathcal{P}$, die *Pareto-Hülle* (vgl. Abbildung 3.28).

Ein Punkt p wird von einem Punkt q *dominiert*, wenn gilt:

$$\forall p_i \in S : \exists p_j \in S : d(q, p_i) \leq d(p, p_i) \wedge d(q, p_j) < d(p, p_j)$$

q ist also an allen Punkten maximal genausoweit entfernt wie p und an mindestens einem Punkt näher dran. Damit lassen sich die effizienten Punkte auch definieren, als die Punkte, die von keinem anderen Punkt dominiert werden.

Einen optimalem $O(n \log n)$ Algorithmus, um die Pareto-Hülle von n Punkten in der Manhattan-Metrik zu berechnen, haben Chalmet, Francis und Kolen [CFK81] vorgestellt. Eigenschaften der Pareto-Hülle und ein $O(n^2)$ Algorithmus haben Wendell, Hurter und Lowe [WHL73] vorgestellt. Im Besonderen ist es wichtig, dass $\mathcal{P}$ *ortho-konvex* ist. Dies bedeutet, dass jeder Schnitt von $\mathcal{P}$ mit einer vertikalen oder horizontalen Geraden zusammenhängt ist und je zwei Punkte aus $\mathcal{P}$ können in $\mathcal{P}$ mit einem Manhattan-Pfad

verbunden werden. Damit ergibt sich $\mathcal{P}$ als eine Vereinigung von – möglicherweise degenerierten – rechtwinkligen Polygonen, den so genannten Blöcken, verbunden entlang der Ecken, den so genannten *Trennpunkten*. Die Pareto-Hülle hat starke Ähnlichkeiten zu der Kontur aus Kapitel 2.1 auf Seite 14, ist allerdings größer als die Kontur. Bei einer Kontur lassen sich manche Blöcke, wie beispielsweise Blöcke, die nur aus zwei Trennpunkten bestehen, durch einen Manhattan-Pfad ersetzen (vgl. Abbildung 3.28(a)).

Lemma 3.28 *Die Pareto-Hülle $\mathcal{P}$ der Punktmenge S enthält mindestens ein minimales Manhattan-Netzwerk für S.*

Beweis Lemma 3.28 Dieser Beweis geht analog zum Beweis von Lemma 2.7 auf Seite 17 und Lemma 2.8 auf Seite 19, die zeigen, dass es ein minimales Manhattan-Netzwerk innerhalb der Kontur gibt.

□

Damit reicht es, um das minimale Manhattan-Netzwerk Problem für S zu lösen, die Trennpunkte zu S hinzuzunehmen. Von dieser neuen Menge S' wird das Manhattan-Netzwerk Problem für jeden Block von $\mathcal{P}$ gelöst. Die die einzelnen Blöcke über eindeutige Manhattan-Pfade zwischen den Trennpunkten verbunden sind, müssen Pfade zwischen zwei Punkten aus S in verschiedenen Blöcken auf jeden Fall über diese Trennpunkte gehen. Deshalb reicht es aus, wenn für einen Block das minimale Manhattan-Netzwerk von den Punkten aus S' berechnet wird, die in diesem Block liegen.

Aufgrund der Zerlegung des Problems in kleinere Probleme, können wir nun ohne Beschränkung der Allgemeinheit annehmen, dass $\mathcal{P}$ nur aus einem einzigen Block mit mindestens drei Punkten besteht. Sei $\partial\mathcal{P}$ der Rand dieses ortho-konvexen, rechtwinkligen Polygons.

Aus Lemma 2.12 auf Seite 20 und Lemma 3.28 können wir folgern, dass es ein minimales Manhattan-Netzwerk auf dem Gitter Γ in $\mathcal{P}$ gibt. Zwei Kanten von Γ heißen Zwillinge, wenn sie gegenüberliegende Kanten einer rechtwinkligen Fläche von Γ sind. Zwei Kanten e und f heißen parallel, wenn es eine Folge $e = e_1, e_2, \ldots, e_{m+1} = f$ gibt, so dass für $1 \leq i \leq m$ die Kanten e_i, e_{i+1} Zwillinge sind. Aus der Definition ergibt sich, dass jede Kante e parallel zu sich selber ist und dass alle Kanten, die parallel zu e sind, die selbe Länge haben. Aufgrund der Ortho-Konvexität liegen genau zwei Kanten, die zu e parallel sind, auf $\partial\mathcal{P}$.

Lemma 3.29 *Sei $\mathcal{P}$ eine Pareto-Hülle, die nur aus einem einzelnen Block besteht. Dann gelten folgende beiden Eigenschaften:*

(i) Jede konvexe Ecke von $\mathcal{P}$ entspricht einem Punkt aus S.

(ii) Der Pfad auf $\partial\mathcal{P}$ zwischen zwei aufeinanderfolgenden konvexen Ecken p_i, p_j auf $\partial\mathcal{P}$ ist der eindeutige Manhattan-Pfad zwischen p_i und p_j in $\mathcal{P}$.

Beweis Lemma 3.29 Der Beweis für (i) geht über Widerspruch (vgl. Abbildung 3.29). Angenommen, es gibt eine konvexe Ecke u, die keinem Punkt aus S entspricht. Dann hat u genau zwei Nachbarn im Gitter Γ. Ohne Beschränkung der Allgemeinheit sei u eine Ecke, die lokal links oben von $\mathcal{P}$ liegt. Sei nun v der Knoten rechts von u und w der unterhalb von u. Sei nun $z = (x_v, y_w)$. Da u eine konvexe Ecke von $\mathcal{P}$ ist, ist die Fläche $uvzw$ des Gitters komplett in $\mathcal{P}$ enthalten. Wähle nun ein δ mit $0 < \delta \leq \min\{d(u,v), d(u,w)\}$ und sei u' definiert als $u' = (x_u - \delta, y_u - \delta)$. Nun teilen wir S in drei Teilmengen. S_z enthält die Punkte $p \in S$, die mit einem Manhattan-Pfad über z mit u verbunden werden können. S_v enthält die Punkte $p \in S \setminus S_z$, die mit einem Manhattan-Pfad über v mit u verbunden werden können. S_w enthält die Punkte $p \in S \setminus S_z$, die mit einem Manhattan-Pfad über w mit u verbunden werden können. Der Punkt u' hat den gleichen Abstand wie u zu jedem Punkt aus $S_v \cup S_w$ und einen geringeren Abstand zu den Punkten aus S_z. Da u ein effizienter Punkt ist, muss S_z leer sein. Damit muss jeder Punkt aus S entweder oberhalb und rechts von v oder unterhalb und links von w liegen.

In diesem Fall kann einfach gezeigt werden, dass jeder Punkt p im offenen Quadranten $\{(x_p, y_p) : x_p > x_w, y_p < y_w\}$ von einem Punkt $p' = (x_p - \alpha, y_p - \alpha)$ mit $\alpha = \min\{x_p - x_w, y_w - y_y\}$ dominiert wird. Analog lässt sich zeigen, dass jeder Punkt in den drei offenen Quadranten

$$(x_q, x_q) : \begin{cases} x_q > x_v\,,\, y_q < y_v \\ x_q < x_w,\, y_q > y_w \\ x_q < x_v\,,\, y_q > y_v \end{cases}$$

dominiert wird. Damit zerfällt $\mathcal{P}$ jedoch in mehrere Blöcke. Dabei bildet die Fläche $uvzw$ einen Block und es gibt einen oder mehrere Blöcke oberhalb und rechts von v, sowie einen oder mehrere Blöcke unterhalb und links von w. Dies ist jedoch ein Widerspruch zu der Annahme, dass $\mathcal{P}$ aus einem Block besteht.

Seien p und q zwei aufeinanderfolgende, konvexe Ecken. Sei $P(p,q)$ der Pfad zwischen p und q entlang $\partial\mathcal{P}$. Der Pfad zwischen p und q besteht entweder aus genau einem horizontalen oder vertikalen Liniensegment oder aus zwei Liniensegmenten. Wenn der Pfad aus einem Liniensegment besteht, dann gilt $P(p,q) = R(\{p,q\})$ und wir sind fertig. Im anderen Fall entsprechen die Segmente den Kanten von $R(\{p,q\})$ und es gilt $P(p,q) \cap R(\{p,q\}) = P(p,q)$, da es sich um konvexe Ecken handelt. Damit ist der Pfad $P(p,q)$ der eindeutige Manhattan-Pfad zwischen p und q innerhalb der Pareto-Hülle $\mathcal{P}$.

□

Weiterhin werden erzeugende Mengen benutzt, wie sie in Definition 2.13 auf Seite 22 definiert wurden. Im Folgenden werden die Paare der erzeugenden Menge Z einfach über die Indizes der Punkte von S bezeichnet. Somit entspricht ij dem Paar (p_i, p_j). Im nächsten Abschnitt werden wir eine dünne, erzeugende Menge beschreiben, die in $Z_\oslash$ enthalten ist.

Um eine LP-Formulierung des minimalen Manhattan-Netzwerk Problems zu erhalten, sei nun Z eine beliebige erzeugende Menge. Für jedes Paar $ij \in Z$ sei Γ_{ij} als $\Gamma \cap R(\{p_i, p_j\})$ definiert und das Gitter als Graph beschrieben:

$$\Gamma_{ij} = (V_{ij}, E_{ij})$$

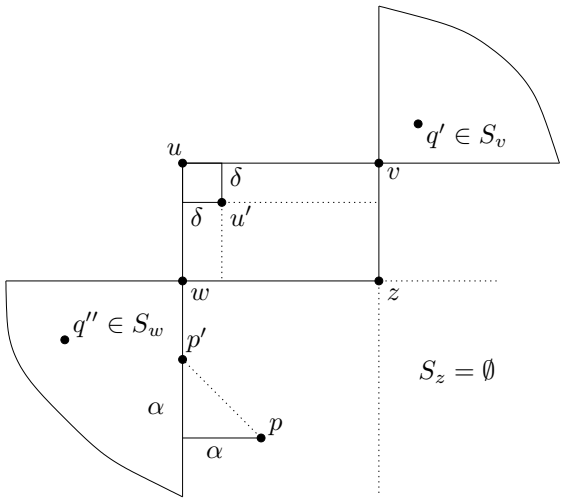


Abbildung 3.29: Konstruktion für den Beweis von Lemma 3.29. Wenn u kein Punkt aus s ist, muss S_z leer sein. Sonst würde u von u' dominiert. Wenn S_z leer ist, wird aber jeder Punkt p von einem Punkt p' dominiert, wodurch S in mindestens drei Blöcke zerfällt. Diese sind $uvwz$ und jeweils ein oder mehrere Blöcke in S_v und S_w.

Wir formulieren das minimale Manhattan-Netzwerk-Problem nun als Cut-Covering Problem unter Benutzung einer exponentiellen Anzahl an Bedingungen. Dies wird dann in eine äquivalente Bedingung umgeformt, die nur eine polynomielle Anzahl an Variablen und Bedingungen enthält. In beiden Formulierungen ist $|e|$ die Länge der Kante e aus dem Netzwerk $\Gamma = (V, E)$. x_e ist jeweils eine $\{0, 1\}$ Entscheidungsvariable, assoziiert mit der Kante e. Eine Teilmenge C von E_{ij} wird (p_i, p_j)-Schnitt genannt, wenn jeder Manhattan-Pfad zwischen p_i und p_j in Γ_{ij} den Schnitt C trifft. Sei $\mathcal{C}_{ij}$ die Vereinigung aller (p_i, p_j)-Schnitte und $\mathcal{C}$ die Vereinigung aller $\mathcal{C}_{ij}$:

$$\mathcal{C} = \bigcup_{ij \in Z} \mathcal{C}_{ij}$$

Dann kann das minimale Manhattan-Netzwerk Problem als die optimale Lösung des folgenden Integer Linear Programm gesehen werden:

$$\begin{array}{lll} \text{minimiere} & \sum_{e \in E} |e| x_e & \\ \text{unter Berücksichtigung} & \sum_{e \in C} x_e \geq 1, & C \in \mathcal{C} \\ & x_e \in \{0, 1\}, & e \in E \end{array} \tag{3.9}$$

Jedes Manhattan-Netzwerk muss eine gültige Lösung für (3.9) sein. Betrachte nun x_e als Kapazität der Kanten e von Γ und wende die Bedingungen des Cut-Cover sowie das Ford-Fulkerson Theorem auf jedes Netzwerk $\Gamma_{ij}, ij \in Z$, gerichtet wie unten angegeben,

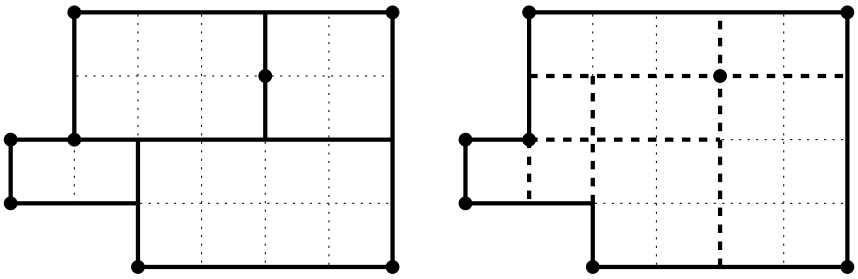
an. Dieses Theorem besagt, dass der maximale Fluss gleich der minimalen Kapazität des Cut ist. Es trägt auch den Namen „Max-Flow Min-Cut Theorem“. Dann können wir die Existenz eines Integer (p_i, p_j)-Flusses von eins folgern, die Existenz eines Manhattan-Pfades zwischen p_i und p_j. Als Konsequenz erhalten wir ein Manhattan-Netzwerk mit den gleichen Kosten. Diese Beobachtung führt zur zweiten Integer Programming Formulierung für das MMN-Problem, aber diesmal nur von polynomieller Größe. Für jedes Paar $ij \in Z$ und jede Kante $e \in E_{ij}$ führe eine Flussvariable f_e^{ij} ein. Orientiere die Kanten von Γ_{ij} so, dass die gerichteten Pfade, die p_i und p_j miteinander verbinden, genau den Manhattan-Pfaden zwischen diesen beiden Punkten entsprechen. Für einen Gitterpunkt $v \in V_{ij} \setminus \{p_i, p_j\}$ sei $\Gamma_{ij}^+(v)$ die in v eingehenden, gerichteten Kanten von Γ_{ij} und $\Gamma_{ij}^-(v)$ die von v ausgehenden, gerichteten Kanten von Γ_{ij}. Das führt zu folgendem Integer Programm:

$$
\begin{array}{lll}
\text{minimiere} & \sum_{e \in E} |e| x_e & \\
\text{unter Berücksichtigung} & \sum_{e \in \Gamma_{ij}^+(v)} f_e^{ij} = \sum_{e \in \Gamma_{ij}^-(v)} f_e^{ij} & ij \in Z, v \in V_{ij} \setminus \{p_i, p_j\} \\
& \sum_{e \in \Gamma_{ij}^-(p_i)} f_e^{ij} = 1 & ij \in Z \\
& 0 \leq f_e^{ij} \leq x_e & ij \in Z, \forall e \in E_{ij} \\
& x_e \in \{0, 1\} & e \in E
\end{array}
\tag{3.10}
$$

Seien nun (3.9') und (3.10') die LP-Relaxation von (3.9) und (3.10), die man erhält, indem die boolesche Bedingung $x_e \in \{0, 1\}$ durch die lineare Bedingung $x_e \geq 0$ ersetzt wird. Da (3.10') nur eine polynomielle Anzahl an Variablen und Ungleichungen enthält, kann es in streng polynomieller Zeit gelöst werden, unter Benutzung des Algorithmus von Tardos [Tar86]. Der x-Teil einer beliebigen, optimalen Lösung von (3.10') ist gleichzeitig auch eine optimale Lösung für (3.9'). Allerdings gibt es Instanzen des MMN-Problems, für die die Kosten einer optimalen, gebrochenen Lösung von (3.9') oder (3.10') kleiner sind als die Kosten einer optimalen, ganzzahligen Lösung von (3.9) oder (3.10) (vgl. Abbildung 3.30). In jeder gültigen Lösung für (3.9') und (3.10') gilt $x_e = 1$ für alle Kanten $e \in \partial\mathcal{P}$, da nach Lemma 3.29 der Pfad zwischen zwei konvexen Randpunkten eindeutig ist.

3.3.2 Streifen und Treppen

Ein degeneriertes Rechteck $R_{ij} = R(\{p_i, p_j\})$ wird *degenerierter horizontaler oder vertikaler Streifen* genannt. Ein nicht degeneriertes Rechteck R_{ij} wird vertikaler Streifen genannt, wenn die x-Koordinaten von p_i und p_j in der sortierten Liste der x-Koordinaten direkt aufeinanderfolgen und der Schnitt von R_{ij} mit jedem degenerierten, vertikalen Streifen entweder leer ist, oder nur den Punkt p_i oder p_j enthält. Analog heißt ein nicht degeneriertes Rechteck horizontaler Streifen, wenn die y-Koordinaten von p_i und p_j in den sortierten y-Koordinaten direkt aufeinanderfolgen und der Schnitt von R_{ij} mit den



(a) Ganzzahlige Lösung der Größe 28, die einem minimalen Manhattan-Netzwerk entspricht. Dabei gilt für alle Kanten $x_e = 1$.

(b) Gebrochen-rationale Lösung der Größe 27,5. Dabei werden die gestrichelten Kanten nur zur Hälfte genutzt ($x_e = \frac{1}{2}$). Für die anderen Kanten gilt $x_e = 1$.

Abbildung 3.30: Ganzzahl-Lücke

degenerierten horizontalen Rechtecken entweder leer ist, oder nur den Punkt p_i oder p_j enthält. Die Seiten eines vertikalen Streifens R_{ij} sind die vertikalen Kanten des Rechtecks R_{ij}. Analog sind die Seiten eines horizontalen Streifen die horizontalen Kanten des entsprechenden Rechtecks. Die Streifen $R_{ii'}$ und $R_{jj'}$ bilden eine Kreuzungs-Konfiguration, wenn sie sich schneiden und der entstehende Block der Pareto-Hülle von $\{p_i, p_{i'}, p_j, p_{j'}\}$ nur zwei Trennpunkte hat (vgl. Abbildung 3.28(a)). Das Interessante an der Konfiguration ist, dass sich damit folgendes Lemma sofort zeigen lässt:

Lemma 3.30 *Wenn die Rechtecke $R_{ii'}$ und $R_{jj'}$ eine Kreuzungs-Konfiguration bilden, dann lassen sich aus dem Manhattan-Pfad von p_i nach $p_{i'}$ und dem Pfad von p_j nach $p_{j'}$ Manhattan-Pfade von p_i nach $p_{j'}$ und von p_j nach $p_{i'}$ konstruieren.*

Beweis Lemma 3.30 Das umschließende Rechteck $R(\{p_i, p_{i'}\})$ und das umschließende Rechteck $R(\{p_j, p_{j'}\})$ schneiden sich nur im Rechteck $R(\{o, o'\})$ (vgl. Abbildung 3.31). Damit muss sich auch der Pfad von p_i nach $p_{i'}$ mit dem Pfad von p_j nach $p_{j'}$ im Reckteck $R(\{o, o'\})$ schneiden. Da aufgrund der Lage der Punkte bei beiden Pfaden die x- sowie die y-Monotonie gleich sind, gibt es auch einen Pfad von p_i nach $p_{j'}$ und einen von p_j nach $p_{i'}$.

□

Für eine Kreuzungs-Konfiguration des vertikalen Streifens $R_{ii'}$ und des horizontalen Streifens $R_{jj'}$ seien o und o' die Trennpunkte des rechteckigen Blocks der Pareto-Hülle von $\{p_i, p_{i'}, p_j, p_{j'}\}$. Dabei verbinde o den Block mit p_i und p_j, sowie o' den Block mit $p_{i'}$ und $p_{j'}$. Weiterhin können wir ohne Beschränkung der Allgemeinheit annehmen, dass p_i und p_j zu $Q(o, 1)$ gehören und $p_{i'}$ und $p_{j'}$ zu $Q(o', 3)$. Alle anderen Fälle sind symmetrisch zu diesem Fall. Sei S_{ij} die Menge von Punkten p_k mit folgenden Eigenschaften (vgl. Abbildung 3.32):

(i) $p_k \in (S \setminus \{p_i, p_j\}) \cap Q(o, 1)$ (p_k liegt im ersten Quadranten bezüglich o).

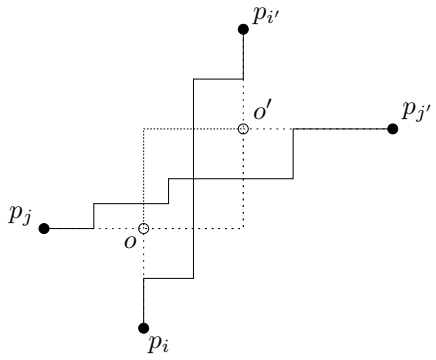


Abbildung 3.31: Konstruktion für den Beweis von Lemma 3.30. Der Pfad von p_i nach $p_{i'}$ und der Pfad von p_j nach $p_{j'}$ schneiden sich im Rechteck $R(\{o, o'\})$. Damit muss es auch einen Pfad von p_i nach $p_{j'}$ und von p_j nach $p_{i'}$ geben.

(ii) $R(\{p_k, o\}) \cap S = \{p_k\}$ (Das Rechteck von p_k und o enthält keine weiteren Punkte).

(iii) Das Gebiet $\{q \in Q(o, 2) : y_q \leq y_{p_k}\} \cup \{q \in Q(o, 4) : x_q \leq x_{p_k}\}$ enthält keinen Punkt aus S.

Wenn die Menge S_{ij} nicht leer ist, dann gehören alle Punkte daraus zum Rechteck R_{ij} und sie liegen auf einem gemeinsamen Manhattan-Pfad zwischen t_i und t_j. Sei $T_{ij|i'j'}$ der Block der Pareto-Hülle von $S_{ij} \cup \{o, p_i, p_j\}$ (vgl. Abbildung 3.32). Aufgrund der Lage der Punkte ist dieser Block nicht degeneriert. Der Punkt o ist der Ursprung des Treppenpolygons. Analog lässt sich die Menge $S_{i'j'}$ und das Treppenpolygon $T_{i'j'|ij}$ definieren. Zwei andere Typen von Treppenpolygonen lassen sich definieren, wenn p_i und p_j im zweiten Quadranten und $p_{i'}$ und $p_{j'}$ im vierten Quadranten liegen. Im Folgenden nehmen wir für die Darstellung an, dass die Treppenpolygone nach einer passenden geometrischen Transformation immer im ersten Quadranten sind. Bei diesen Polygonen handelt es sich um Treppenpolygone, da der Punkt o links unterhalb von allen p_k liegt und die Rechtecke $R(\{p_k, o\})$ keinen weiteren Punkt aus S enthalten.

Diese Treppenpolygone unterscheiden sich nur minimal von den Treppenpolygonen von Benkert, Widmann und Wolff [BWW04], wie sie in Kapitel 3.2.4 auf Seite 54 vorgestellt wurden. Der Grund liegt, dass ein Streifen, so wie er hier definiert wurde und für die Treppenpolygone verantwortlich ist, nicht zwangsläufig einem Paar von $Z_{\text{ver}} \cup Z_{\text{hor}}$ von Seite 35 entsprechen muss.

Betrachte nun wieder das Treppenpolygon $T_{ij|i'j'}$. Sei α der Punkt ganz oben links vom Treppenpolygon $T_{ij|i'j'}$ und β der Punkt ganz unten rechts. Nenne M_{ij} den monotonen Pfad am Rand von $T_{ij|i'j'}$ zwischen α und β, der über alle Punkte von S_{ij} geht. Aufgrund der Definition gilt $T_{ij|i'j'} \cap S = S_{ij}$ und es befinden sich keine Punkte von S in der Region $Q^+(o, 2) = \{q \in Q(o, 2) : y_q \leq y_\alpha\}$ und ebenso keine in der Region $Q^+(o, 4) = \{q \in Q(o, 4) : x_q \leq x_\beta\}$. Im Besonderen bedeutet dies, dass kein kein Streifen

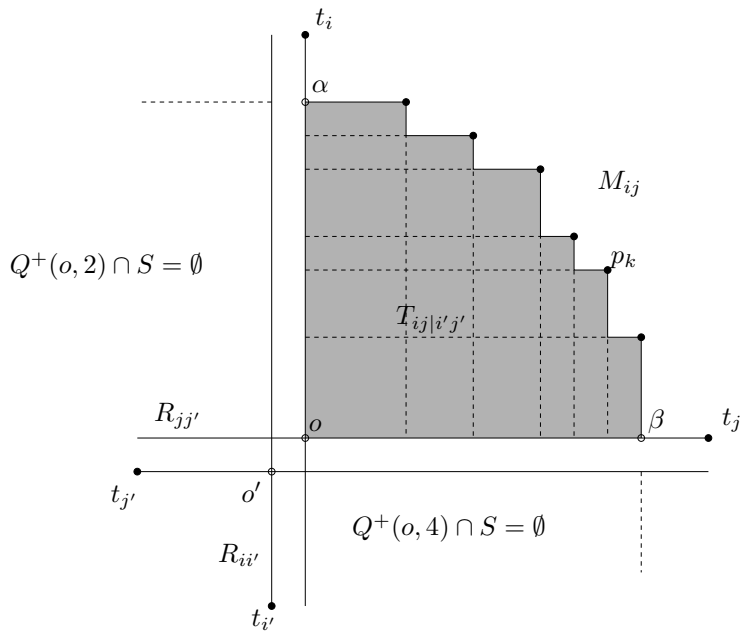


Abbildung 3.32: Das Treppenpolygon $T_{ij|i'j'}$. Die Punkte p_k liegen auf dem monotonen Pfad von α nach β. Unterhalb des Segmentes zwischen o und β, sowie links von dem Segment zwischen o und α liegen keine weiteren Punkte von S.

das Treppenpolygon $T_{ij|i'j'}$ durchquert. Außerdem ergibt sich automatisch aus der Definition, dass der Schnitt von Treppenpolygonen entweder leer ist oder einer Teilmenge von S entspricht. Dies heißt, zwei Treppenpolygone haben maximal Punkte aus S gemeinsam. Dies ist ähnlich zu den Treppenpolygonen aus Kapitel 3.1. Damit gehört jede Kante des Gitters Γ zu maximal einem Treppenpolygon.

Sei Z' die Menge aller Paare ij, so dass R_{ij} ein Streifen ist. Sei Z'' die Menge aller Paare $i'k$, mit der Eigenschaft, dass es ein Treppenpolygon $T_{ij|i'j'}$ gibt, so dass p_k zu der Menge S_{ij} gehört.

Lemma 3.31 *$Z = (Z' \cup Z'') \subseteq Z_\oslash$ ist eine erzeugende Menge.*

Beweis Lemma 3.31 Sei N ein Netzwerk, das Manhattan-Pfade für alle Punkte aus Z enthält. Um zu zeigen, dass N ein Manhattan-Netzwerk zu S ist, genügt es zu zeigen, dass N für ein beliebiges Paar $kk' \in Z_\oslash \setminus Z$ einen Manhattan-Pfad enthält. Ohne Beschränkung der Allgemeinheit sei $x_{p_{k'}} \leq x_{p_k}$ und $y_{p_{k'}} \leq y_{p_k}$ (vgl. Abbildung 3.33). Die anderen Fälle sind zu diesem symmetrisch.

Die vertikalen und horizontalen Geraden durch die Punkte p_k und $p_{k'}$ teilen die Ebene in das Rechteck $R_{kk'}$, vier offene Quadranten und vier geschlossene, unbeschränkte Halb-Streifen, die gegen den Uhrzeigersinn mit B_1 bis B_4 nummeriert sind. Da $p_k \in B_1 \cap S$

und $p_{k'} \in B_3 \cap S$ gilt, muss es mindestens einen vertikalen Streifen zwischen einem Punkt in B_1 und einem Punkt in B_3 geben. Sei $R_{i_1 i'_1}$ der Streifen ganz links, der durch $R_{kk'}$ durchgeht und $R_{i_2 i'_2}$ der Streifen ganz rechts, der durch $R_{kk'}$ durchgeht. Diese Streifen können zusammenfallen und können degeneriert sein, müssen jedoch von $R_{kk'}$ unterschiedlich sein, da $kk' \notin Z$ gilt. Ohne Beschränkung der Allgemeinheit können wir wieder aus Symmetriegründen annehmen, dass $p_{i_1}, p_{i_2} \in B_1$ und $p_{i'_1}, p_{i'_2} \in B_3$ gilt. Analog lässt sich der unterste horizontale Streifen $R_{j_1 j'_1}$ und der oberste horizontale Streifen $R_{j_2 j'_2}$ definieren. Dabei können wir wieder ohne Beschränkung der Allgemeinheit annehmen, dass p_{j_1} und p_{j_2} in B_4 liegen sowie $p_{j'_1}$ und $p_{j'_2}$ in B_2. Auch diese zwei Streifen können zusammenfallen oder degeneriert sein. Dabei muss der Streifen $R_{i_1 i'_1}$ so liegen, dass $p_{i'_1}$ auf der zu p_k gewandten Seite liegt. Sollte dies nicht der Fall sein, so würde es einen Streifen links von diesem Streifen geben, im Zweifelsfall den Streifen $R_{i_1 k'}$. Für die anderen drei Streifen gilt dieses Argument analog.

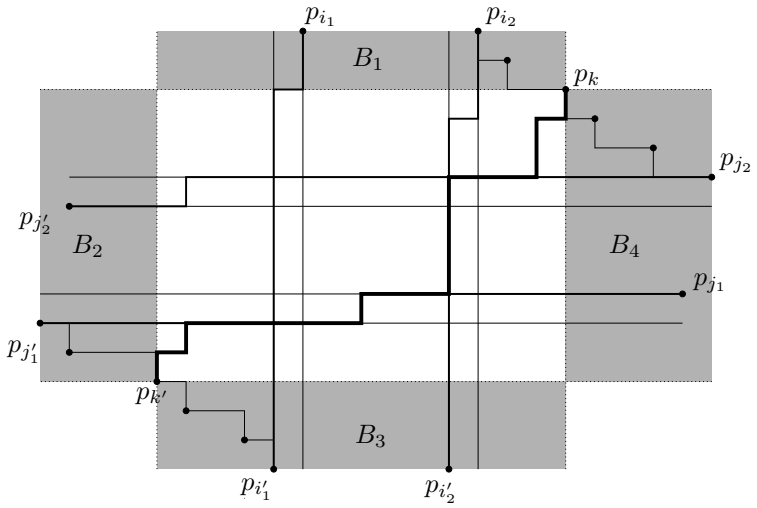


Abbildung 3.33: Konstruktion des Pfades von p_k nach $p_{k'}$. Da die vertikalen und horizontalen Streifen zwischen p_k und $p_{k'}$ eine Kreuzungs-Konfiguration bilden, sind sowohl $p_{k'}$ als auch p_k Teil eines Treppenpolygons und es lässt sich ein Pfad konstruieren.

Damit definiert jede Kombination eines vertikalen und eines horizontalen Streifens eine Kreuzungs-Konfiguration. Somit müssen $T_{i_2 j_2 | i'_2 j'_2}$ und $T_{i'_1 j'_1 | i_1 j_1}$ existieren. Dann muss der Punkt p_k entweder zu $S_{i_2 j_2}$ gehören oder mit einem der Punkte p_{i_2}, p_{j_2} identisch sein. Analog gilt $p_{k'} \in S_{i'_1 j'_1} \cup \{p_{i'_1}, p_{j'_1}\}$. Nach Lemma 3.30 muss jeder der Punkte $\{p_{i_1}, p_{i_2}, p_{j_1}, p_{j_2}\}$ mit jedem Punkt aus $\{p_{i'_1}, p_{i'_2}, p_{j'_1}, p_{j'_2}\}$ über einen Manhattan-Pfad verbunden sein. Außerdem muss ein Manhattan-Pfad zwischen p_k und $p_{i'_2}, p_{j'_2}$ existieren, sowie zwischen $p_{k'}$ und p_{i_1}, p_{j_1}. Aus diesen Pfaden lässt sich dann ein Manhattan-Pfad von p_k nach $p_{k'}$ konstruieren (vgl. Abbildung 3.33).

□

3.3.3 Der Rundungs-Algorithmus

Sei $(x, f) = \left((x_e)_{e\in E}, \left(f_e^{ij}\right)_{e\in E, ij\in F}\right)$ eine optimale Lösung des linearen Programms von (3.10'). Der Algorithmus rundet die Lösung (x, f) in drei Phasen.

Phase 0 In Phase 0 fügen wir alle Kanten von ∂P in die ganzzahlige Lösung an.

Phase 1 In dieser Phase wird jeder Streifen $R_{ii'}$ gerundet um einen Manhattan-Pfad von p_i nach $p_{i'}$ sicherzustellen.

Phase 2 Hier wird eine iterative Rundungsfunktion auf jedes Treppen-Polygon angewandt.

Phase 1 Sei nun $R_{ii'}$ ein Streifen. Wenn $R_{ii'}$ degeneriert ist, dann entspricht das Segment $\text{Seg}[p_i, p_{i'}]$ dem eindeutigen Manhattan-Pfad zwischen p_i und $p_{i'}$, womit $x_e = f_e^{ii'} = 1$ für jede Kante $e \in \text{Seg}[p_i, p_{i'}]$ gelten muss. Wenn $R_{ii'}$ nicht degeneriert ist, dann hat jeder Manhattan-Pfad von p_i nach $p_{i'}$ folgende Form: Er geht an der Seite von $R_{ii'}$, auf der p_i liegt, entlang und knickt dann ab, um entlang einer Kante e, der Wechselkante, den Streifen $R_{ii'}$ zu überqueren und geht dann an der anderen Seite von $R_{ii'}$ entlang bis zu $p_{i'}$. Es kann mehrere solcher Manhattan-Pfade geben, die vom gebrochenen Fluss $f^{ii'}$ zwischen p_i und $p_{ii'}$ genutzt werden, allerdings sorgt die Cut-Bedingung dafür, dass $x_e + x_{e'} \geq f_e^{ii'} + f_{e'}^{ii'} \geq 1$ für jedes Zwillingspaar e und e' auf gegenüberliegenden Seiten des Streifen $R_{ii'}$ gilt. Daraus ergibt sich, dass $\max\{x_e, x_{e'}\} \geq \frac{1}{2}$ gilt.

Sei nun p der Punkt, der am weitesten von p_i auf der gleichen Seite von $R_{ii'}$ entfernt ist, so dass alle Kanten e auf dem Segment zwischen p und p_i den Fluss $x_e \geq \frac{1}{2}$ haben. Entspreche nun das Segment $\text{Seg}[pp']$ der Kante von Γ, die $R_{ii'}$ durchquert. Dann gilt für das Segment $\text{Seg}[p'p_{i'}]$, dass der Fluss auf allen Kanten von diesem Segment ebenfalls mindestens den Wert $\frac{1}{2}$ hat. Damit ergibt sich für Phase 1 folgendes Verfahren:

Funktion RoundStrip Wir betrachten jeden Streifen $R_{ii'}$. Sollte $R_{ii'}$ degeneriert sein, dann nehmen wir die ganzzahlige Lösung, bestehend aus den Kanten zwischen p_i und $p_{i'}$. Ansonsten werden die Kanten zwischen p_i und p aufgerundet, sowie die Kanten zwischen p' und $p_{i'}$. Weiterhin nehmen wir die Wechselkante $\text{Seg}[pp']$ hinzu. In beiden Fällen wird der entstehende Manhattan-Pfad mit $P_{ii'}$ bezeichnet. Damit ist, wie im vorhergehenden Absatz gezeigt, gewährleistet, dass alle gewählten Kanten, außer der Wechselkante $\text{Seg}[pp']$, mindestens den Wert $\frac{1}{2}$ hatten.

Phase 2 Betrachte nun ein Treppenpolygon $T_{ii'|jj'}$. Sei ϕ der gemeinsame Punkt der beiden Manhattan-Pfade $P_{ii'}$ und $P_{jj'}$, der den geringsten Abstand zu p_i hat. Dieser Punkt muss ein Eckpunkt der Fläche von Γ sein, die o und o' enthält. Seien $P_{ii'}^+$ und $P_{jj'}^+$ die Teilpfade von $P_{ii'}$ und $P_{jj'}$, bestehend aus den Pfaden zwischen ϕ und p_i bzw. p_j. Nun erweitern wir das Treppenpolygon leicht, indem wir jetzt als $T_{ii'|jj'}$ die Region begrenzt von den Pfaden $P_{ii'}^+, P_{jj'}^+$ und M_{ij} betrachten. Jeder Fluss $f^{ki'}$ oder $f^{kj'}$ in $T_{ii'|jj'}$ kann gebrochen sein, da mehrere Manhattan-Pfade zwischen p_k und $p_{i'}$ den Fluss

$f^{ki'}$ tragen. Jeder dieser Pfade schneidet aber den Pfad $P^+_{ii'}$ oder $P^+_{jj'}$, somit erreicht der gesamte Fluss $f^{ki'}$ die Pfade $P^+_{ii'} \cup P^+_{jj'}$. Dieser Fluss kann nun über ϕ und von da aus zu $p_{i'}$ umgeleitet werden. Damit muss nur noch der Fluss im Inneren des erweiterten Treppenpolygons $S_{ii'|jj'}$ gerundet werden. Entweder am Pfad $P^+_{ii'}$ oder am Pfad $P^+_{jj'}$ muss mindestens $\frac{1}{2}$ vom $f^{ki'}$-Fluss ankommen.

Funktion RoundStaircase Für ein Treppenpolygon $S_{ii'|jj'}$, das durch die Manhattan-Pfade $P^+_{ii'}$, $P^+_{jj'}$ und den monotonen Pfad M_{ij} definiert ist, suchen wir nun den niedrigsten Punkt $p_m \in S_{ij}$, so dass der $f^{mi'}$-Fluss auf den Manhattan-Pfaden zwischen p_m und $p_{i'}$, die zuerst auf $P^+_{ii'}$ ankommen, mindestens $\frac{1}{2}$ ist. Da der Fluss entweder an $P^+_{ii'}$ oder $P^+_{jj'}$ den Wert $\geq \frac{1}{2}$ haben muss, können wir ohne Beschränkung der Allgemeinheit annehmen, dass dieser Punkt existiert. Der Fall für $P^+_{jj'}$ geht bis auf Symmetrie analog. Sei nun p_s der Punkt von S_{ij}, der direkt unterhalb von p_m liegt. Dieser Punkt muss nicht existieren. Wenn er jedoch existiert, ist der $f^{si'}$-Fluss, der $P^+_{jj'}$ erreicht, mindestens $\frac{1}{2}$. Sei nun ϕ' der Schnittpunkt der horizontalen Geraden durch den Punkt p_m mit dem Pfad $P^+_{ii'}$. Analog sei ϕ'' der Schnittpunkt der vertikalen Geraden durch p_s mit dem Pfad $P^+_{jj'}$. Runde nun alle Kanten des horizontalen Segments $\text{Seg}[p_m, \phi']$ und alle Kanten des vertikalen Segmentes $\text{Seg}[p_s, \phi'']$ auf. Wenn S_{ij} Punkte oberhalb von p_m enthält, dann rufe RoundStaircase rekursiv mit dem erweiterten Treppenpolygon auf, das durch folgende Stücke definiert ist:

- Das Segment $\text{Seg}[p_m, \phi']$
- Der Teilpfad von $P^+_{ii'}$ zwischen p_i und ϕ'
- Der Teilpfad M_{im} zwischen α und p_m von M_{ij}

Analog, wenn es Punkte in S_{ij} unterhalb von p_s gibt, dann rufe RoundStaircase rekursiv mit dem erweiterten Treppenpolygon auf, das durch das Segment $\text{Seg}[p_s, \phi'']$, dem Teilpfad von $P^+_{jj'}$ zwischen p_j und ϕ'' und dem Teilpfad M_{sj} zwischen p_s und β definiert ist (vgl. Abbildung 3.34).

Sei nun E_0 die Menge aller Kanten von Γ, die zum Rand der Pareto-Hülle von S gehören. E_1 sei die Menge aller Kanten der Funktion RoundStrip, die nicht zu E_0 gehören, und E_2 sei die Menge aller Kanten, die von der rekursiven Funktion RoundStaircase gewählt werden und die nicht zu $E_0 \cup E_1$ gehören. Sei N das Netzwerk, das durch diese Kanten und die Punkte von S definiert ist. Aus Lemma 3.31 und den Rundungsfunktionen ergibt sich, dass N ein Manhattan-Netzwerk für S ist. Sei x^* die ganzzahlige Lösung für Gleichung (3.9), die zu N gehört. Das heißt, es gilt $x^*_e = 1$ genau dann, wenn e zu $E_0 \cup E_1 \cup E_2$ gehört und $x^*_e = 0$ sonst.

3.3.4 Analyse

Nun zeigen wir, dass das Manhattan-Netzwerk N aus dem vorherigen Abschnitt maximal doppelt so groß ist, wie die Kosten einer optimalen, gebrochenen Lösung von (3.9'). Also

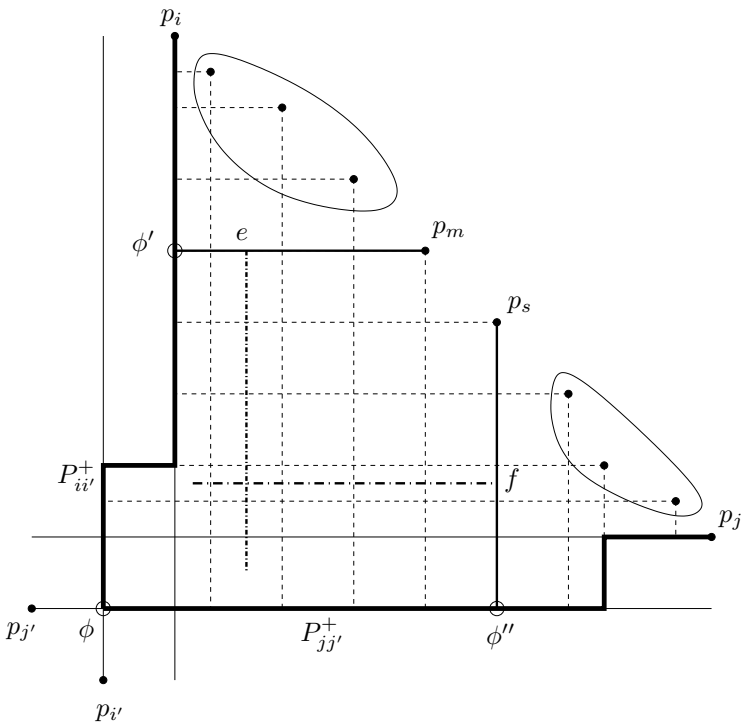


Abbildung 3.34: Die Funktion RoundStaircase fügt das Segment $\text{Seg}[p_m, \phi']$ und das Segment $\text{Seg}[p_s, \phi'']$ hinzu und ruft für die eingekreisten Punkte sich selbst rekursiv auf. Der Kante e bzw der Kante f werden jeweils die Kanten auf dem dick gestrichelten Pfad zugeordnet, so dass sie jeweils Kanten mit einem Fluss von mindestens $\frac{1}{2}$ abdecken.

ist folgende Gleichung zu zeigen:

$$\text{cost}(x^*) = \sum_{e \in E} |e| x^*_e \leq 2 \sum |e| x_e = 2\text{cost}(x) \tag{3.11}$$

Es gilt $x_e = x^*_e = 1$ für jede Kante $e \in E_0$. Um nun die Ungleichung (3.11) zu zeigen, weisen wir jeder Kante $e \in E_1 \cup E_2$ eine Menge E_e von parallelen Kanten zu, so dass folgende zwei Bedingungen gelten:

(i) $\sum_{e' \in E_e} x_{e'} \geq \frac{1}{2}$

(ii) $E_e \cap E_f = \emptyset$ für $e, f \in E_1 \cup E_2$ und $e \neq f$.

Da die Kanten aus E_0 den Fluss von eins haben, können sie zur Hälfte sich selbst in Rechnung gestellt werden und die andere Hälfte kann für eine beliebige Menge E_e verwendet werden.

Sei e nun eine Kante aus E_1, die zu dem Pfad $P_{ii'}$ des Streifen $R_{ii'}$ gehört. Sollte e auf einer der beiden Seiten des Streifen liegen, so gilt aufgrund der Konstruktion $x_e \geq \frac{1}{2}$ und in diesem Fall ist $E_e = \{e\}$. Wenn e die Wechselkante des Streifen ist, dann enthält E_e eine der beiden parallelen Kanten von $\partial\mathcal{P}$ auf dem Rand der Pareto-Hülle. Dies geht, da die Kanten der Hülle nur zur Hälfte sich selber in Rechnung gestellt wurden. Aus der Definition der Streifen ergibt sich, dass keine andere Wechsel-Kante parallel zu diesen Kanten von $\partial\mathcal{P}$ sein kann, womit jedes Paar von parallelen Kanten von $\partial\mathcal{P}$ maximal in einer Menge E_e für die Wechselkante e auftauchen kann.

Betrachte nun eine Kante $e \in E_2$. Diese Kante e gehört zu dem erweiterten Treppenpolygon $S_{ii'|jj'}$. Wenn e auf dem Segment $\text{Seg}[p_m, \phi']$ liegt, dann besteht E_e aus der Kante e und allen Kanten, die parallel zu e sind und unter dem Segment $\text{Seg}[p_m, \phi']$ liegen (vgl. Abbildung 3.34). Da jeder Manhattan-Pfad zwischen p_m und $p_{i'}$, der über $P^+_{ii'}$ geht, eine dieser Kanten schneiden muss und der Fluss $f^{mi'}$ an $P^+_{ii'}$ aufgrund der Konstruktion mindestens $\frac{1}{2}$ ist, gilt $\sum_{e' \in E_e} x_{e'} \geq \frac{1}{2}$, woraus sich (i) ergibt. Analog, wenn f eine Kante des Segments $\text{Seg}[p_s, \phi'']$ ist, dann besteht E_f aus der Kante f und allen Kanten, die links von f liegen und die parallel zu f sind. Damit gilt $E_e \cap E_f = \emptyset$. Da sowohl E_e, als auch E_f zu der Region von $S_{ii'|jj'}$ gehören, die durch die Segmente $\text{Seg}[p_m, \phi']$ und $\text{Seg}[p_s, \phi'']$ begrenzt ist, sind alle Kanten e', die von den rekursiven Aufrufen betrachtet werden, disjunkt von diesen Kanten, somit gilt $E_e \cap E_{e'} = \emptyset$ und $E_f \cap E_{e'} = \emptyset$. Weiterhin gehört jede Kante des Gitters Γ zu maximal einem Treppenpolygon, so dass diese Aussage auch für verschiedene Treppenpolygone gültig bleibt. Da es keinen Streifen links oder unterhalb des Treppenpolygons geben kann, der das Treppenpolygon durchquert, kann keine Kante aus E_1 zu einer Menge E_e für ein $e \in E_2 \cap S_{ii'|jj'}$ hinzugefügt werden, womit sich Eigenschaft (ii) ergibt. Aus dieser Eigenschaft folgt dann sofort die Gleichung (3.11). Mit dieser Gleichung lässt sich nun sofort folgendes Theorem zeigen:

Theorem 3.32 *Der Rundungs-Algorithmus erreicht einen Approximationsfaktor von zwei für das minimale Manhattan-Netzwerk Problem.*

Die Laufzeit wird nicht explizit angegeben, allerdings hat das lineare Programm $O(n^3)$ Variablen und Bedingungen.

3.4 Weitere Algorithmen

Von Benkert, Wolff, Widmann und Shirabe [BWWS06] wird kurz ein Mixed-Integer Programm zur Berechnung des minimalen Manhattan-Netzwerkes vorgestellt, das in exponentieller Zeit ein minimales Manhattan-Netzwerk berechnet. Weiterhin wird im gleichen Paper ein Lineares Programm mit nachfolgender Rundung vorgestellt, was einfacher als das Programm aus Kapitel 3.3 von Chepoi, Nouioua und Vaxès [CNV05] ist, jedoch keinen Approximationsfaktor garantiert. Allerdings hat es auch eine Laufzeit, die zwar polynomiell, in der Praxis aber nicht viel schneller ist als das Mixed-Integer Programm zur exakten Lösung.

Von Kato, Imai und Asano [KIA02] wird ein Faktor zwei Approximations-Algorithmus vorgestellt. Allerdings ist der Beweis, dass dieser Algorithmus eine 2-Approximation be-

rechnet, lückenhaft. Daher wurde auf diesen Algorithmus hier nicht weiter eingegangen. Allerdings werden in diesem Paper erstmalig Cover und erzeugende Mengen verwendet, die dann in Kapitel 3.2 als Grundlage verwendet werden.

Einen $1,5$-Approximations-Algorithmus stellen Seibert und Unger [SU05] vor, allerdings sind dort Beweis und Algorithmusteile unklar, die in der noch nicht verfügbaren, längeren Journal Version ausführlicher dargestellt werden sollen. Aus diesem Grund wird auch auf diesen Algorithmus hier nicht weiter eingegangen.

3.5 Zusammenfassung

Allen drei, im Detail vorgestellten, Algorithmen aus den Papern ist gemeinsam, dass sie versuchen, die Punktmenge zu unterteilen. Für diese lokalen Mengen werden dann Netzwerke berechnet, die sich abschätzen lassen. Der Algorithmus aus Kaptitel 3.1 definiert für jeden Punkt Einflussbereiche, für die dann jeweils ein optimales Netzwerk in polynomieller Zeit berechnet werden kann. Einflussbereiche eines Typs von verschiedenen Punkten können sich nicht überlappen, aber Einflussbereiche von verschiedenen Typen und Punkten schon, weshalb der Algorithmus maximal einen Faktor von vier schafft.

Die beiden anderen Algorithmen benutzen erzeugende Mengen, die jeweils eine Teilmenge von $Z_{\oslash}$ sind. Diese sind sich zwar ähnlich, unterscheide sich jedoch im Detail. Die Menge $Z_{\text{ver}} \cup Z_{\text{hor}}$ aus Kapitel 3.2 auf Seite 35 ist der Menge Z' aus Kapitel 3.3 auf Seite 78 sehr ähnlich, allerdings nicht gleich (vgl. Abbildung 3.35). Beiden Algorithmen

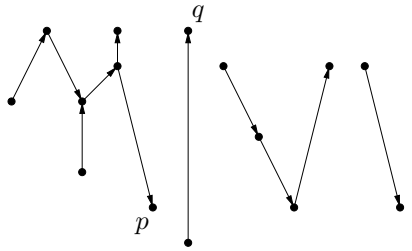


Abbildung 3.35: Vergleich von Z_{ver} und Z'. Abbgebildet ist die Menge Z_{ver}. Die Menge Z' enthält alle Paare, die auch in Z_{ver} enthalten sind, aber zusätzlich unter anderem auch das Paar (p, q), dass weder in Z_{ver} noch in Z_{hor} enthalten ist.

ist jedoch wieder gemeinsam, dass die jeweilige erzeugende Menge in Untermengen aufgeteilt wird. Beim Algorithmus aus Kapitel 3.2 ergeben sich dann am Ende drei Netze in zwei disjunkten Unterteilungen der Ebene. Für jede der beiden Regionen der Ebene lässt sich dann der Faktor drei als Approximationsfaktor zeigen, was damit den Faktor drei für das gesamte Netzwerk ergibt. Der Algorithmus aus Kapitel 3.3 erzeugt ebenfalls zwei disjunkte Unterteilungen. Allerdings wird mittels einem linearen Programm dort ein gesamtes, gebrochen-rationales Manhattan-Netzwerk berechnet. Für jede der beiden

Unterteilungen wird dann die gebrochen-rationale Lösung so aufgerundet, dass sich ein Faktor von zwei ergibt.

Interessant ist, dass alle drei Algorithmen in einem ihrer Schritte Treppenpolygone benutzen. Zwar sind diese in jedem Algorithmus anders definiert, aber der komplexeste Teil aller Algorithmen ist die Berechnung einer möglichst guten Lösung für diese Treppenpolygone.

4 Minimale Manhattan-Netzwerke von Rechtecken

In diesem Kapitel werden Manhattan-Netzwerke von Punktmengen betrachtet, die auf dem Rand eines Rechtecks liegen. Dieser Spezialfall ist in der Literatur noch nicht behandelt worden. Das neue, hier vorgestellte Verfahren läuft in $O(n \log n)$ und berechnet ein exaktes, minimales Manhattan-Netzwerk.

4.1 Eigenschaften dieser Netzwerke

Die Kontur einer Punktmenge S auf einem Rechteck entspricht nur dann exakt dem Rechteck, wenn die vier Eckpunkte des Rechtecks in der Punktmenge enthalten sind. Ansonsten ist das Kontur-Polygon kleiner als das Rechteck und überdeckt nicht das ganze Rechteck (vgl. Abbildung 4.1).

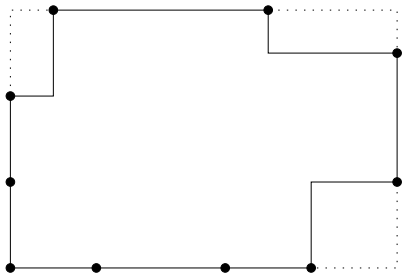


Abbildung 4.1: Kontur einer Punktmenge auf einem Rechteck. Diese entspricht nicht dem Rechteck (gestrichelt skizziert), sondern ist kleiner.

Auf der Kontur liegen sämtliche Punkte der Punktmenge, im Innern der Kontur liegen keine Punkte. Die Kontur deckt damit bereits einen Großteil der Manhattan-Pfade ab. Nur zwischen Punkten, die auf dem oberen Rand des umschließenden Rechtecks $R(S)$ und dem unteren Rand des Rechtecks $R(S)$ befinden, so wie zwischen Punkten, die auf dem linken Rand des umschließenden Rechtecks $R(S)$ und dem rechten Rand des Rechtecks $R(S)$ befinden, gibt es keine Manhattan-Pfade entlang der Kontur.

Lemma 4.1 *Die Manhattan-Pfade zwischen Punkten auf dem unteren und dem oberen Rand von $R(S)$ sind unabhängig von den Pfaden zwischen Punkten auf dem linken und dem rechten Rand von $R(S)$.*

Beweis Lemma 4.1 Betrachte einen Punkt p auf dem unteren Rand und einen Punkt q auf dem oberen Rand, so dass $R(\{p, q\}$ ein kritisches Rechteck ist (vgl. Abbildung 4.2). Die Kontur deckt dabei die beiden horizontalen Kanten des Rechtecks bereits ab. Damit

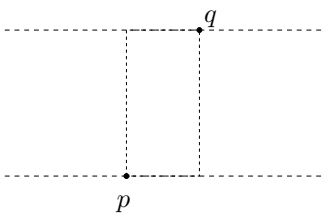


Abbildung 4.2: Pfad zwischen 2 Punkten auf dem oberen und unteren Rand der Kontur. Die beiden horizontalen Kanten werden bereits von der Kontur abgedeckt.

reicht es, eine vertikale Kante hinzuzufügen, um einen Manhattan-Pfad von p nach q zu erhalten. Kürzere Kanten reichen nicht aus, um einen Manhattan-Pfad zu erhalten. Damit müssen jedoch Kanten von dieser Länge hinzugefügt werden und diese Kanten reichen aus, um ein Manhattan-Netzwerk zu erreichen. Der einzige Fall, der Probleme bereitet, ist, wenn auch Teile der Kontur entlang des linken oder rechten Randes des Rechtecks $R(\{p, q\}$ laufen. Aber auch in diesem Fall kommt man mit einer vertikalen Kante aus, diese muss allerdings dann nicht die komplette Höhe des Rechtecks $R(\{p, q\}$ überbrücken, sondern nur den Teil, der innerhalb der Kontur liegt.

Analog reicht es für zwei Punkte p' und q' auf dem linken und rechten Rand eine horizontale Kante hinzuzufügen, um einen Manhattan-Pfad von p' nach q' zu erhalten. Somit gibt es ein minimales Manhattan-Netzwerk, bei dem die Pfade zwischen den Punkten auf dem oberen und unteren Rand mit den Pfaden zwischen Punkten auf dem linken und rechten Rand keine Liniensegmente innerhalb der Kontur gemeinsam haben.

□

Aus diesem Lemma folgt, dass sich die horizontalen und vertikalen Kanten voneinander unabhängig berechnen lassen. Die Kanten, die benötigt werden, um die Punkte auf dem oberen Rand mit den Punkten auf dem unteren Rand von $R(S)$ zu verbinden, wenn die Kontur bereits vorhanden ist, wird *vertikales Netzwerk* genannt. Analog ist das horizontale Netzwerk definiert. Diese beiden Netzwerke sind keine echten Manhattan-Netzwerke, da sie nur Teilpfade enthalten und erst die Vereinigung mit der Kontur die gesamten Pfade enthalten.

Korollar 4.2 *Ein minimales Manhattan-Netzwerk von Punkten auf einem Rechteck ist gegeben durch:*

$$\begin{array}{ll} & \textit{Kontur} \\ \bigcup & \textit{minimales vertikales Netzwerk} \\ \bigcup & \textit{minimales horizontales Netzwerk} \end{array}$$

Beweis Korollar 4.2 Da sich nach Lemma 4.1 die horizontalen und vertikalen Netzwerke getrennt berechnen lassen und nach Lemma 2.8 die Kontur enthalten ist, kann man jeweils für diese drei Netzwerke das minimale Netzwerk berechnen. Die Vereinigung dieser drei Netzwerke ist dann das minimale Manhattan-Netzwerk.

□

Interessant ist der Zusammenhang mit dem Cover, dass in Kapitel 3.11 auf Seite 38 definiert wird. Das minimale vertikale Netzwerk entspricht einem Cover für die Menge Z_{ver} aus Kapitel 3.2, allerdings sind keine weiteren Liniensegmente nötig, da die horizontalen Stücke bereits von der Kontur abgedeckt werden. Damit lässt sich aus Lemma 3.12 auf Seite 38 die gleiche Aussage, wie aus dem obigen Lemma, folgern.

4.2 Berechnung eines minimalen vertikalen Netzwerkes

Zuerst wird ein Algorithmus definiert, der kein minimales Netzwerk findet, sondern nur ein günstiges. *Günstig* ist das Netzwerk in dem Sinne, dass ein weiterer Algorithmus mit diesem Netzwerk später das minimale Netzwerk berechnen kann. Voraussetzung für den Algorithmus ist, dass es – außer auf dem linken oder rechten Rand des Rechtecks – keine zwei Punkte gibt, die die gleiche x-Koordinate haben. Dies lässt sich aber leicht dadurch erreichen, dass das Rechteck an der Stelle, wo die Voraussetzung nicht erfüllt ist, in jeweils zwei Rechtecke geteilt wird (vgl. Abbildung 4.3(b)). Da an dieser Stelle auf jeden Fall eine Kante eingefügt werden muss, kann der Algorithmus dann sowohl für das linke Rechteck R_l als auch für das rechte Rechteck R_r angewandt werden.

Die Berechnung des günstigen, vertikalen Netzwerkes geschieht mittels eines Sweep-Algorithmus. Dieser Algorithmus geht mit einer Sweep-Line von links nach rechts durch die Punktmenge S. Als Ereignisse des Sweep-Algorithmus werden die Punkte auf dem oberen und dem unterem Rand von $R(S)$ betrachtet, die nicht auf dem linken oder rechten Rand von $R(S)$ liegen. Die Eckpunkte des Rechtecks werden dadurch – sofern sie in S überhaupt enthalten sind – nicht beachtet. Bei jedem Ereignis wird überprüft, ob die Manhattan-Bedingung erhalten bleibt, wenn an dieser Stelle keine senkrechte Kante eingefügt wird. Sollte dies nicht der Fall sein, so wird eine Kante eingefügt. Die Kanten werden dabei innerhalb der Kontur eingefügt, sie verlassen die Kontur nicht. Der Algorithmus für die horizontalen Kanten ist analog definiert.

Theorem 4.3 *Das vom Algorithmus 4.1 berechnete Netzwerk vereinigt mit der Kontur verbindet jeden Punkt auf dem oberen Rand mit jedem Punkt auf dem unteren Rand über einen Manhattan-Pfad.*

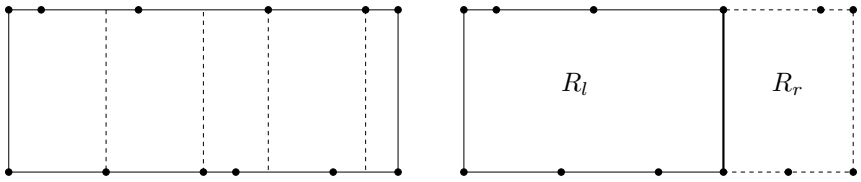
Beweis Theorem 4.3 Angenommen, es gibt zwischen einem Punkt p_1 auf dem oberen Rand und einem Punkt p_2 auf dem unteren Rand keinen Manhattan-Pfad. Es gelte $x_{p_1} < x_{p_2}$. Da der Algorithmus nur senkrechte Kanten einfügt, kann es in dem Intervall $[x_{p_1}, x_{p_2}]$ keine senkrechte Kante geben, da diese einen Manhatten-Pfad ermöglichen würde. Dies bedeutet, dass bei dem Ereignis für Punkt p_1 keine Kante eingefügt werden

Algorithmus 4.1 Berechnung eines günstigen vertikalen Netzwerkes

Eingabe: Eine Punktmenge $P_v \subset S$, die die Punkte auf dem oberen und unteren Rand des Rechtecks enthält, die nicht auch auf dem rechten oder linken Rand enthalten sind, und die Kontur der Punktmenge S.

Ausgabe: Ein Netzwerk M für das gilt: $\forall p, q \in P_v, y_p \neq y_q : \exists$ Manhattan-Pfad in $M \cup$ Kontur

1: $P_{\text{sort}} = P_v$ nach x-Koordinaten sortiert.
2: $P_{\text{unten}} = \emptyset$
3: $P_{\text{oben}} = \emptyset$
4: **for** $p \in P_{\text{sort}}$ **do**
5: **if** p liegt auf dem unteren Rand **then**
6: **if** $P_{\text{oben}} \neq \emptyset$ **then**
7: Füge eine senkrechte Kante bei p ein ▷ Sonst gibt es zwischen den Punkten aus P_{oben} und p keinen Manhattan-Pfad
8: $P_{\text{oben}} = \emptyset$
9: **else**
10: $P_{\text{unten}} = \{p\}$
11: **end if**
12: **else**
13: **if** $P_{\text{unten}} \neq \epsilon$ **then**
14: Füge eine senkrechte Kante bei p ein
15: $P_{\text{unten}} = \emptyset$
16: **else**
17: $P_{\text{oben}} = \{p\}$
18: **end if**
19: **end if**
20: **end for**



(a) Minimales vertikales Netzwerk. Dabei werden alle Punkte auf dem oberen Rand mit allen Punkten auf dem unteren Rand verbunden.

(b) Sollten ein Punkt auf dem oberen und ein Punkt auf dem unteren Rand die gleichen x-Koordinaten haben, wird das Rechteck in zwei Rechtecke geteilt. Dann lässt sich das Manhattan-Netzwerk für R_l und R_r getrennt berechnen.

Abbildung 4.3: Vertikale Kanten des Manhattan-Netzwerkes

darf. Damit gilt nach dem Ereignis $P_{\text{oben}} = \{p_1\} \neq \emptyset$ (vgl. Zeile 17 in Algorithmus 4.1). Bis zum Punkt p_2 ist keine Kante eingefügt worden. Damit muss weiterhin $P_{\text{oben}} \neq \emptyset$ gelten. Bei dem Ereignis von Punkt p_2 handelt es sich um einen Punkt auf dem unteren Rand. Da $P_{\text{oben}} \neq \emptyset$ gilt, fügt der Algorithmus an dieser Stelle eine Kante ein (vgl. Zeile 7 in Algorithmus 4.1). Der Fall $x_{p_2} < x_{p_1}$ geht analog.

□

Sollten die Ecken des Rechtecks nicht in der Punktmenge enthalten sein, so findet der Algorithmus nicht das minimale Netzwerk. Das liegt daran, dass die Kanten auf dem linken bzw. rechten vertikalen Rand der Kontur entlanglaufen können. Dies berücksichtigt der Algorithmus nicht. Formal lassen sich die problematischen Kanten wie folgt beschreiben (vgl. Abbildung 4.4(a)):

Kante a: Diese Kante ist definiert als die Kante mit der x-Koordinate des ersten Punktes auf dem oberen oder unteren Rand. Dabei werden, genau wie im Algorithmus 4.1, die Eckpunkte des Rechtecks nicht beachtet.

Kante b: Diese Kante entspricht der Kante mit der x-Koordinate des ersten Punktes auf dem oberen Rand, wenn die Kante a durch einen Punkt auf dem unteren Rand definiert wurde, ansonsten entspricht sie der Kante mit der x-Koordinate des ersten Punktes auf dem unteren Rand. Dabei werden, genau wie bei Kante a, die Eckpunkte des Rechtecks nicht beachtet.

Kante c: Diese Kante wird analog zur Kante b definiert, allerdings vom rechten Rand aus gesehen.

Kante d: Diese Kante wird analog zur Kante a definiert, allerdings vom rechten Rand aus gesehen.

Für den Vergleich der Größe von vertikalen Netzwerken sind nur die Teilstücke interessant, die echt innerhalb der Kontur verlaufen. Nach Lemma 2.8 ist die Kontur auf jeden Fall im minimalen Manhattan-Netzwerk enthalten. Daher wächst beim Einfügen von vertikalen Kanten das Netzwerk nur um die Länge des Stückes, das nicht auf der Kontur verläuft. Deshalb werden a', b', c', d' definiert als die größten Teilstücke der jeweiligen Kante, die echt innerhalb der Kontur verlaufen (vgl. Abbildung 4.4(b)).

Lemma 4.4 *Der Algorithmus 4.1 fügt als erste Kante die Kante b hinzu.*

Beweis Lemma 4.4 Ohne Beschränkung der Allgemeinheit sei die Kante a durch einen Punkt auf dem oberen Rand definiert, die Kante b damit durch einen Punkt auf dem unteren Rand. Damit ergibt sich ein Aussehen wie in Abbildung 4.5. Da Kante b durch den ersten Punkt auf dem unteren Rand definiert ist, handelt es sich bei allen vorher auftretenden Ereignissen um Punkte auf dem oberen Rand. Damit fügt der Algorithmus 4.1 bei diesen Ereignissen keine Kante ein, da eine Kante nur nach einem Wechsel vom oberen zum unteren oder vom unteren zum oberen Rand eingefügt werden kann

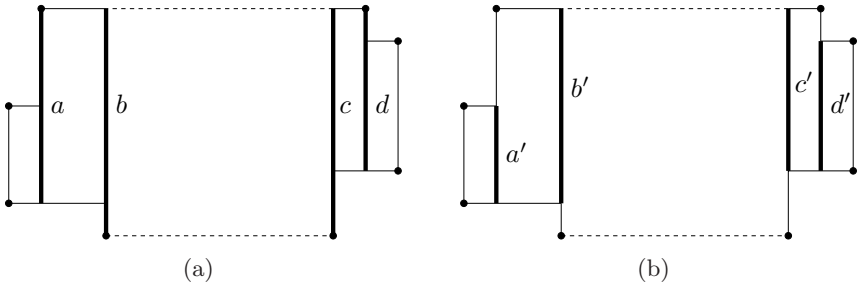


Abbildung 4.4: Problematische Kanten des Algorithmus 4.1. Der Algorithmus berücksichtigt nicht, dass von den Kanten a bis d nur die Anteile a' bis d' für die Länge des Manhattan-Netzwerkes relevant sind.

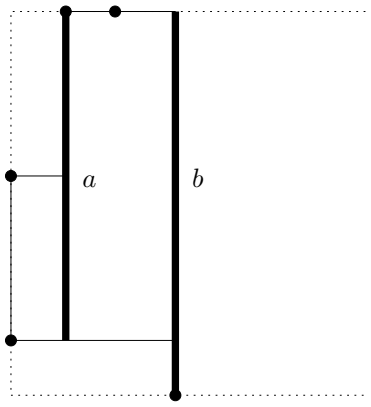


Abbildung 4.5: Der Algorithmus 4.1 fügt als erstes die Kante b ein, da es vorher nur Punkte auf dem oberen Rand gibt. Daher ist die Kante b die erste Kante, die wichtig ist, um die Manhattan-Pfade sicherzustellen.

(vgl. Zeile 17 in Algorithmus 4.1). Da die Kante a links von der Kante b liegt, muss es aber auch mindestens ein solches Ereignis, bei dem keine Kante eingefügt wird, geben. Nach diesem Ereignis gilt dann $P_{\text{oben}} \neq \emptyset$. Sobald der Algorithmus auf den Punkt trifft, durch den die Kante b definiert wird, wird eine Kante eingefügt, da immer noch $P_{\text{oben}} \neq \emptyset$ gelten muss (vgl. Zeile 7 in Algorithmus 4.1).

□

Korollar 4.5 *Es gibt ein minimales vertikales Netzwerk, das als erste Kante entweder die Kante a oder die Kante b enthält.*

Beweis Korollar 4.5 Annahme: Es gibt eine Punktmenge auf einem Rechteck, wo die erste Kante des vertikalen Netzwerkes weder der Kante a noch der Kante b entsprechen

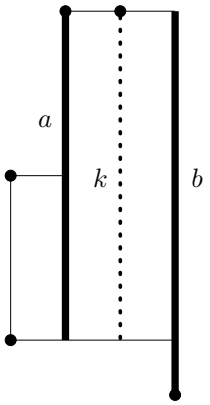


Abbildung 4.6: Es gibt ein minimales Manhattan-Netzwerk, in dem entweder die Kante a oder die Kante b die erste Kante ist. Die Kante k hingegen könnte durch die Kante b ersetzt werden.

kann. Aus Lemma 4.4 ergibt sich, dass der Algorithmus die Kante b hinzufügt. Er macht dies, weil sonst die Manhattan-Bedingung verletzt wird. Damit kann die erste Kante in einem minimalen Netzwerk nicht rechts von der Kante b liegen. Dann muss die erste Kante einer Kante k zwischen a und b entsprechen (vgl. Abbildung 4.6). Diese Kante k hat aber die gleiche Länge wie b', womit man anstelle der Kante k auch die Kante b wählen kann, ohne dass das Netzwerk größer wird oder die Manhattan-Eigenschaft verloren geht.

Die Kante a hingegen lässt sich nicht immer durch die Kante b ersetzen, da $|a'| < |b'|$ gelten kann (vgl. Abbildung 4.4(b)).

□

Korollar 4.6 *Ein minimales vertikales Netzwerk enthält immer als zweite Kante eine Kante rechts von b, sofern eine zweite Kante notwendig ist.*

Beweis Korollar 4.6 Annahme: Es gibt ein minimales Manhattan-Netzwerk zu einer beliebigen Punktmenge S auf einem Rechteck, das als zweite Kante die Kante b oder eine Kante links davon enthält. Wir zeigen nun, dass man in diesem Fall eine Kante streichen kann, was ein Widerspruch zu der Minimalitäts-Bedingung des Netzwerkes wäre.

Laut Korollar 4.5 muss die erste Kante dann die Kante a sein. Für die zweite Kante gibt es nun zwei Fälle:

Fall a) Die zweite Kante entspricht der Kante b: Dann kann man die Kante a wegfallen lassen, damit wird das Netzwerk kleiner, aber die Manhattan-Eigenschaft bleibt erhalten. Dies ist ein Widerspruch zu der Annahme, dass das Netzwerk minimal ist.

Fall b) Die zweite Kante entspricht einer Kante k aus Abbildung 4.6: Dann gilt, dass die Kante die gleiche Länge wie die Kante b' hat, womit sich k durch b ersetzen lässt und sowohl die Kante k, als auch die Kante a gestrichen wird. Damit wird das Netzwerk kleiner, aber die Manhattan-Eigenschaft bleibt erhalten. Dies ist ein Widerspruch zu der Annahme, dass das Netzwerk minimal ist.

□

Den rechten Rand des Netzwerkes muss man nicht mehr betrachten. Sollte der Algorithmus die Kante c einfügen, so lässt sich diese nicht durch die Kante d ersetzen. Sollte er die Kante d einfügen, so ist diese auf jeden Fall maximal so groß wie die Kante c, so dass es in diesem Fall keinen Sinn macht, die Kante d durch die Kante c zu ersetzen.

Darauf aufbauend ergibt sich dann der endgültige Algorithmus 4.2, der ein minimales vertikales Manhattan-Netzwerk berechnet. Dieser Algorithmus berechnet mittels des Algorithmus 4.1 zwei vertikale Netze, eins mit der Startkante b und eins mit der Startkante a.

Algorithmus 4.2 Berechnung eines minimalen vertikalen Netzwerkes

Eingabe: Eine Punktmenge $P_v \subset S$, die die Punkte auf dem oberen und unteren Rand des Rechtecks enthält, exklusive der Eckpunkte des Rechtecks.

Ausgabe: Ein minimales Netzwerk M mit $\forall p, q \in P_v, y_p \neq y_p : \exists$ Manhattan-Pfad in $M \cup$ Kontur

1: Berechne mit Algorithmus 4.1 ein günstiges Netzwerk N_1. ▷ Dieses Netzwerk enthält die Kante b

2: Berechne mit Algorithmus 4.1 ein günstiges Netzwerk N_2, das jedoch abweichend im ersten Schritt die Kante a einfügt

3: Wähle min $\{|N_1|, |N_2|\}$

Theorem 4.7 *Der Algorithmus 4.2 berechnet ein vertikales Manhattan-Netzwerk minimaler Länge von Punkten auf einem Rechteck.*

Beweis Theorem 4.7 Annahme: Das so entstandene Netzwerk M ist nicht minimal. Dies bedeutet, es gibt ein Netzwerk M', mit $|M'| < |M|$. Aufgrund von Korollar 4.5 und Korollar 4.6 können wir ohne Beschränkung der Allgemeinheit annehmen, dass das Netzwerk M' als erste Kante die Kante a oder die Kante b enthält und als zweite Kante eine Kante rechts von der Kante b.

Vergleiche nun das Netzwerk M' mit dem Netzwerk N. Dabei entspricht N dem Netzwerk N_1 aus Algorithmus 4.2, wenn die erste Kante in M' die Kante b ist, bzw. dem Netzwerk N_2, wenn die erste Kante in M' die Kante a ist.

Damit enthalten die zu vergleichenden Netzwerke beide entweder die Kante a oder die Kante b. Da ein minimales Netzwerk nach Korollar 4.6 als zweite Kante eine Kante rechts von b enthalten muss, müssen wir damit nur noch die Kanten rechts von b betrachten. Relevant für die Längen der Netzwerke sind nur die Teile der Kanten, die nicht entlang

der Kontur verlaufen. Daher wird im Beweis mit $|l|_u$ die Länge der Kante l bezeichnet, die echt innerhalb der Kontur verläuft. Für $l \neq a, b, c, d$ gilt: $|l|_u = |l|$, für $l = a, b, c, d$ gilt $|l|_u = |a'|, |b'|, |c'|, |d'|$.

Jeder vertikale Kante l des Netzwerkes N wird genau eine Kante l' des minimalen Netzwerkes M' in Rechnung gestellt. Dabei wird jede Kante l' nur für eine Kante l verwendet. Weiterhin wird gezeigt, dass auch immer $|l|_u \leq |l'|_u$ gilt. Die Kanten l, die sowohl in N als auch in M' vorhanden sind, werden sich selbst in Rechnung gestellt. Damit müssen nur noch die Kanten l aus N betrachtet werden, die nicht in M' vorhanden sind.

Sei l eine Kante aus N, die nicht in M' vorhanden ist. Der Algorithmus hat diese Kante in N eingefügt, da sonst die Manhattan-Eigenschaft des Netzwerkes verletzt wäre. Damit muss es auch eine Kante l' in M' geben, die die Manhattan-Eigenschaft sicherstellt und die links von l liegt. Sei l'' die nächste Kante links in N. Da durch l'' in N die Manhattan-Eigenschaft nicht sichergestellt werden kann, muss l' zwischen l'' und l liegen. Wenn l weder der Kante c noch der Kante d entspricht, dann gilt $|l|_u = |l'|_u$. Wenn l der Kante c entspricht, dann gilt $|l|_u \leq |l'|_u$. Sollte l der Kante d entsprechen, dann kann l' der Kante c entsprechen. Da aber $|d|_u \leq |c|_u$ gilt, folgt daraus dass auch $|l|_u \leq |l'|_u$. Aufgrund der Wahl der Kanten l' kann auch keine Kante zweimal für eine Kante l gewählt werden. Damit gilt aber $|M| \leq |N| \leq |M'|$, was ein Widerspruch zu der Annahme $|M'| < |M|$ ist.

□

Der Algorithmus für die Konstruktion des minimalen horizontalen Netzwerkes geht analog. Nach Korollar 4.2 ist dann das minimale Manhattan-Netzwerk die Vereinigung der Kontur mit den horizontalen und vertikalen Netzwerken, wie in Algorithmus 4.3 beschrieben wird (vgl. Abbildung 4.7).

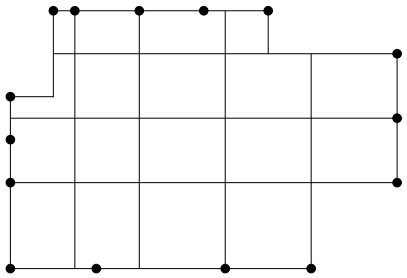


Abbildung 4.7: Minimales Manhattan-Netzwerk von Punkten auf einem Rechteck

4.3 Laufzeit

Die Laufzeiten für die einzelnen Teile des Algorithmus 4.3 sind jeweils:

Algorithmus 4.3 Berechnung eines minimalen Manhattan-Netzwerkes von Punkten auf einem Rechteck

Eingabe: Eine Punktmenge S, die auf einem Rechteck liegt.
Ausgabe: Ein minimales Manhattan-Netzwerk M zu der Punktmenge S

1: $P_l = \{p | p \in S, p \text{ liegt auf dem linken Rand}\}$
2: $P_r = \{p | p \in S, p \text{ liegt auf dem rechten Rand}\}$
3: $P_u = \{p | p \in S, p \text{ liegt auf dem unteren Rand}\}$
4: $P_o = \{p | p \in S, p \text{ liegt auf dem oberen Rand}\}$
5: K=Kontur(S)
6: P_v mit $S \setminus (P_l \cup P_r)$
7: N_1 = Berechnung eines minimalen vertikalen Netzwerkes(P_v)
8: P_v mit $S \setminus (P_u \cup P_o)$
9: N_2 = Berechnung eines minimalen horizontalen Netzwerkes(P_v)
10: $M = N_1 \cup N_2 \cup K$

Berechnung von P_l, P_r, P_u, P_o	$O(n)$
Berechnung der Kontur	$O(n \log n)$
Berechnung des minimalen vertikalen Netzwerkes	$O(n \log n)$
Berechnung des minimalen horizontalen Netzwerkes	$O(n \log n)$

Als Gesamtlaufzeit ergibt sich damit $O(n \log n)$.
Die einzelnen Laufzeiten bedürfen noch einer Erklärung:

Berechnung von P_l, P_r, P_u, P_o: Diese Mengen, die jeweils die Punkte auf dem linken, rechten, unteren oder oberen Rand enthalten, lassen sich dadurch berechnen, indem man jeden Punkt genau einmal betrachtet und jeweils die minimalen oder maximalen Punkte in x- oder y-Richtung speichert. Bei der Berechnung von P_l würde man beispielsweise die Punkte mit minimalen x-Koordinaten speichern. Da man jeden Punkt nur einmal betrachten, aber vorher nicht sortieren muss, ergibt sich als Laufzeit $O(n)$ für diesen Schritt.

Berechnung der Kontur: Das Verfahren zur Konstruktion der Kontur ist dem Buch von Rolf Klein [Kle05, Seite 167ff] entnommen, die Modifizierung auf Seite 15 ändert nichts an der Laufzeit von $O(n \log n)$. Sollten die Punkte dabei bereits nach ihren x-Koordinaten sortiert sein, so sinkt die Laufzeit auf $O(n)$.

Berechnung des minimalen vertikalen Netzwerkes: Hierzu müssen die Punkte zuerst nach x-Koordinaten sortiert werden. Danach werden zwei Sweeps mit $O(n)$ Ereignissen durchgeführt. Jedes Ereignis verursacht Kosten von $O(1)$, so dass die Laufzeit des Sweep-Algorithmus mit dem Sortieren zusammen $O(n \log n)$ beträgt. Sollten auch hier die Punkte nach x-Koordinaten vorsortiert sein, so sinkt die Laufzeit auf $O(n)$.

Berechnung des minimalen horizontalen Netzwerkes: Hierzu müssen die Punkte zuerst nach y-Koordinaten sortiert werden. Ansonsten bleibt die Betrachtung für die Be-

rechnung des minimalen horizontalen Netzwerkes gültig, so dass sich mit dem Sortieren eine Laufzeit von $O(n \log n)$ ergibt. Sollten Punkte hier nach y-Koordinaten vorsortiert sein, so sinkt die Laufzeit auf $O(n)$.

Gesamtlaufzeit Sollten die Punkte sowohl nach x-, als auch nach y-Koordinaten sortiert sein, lässt sich eine Laufzeit von $O(n)$ erreichen. Dies ist in diesem Fall auch nicht unrealistisch, da bereits eine Bedingung an die Punktmenge gestellt wird. Die Punkte müssen alle auf einem Rechteck liegen. Sollte man die Punkte in der Reihenfolge ihres Auftretens auf dem Rechteck bekommen, so lassen sie sich in Zeit $O(n)$ sortieren.

5 Ausblick und offene Probleme

Das minimale Manhattan-Netzwerk-Problem bietet noch viele offene Probleme, die noch nicht vollständig erforscht sind. Im Folgenden werden ein paar dieser Probleme aufgezählt.

5.1 Komplexität

Es ist bis jetzt noch nicht bekannt, ob das minimale Manhattan-Netzwerk Problem NP-vollständig ist. Exakte Lösungen lassen sich bis jetzt nur für verhältnismäßig einfache Punktmengen in polynomieller Zeit berechnen. Für beliebige Punktmengen existieren bis jetzt nur Approximations-Algorithmen. Der beste Approximations-Algorithmus, der bis jetzt bewiesen ist, von Chepoi, Nouioua und Vaxès [CNV05] garantiert einen Faktor von zwei und wurde in Kapitel 3.3 vorgestellt. Für den 3-Approximations-Algorithmus von Benkert, Widmann und Wolff [BWW04], der in Kapitel 3.2 vorgestellt wurde, gilt jedoch, dass dieser nur in speziell konstruierten Fällen dem Faktor drei beliebig nahe kommt. Für zufällige Punktmengen jedoch liegt der erreichte Faktor nahe bei eins.

Die Komplexität dieses Problems erwächst unter anderem daraus, dass ein neuer Punkt in einem minimalen Manhattan-Netzwerk einen Großteil der Pfade beeinflussen kann. Dadurch lässt sich aus einer Unterteilung in Teilprobleme, wie sie von allen Approximations-Algorithmen vorgenommen wird, meistens keine optimale Lösung konstruieren oder die Komplexität sinkt durch die Aufteilung nicht.

5.2 F-beschränktes Manhattan-Netzwerk-Problem

Beim F-beschränkten Manhattan-Netzwerk-Problem muss nicht zu jedem Paar aus $S \times S$ ein Manhattan-Pfad konstruiert werden, sondern nur zu einer Teilmenge $F \subseteq S \times S$. Dieses Problem kann offensichtlich nicht leichter sein als das unbeschränkte Manhattan-Netzwerk Problem, da das normale Manhattan-Netzwerk-Problem ein Spezialfall des beschränkten Problems ist. Wenn $F = S \times S$ gilt oder wenn F eine erzeugende Menge ist, erhält man das originale minimale Manhattan-Netzwerk Problem. Allerdings lässt sich die Komplexität des F-beschränkten Manhattan-Netzwerk Problems angeben. Dieses Problem ist NP-vollständig, da es das Rectlinear Steiner Arborescene Problem verallgemeinert, Shi und Su [SS00] gezeigt haben. Bei dem Rectlinear Steiner Arborescene Problem handelt es sich um das Problem, eine Menge S von n Punkten, die alle im ersten Quadranten liegen, auf rechtwinkligen Pfade mit dem Ursprung zu verbinden. Dabei soll die Gesamtlänge dieser Pfade minimiert werden. Dies ist äquivalent dazu, dass F-beschränkte Manhattan-Netzwerk Problem für die Menge $S' = S \cup \{o\}$ und $F = \{o\} \times S$

zu lösen, wobei o dem Ursprung entspricht.

5.3 Real Manhattan-Problem

Bei dem Real Manhattan-Problem müssen, genauso wie bei dem normalen minimalen Manhattan-Netzwerk-Problem, n Punkte untereinander verbunden werden. Allerdings ist diesmal das Gitter vorgegeben und die Punkte können auch auf den Kanten des Gitters sitzen. Damit sind die kürzesten Pfade zwischen zwei Punkten unter Umständen länger als die Manhattan-Distanz zwischen ihnen (vgl. Abbildung 5.1).Dieses Problem

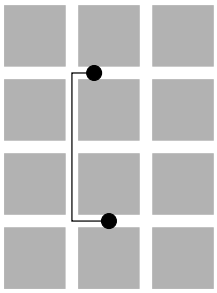


Abbildung 5.1: Kürzester Pfad zwischen zwei Punkten im Real Manhattan Problem.

wurde von Chepoi, Nouioua und Vaxès [CNV05] kurz vorgestellt, ohne jedoch näher darauf einzugehen.

5.4 Ausblick

Das minimale Manhattan-Netzwerk Problem lässt sich in verschiedene Richtungen weiter betrachten. Interessant wäre es, die Arbeit aus Kapitel 4 fortzuführen und weitere Punktmengen zu charakterisieren, für die sich minimale Manhattan-Netzwerke in polynomieller Zeit berechnen lassen. Dabei lassen sich unter Umständen interessante Eigenschaften von minimalen Manhattan-Netzwerken charakterisieren, die bessere Approximations-Algorithmen liefern oder Aussagen über die Komplexität erlaubten. Am einfachsten sind vermutlich dabei Punktmengen auf dem Rand von einfachen Polygonen.

Eine andere Richtung wäre, schnellere oder bessere Approximations-Algorithmen zu finden. So stellt sich die Frage, wie ein schnellerer Faktor-2-Algorithmus aussehen kann oder ob es einen polynomiellen Approximations-Algorithmus mit einem Faktor unterhalb von zwei gibt. Dafür lassen sich verschiedene Ansätze, wie Divide & Conquer betrachten.

Literaturverzeichnis

[BM02] Bose, Prosenjit und Pat Morin (Herausgeber): *Algorithms and Computation, 13th International Symposium, ISAAC 2002 Vancouver, BC, Canada, November 21-23, 2002, Proceedings*, Band 2518 der Reihe *Lecture Notes in Computer Science*. Springer, 2002.

[BWW04] Benkert, Marc, Florian Widmann und Alexander Wolff: *The Minimum Manhattan Network Problem: A Fast Factor-3 Approximation*. In: *Abstracts 8th Japanese Conf. on Discrete and Computational Geometry (JCDCG'04)*, Seiten 85–86, 8–11 Oktober 2004.

[BWWS06] Benkert, Marc, Alexander Wolff, Florian Widmann und Takeshi Shirabe: *The Minimum Manhattan Network Problem: Approximations and Exact Solution*. 2006.

[CFK81] Chalmet, L.G., R.L. Francis und A. Kolen: *Finding Efficient Solutions for Rectilinear Distance Location Problems Efficiently*. European Journal of Operational Research, 6:117–124, 1981.

[CJRT05] Chekuri, Chandra, Klaus Jansen, José D. P. Rolim und Luca Trevisan (Herausgeber): *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th International Workshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, Band 3624 der Reihe *Lecture Notes in Computer Science*. Springer, 2005.

[CNV05] Chepoi, Victor, Karim Nouioua und Yann Vaxès: *A Rounding Algorithm for Approximating Minimum Manhattan Networks.* In: Chekuri, Chandra et al. [CJRT05], Seiten 40–51.

[DD05] Deng, Xiaotie und Ding-Zhu Du (Herausgeber): *Algorithms and Computation, 16th International Symposium, ISAAC 2005, Sanya, Hainan, China, December 19-21, 2005, Proceedings*, Band 3827 der Reihe *Lecture Notes in Computer Science*. Springer, 2005.

[GLN01] Gudmundsson, Joachim, Christos Levcopoulos und Giri Narasimhan: *Approximating a minimum Manhattan network*. Nordic J. of Computing, 8(2):219–232, 2001.

[KIA02] Kato, Ryo, Keiko Imai und Takao Asano: *An Improved Algorithm for the Minimum Manhattan Network Problem.* In: Bose, Prosenjit und Pat Morin [BM02], Seiten 344–356.

[Kle05] Klein, Rolf: *Algorithmische Geometrie – Grundlagen, Methoden, Anwendungen.* eXamen.press. Springer, 2. vollständig überarbeitete Auflage, 2005.

[LAP03] Lam, F., M. Alexandersson und L. Pachter: *Picking Alignments from (Steiner) Trees.* Journal of Computational Biology, 10(3-4):509–520, 2003.

[LÖ96] Levcopoulos, Christos und Anna Östlin: *Linear-Time Heuristics for Minimum Weight Rectangulation (Extended Abstract).* In: *Scandinavian Workshop on Algorithm Theory*, Seiten 271–283, 1996.

[LPRS82] Lingas, A., R. Pinter, R. Rivest und A. Shamir: *Minimum edge length partitioning of rectlinear polygons.* In: *Proc. 20th Allerton Conf. Commun. Control Comput.*, Seiten 53–63, 1982.

[SS00] Shi, W. und C. Su: *The rectilinear Steiner arborescence problem is NP-complete.* Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms, Seiten 780–787, 2000.

[SU05] Seibert, Sebastian und Walter Unger: *A 1.5-Approximation of the Minimal Manhattan Network Problem.* In: Deng, Xiaotie und Ding-Zhu Du [DD05], Seiten 246–255.

[Tar86] Tardos, Eva: *A strongly polynomial algorithm to solve combinatorial linear programs.* Oper. Res., 34(2):250–256, 1986.

[WHL73] Wendell, RE, AP Hurter und TJ Lowe: *Efficient points in location theory.* AIEE Transactions, 9:238–246, 1973.

Index