

Aroras PTAS für das euklidische TSP
Seminar: Perlen der theoretischen Informatik
AG Meyer auf der Heide
Wintersemester 04/05

Tim Süß
Matrikelnummer: 6089370
Betreuer: Gereon Frahling

17. Januar 2005

Zusammenfassung

Ein Händler will n Städte besuchen. Ihm sind die geographischen Positionen der Städte und die Abstände zwischen den einzelnen Städten bekannt. Der Handlungsreisende sucht eine möglichst kurze Tour, die jede Stadt nur genau einmal besucht und will am Ende wieder dort ankommen, wo er gestartet ist. Dies ist die informale Beschreibung des *Traveling Salesman Problems*. Dieses und ähnliche Probleme beflügelten lange Zeit die Mathematik und hier insbesondere die Graphentheorie. Seit den 70er Jahren, nach dem Aufkommen der Komplexitäts-Theorie, ist bekannt, dass das TSP *NP-Vollständig* ist und somit eine optimale Lösung nicht in polynomieller Zeit berechnet werden kann, wenn $P \neq NP$. Es gibt Heuristiken, die in polynomieller Zeit eine Lösung finden können, die nur wenige Prozent vom Optimum abweicht. Diese Verfahren können jedoch keine feste Güte des Ergebnisses garantieren; Approximationsalgorithmen haben diese Eigenschaft. Ein Approximationsalgorithmus kann in polynomieller Zeit eine Lösung berechnen, die nur um einen festen Faktor von einer optimalen Lösung abweicht. Sahni und Gonzales konnten 1976 jedoch zeigen, dass unter der Annahme $P \neq NP$ für das allgemeine TSP kein effizienter Approximationsalgorithmus existiert. Im gleichen Jahr stellte Christofides ein Verfahren vor, welches das metrische TSP mit dem Faktor $\frac{3}{2}$ annähert. In einem metrischen System kann eine l_p -Norm gegeben sein (dies wird zum Beweis des Struktur-Theorems benötigt). Die Punkte $x, y \in \mathbb{R}^d$ sind gegeben durch ihre Koordinaten $(x_1, \dots, x_d), (y_1, \dots, y_d)$. Die Zwischenknotendistanz der Punkte in einer l_p -Norm wird bestimmt durch: $\sum_{i=1}^d (|x_i - y_i|^p)^{\frac{1}{p}}$.

Das euklidische TSP ist ein Spezialfall des metrischen TSPs, es gilt die l_2 -Norm. 1976 konnte nicht beantwortet werden, ob für diese Spezialfälle ein Polynomial Time Approximation Scheme, kurz PTAS, existiert. Polynomial Time Approximation Schemes sind eine Algorithmenfamilie, deren Elemente eine approximierte Lösung liefern, die beliebig nahe an der optimalen Lösung liegt. Arora, Lund, Motwani, Sudan und Szegedy konnten nachweisen, dass für das metrische TSP kein PTAS existiert.

1996 stellten Arora und Mitchell unabhängig voneinander je ein PTAS für das euklidische TSP vor. Die beiden Verfahren liefern eine Lösung, die für ein festes $c > 2$ um den Faktor $1 + \frac{1}{c}$ von einer optimalen Lösung abweichen. Das Verfahren von Arora ist allgemeiner und kann für beliebig fest dimensionierte Räume genutzt werden. Die Laufzeit der randomisierten Version beträgt im \mathbb{R}^2 $O(n(\log n)^{O(c)})$ und im \mathbb{R}^d $O(n(\log n)^{O(\sqrt{d}c)^{d-1}})$. Der Algorithmus kann derandomisiert werden, dabei erhöht sich die Laufzeit um $O(n^d)$.

Aroras PTAS kann durch leichte Modifikationen so verändert werden, dass das Verfahren für viele andere Probleme wie *Minimum Steiner Tree*, *k-TSP* oder *k-Minimum Steiner Tree* eingesetzt werden kann. Für das *k-TSP* und das *k-MST* erhöht sich dabei die Laufzeit des PTAS um $O(k)$. Anzumerken ist, dass die Laufzeit eines PTAS exponentiell in c sein darf, jedoch nicht in der Eingabe.

In dieser Arbeit wird Aroras Verfahren im \mathbb{R}^2 beschrieben o.B.d.A..

1 Überblick

Aroras PTAS basiert auf der rekursiven Partitionierung des Raumes, einer TSP-Instanz, in einem randomisierten *Quadtree*. Es wird eine approximierte Rundreise berechnet, die nur um einen Faktor $1 + \frac{1}{c}$ von einer optimalen Rundreise abweicht. Dabei wird jede Begrenzung einer Zelle höchstens $r = O(c)$ -mal geschnitten. Für jede Zelle der Instanz „rät“ der Algorithmus unabhängig von den anderen Zellen, wo die Rundreise die Seiten schneidet. Dies geschieht durch dynamische Programmierung. Die Partitionierung des Raumes in einen *Quadtree* enthält $n \log n$ verschiedene Quadrate. Für jedes Quadrat benötigt der Algorithmus im worst-case $O(\log n)^{O(r)} = O(\log n)^{O(c)}$ zum „Raten“ des Pfades. Somit ergibt sich eine Gesamtlaufzeit von $O(n \log n)^{O(c)}$.

Im zweiten Abschnitt dieser Arbeit wird Aroras Algorithmus beschrieben. Dieser kann in drei Bereiche unterteilt werden: Normierung der TSP-Instanz, Partitionierung in einen randomisiert verschobenen *Quadtree* und Berechnung der minimalen Teillösungen durch dynamische Programmierung. In Abschnitt drei wird das *Struktur-Theorem* bewiesen, welches besagt, dass für einen zufällig verschobenen *Quadtree* mit Wahrscheinlichkeit $> \frac{1}{2}$ eine $(1 + \frac{1}{c})$ -Approximation der optimalen Rundreise existiert. Hierbei wird jedes Quadrat der Partitionierung höchstens $O(c)$ -mal an festgelegten Punkten durchkreuzt. Zuletzt wird in Abschnitt vier die Anwendbarkeit des Algorithmus untersucht.

2 Der Algorithmus

Sei I eine TSP-Instanz mit n Knoten und opt_I die optimale Rundreise von I . Das kleinst mögliche Quadrat, das alle Knoten von I umspannt, bezeichnen wir als *exakte Bounding-Box*. Die Seitenlänge dieses Quadrates wird als L bezeichnet. Es kann o.B.d.A. angenommen werden, dass die untere linke Ecke der Bounding-Box im Ursprung des Koordinatensystems liegt. Dies kann im Allgemeinen durch eine Translation erreicht werden. Die Punkte der Bounding-Box liegen dann alle in $[0, L]^2$. Eine *Dissection* ist eine rekursive Aufteilung der Bounding-Box in gleichgroße Quadrate, bis jedes einzelne Quadrat nur noch höchstens einen Knoten enthält. Dies wird erreicht, indem die Bounding-Box,

die Wurzel der Dissection, entlang der Mitte in vier *Kindquadrate* unterteilt wird. Die Kindquadrate haben damit die Seitenlänge $\frac{L}{2}$. Befinden sich in einem oder mehreren Quadraten mehr als ein Knoten, so werden alle wiederum entlang ihrer Mitte geteilt (Seitenlänge $\frac{L}{4}$). Dies wird solange fortgesetzt, bis alle Punkte separiert sind.

Ein *Quadtree* ist ähnlich definiert. Die weitere Unterteilung eines Kindquadrates wird jedoch gestoppt, wenn es nur noch höchstens einen Knoten beinhaltet. Im Allgemeinen besitzt ein Quadtree weniger Quadrate als eine Dissection.

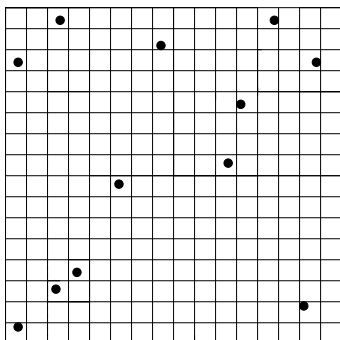


Abb.:2.1 Dissection

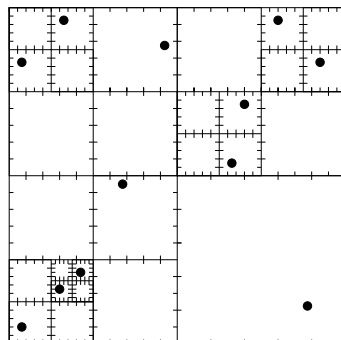


Abb.:2.2 Quadtree

Durch die Struktur des Quadtree und der Dissection kann man jedem Quadrat einen *Level* zuordnen. Die Wurzel der Dissection oder des Quadtree, also die Bounding-Box selber, hat Level 0. Die vier Kindquadrate, also die Quadrate mit Seitenlänge $\frac{L}{2}$, haben Level 1 usw. Die Tiefe der verschiedenen Partitionierungen entspricht der größten Levelnummer. Abbildung 2.1 und Abbildung 2.2 haben somit Tiefe 4.

Seien a, b ganzzahlig, $0 \leq a, b < L$, dann ist eine (a, b) -verschobene Dissection dadurch definiert, dass alle x-Koordinaten um a und alle y-Koordinaten um b modular verschoben sind. Daraus ergibt sich, dass die mittlere senkrechte Linie $\frac{L}{2}$ der Dissection auf $a + \frac{L}{2} \pmod{L}$ und die mittlere waagerechte Linie $\frac{L}{2}$ auf $b + \frac{L}{2} \pmod{L}$ verschoben wird. Der verschobene Quadtree lässt sich leicht aus der verschobenen Dissection berechnen. Im Allgemeinen ist der verschobene Quadtree von anderer Struktur als die ursprüngliche Unterteilung der Bounding-Box. In Abbildung 2.4 wird ein $(5, 5)$ verschobener Quadtree gezeigt.

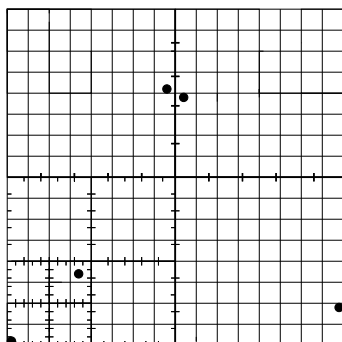


Abb.:2.3 Nicht verschobener Quadtree

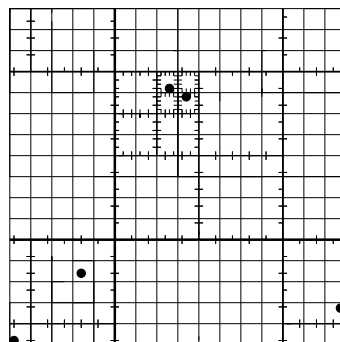


Abb.:2.4 Verschobener Quadtree

Der Algorithmus basiert auf dem Struktur-Theorem. Dieses besagt, dass für ein festes c in einem verschobenen Quadtree mit Wahrscheinlichkeit $\geq \frac{1}{2}$ eine Rundreise existiert, die maximal $(1 + \frac{1}{c})$ -mal länger ist, als die optimale Tour und die jedes Quadrat des Quadrates höchstens $O(c)$ -mal in Portalen schneidet. Portale sind Punkte auf der Seite eines Quadrates, an denen ein Quadrat betreten und verlassen werden darf. Es wird dem Handlungsreisenden erlaubt, auf dem Weg zwischen zwei Punkten vom geradlinigen Pfad abzuweichen, dadurch ist der Weg zwischen zwei Punkten eventuell gebogen. Eine so gebogene Rundreise bezeichnet man als *Salesman Pfad*. Ein solcher Pfad wird durch den hier beschriebenen Algorithmus berechnet. Dieser gebogene Weg kann im Nachhinein begradigt werden; da im euklidischen TSP die Dreiecksungleichung gilt, erhöhen sich dadurch nicht die Gesamtkosten.

Definition: Seien $m, r \in \mathbb{N}$. Jedes Quadrat eines Quadrates hat in jeder der vier Ecken ein und auf jeder der vier Kanten m Portale mit gleichem Abstand. Ein Salesman Pfad ist (m, r) -leicht, wenn jedes Quadrat des verschobenen Quadrates höchstens r -mal an jeder Kante geschnitten wird und jede Durchkreuzung an einem Portal stattfindet. Jedes Quadrat hat auf jeder Seite $m + 2$ Portale. Daraus folgt, dass Quadrate im gleichen Level an den gleichen Stellen Portale besitzen. Ist ein Quadrat von Level i mit einem Quadrat von Level $i - 1$ benachbart, so kommt es bei dem Quadrat von Level i nur in jedem zweiten Portal zur Deckung (siehe Abb. 2.3). Portale können mehrfach durchlaufen werden. Laut Definition eines (m, r) -leichten Salesman Pfades dürfen Quadrate nur an Portalen betreten und verlassen werden. Daher können einige Portale bei benachbarten Quadraten unterschiedlicher Größe nicht genutzt werden.

Struktur-Theorem: Sei $c > 1$ eine reelle Zahl. Sei der Abstand zweier Punkte einer TSP-Instanz I entweder 0 oder mindestens 8 und sei L die Größe der Bounding-Box zu I . Weiter seien $0 \leq a, b < L$ zufällig gewählte ganzzahlige Verschiebungen. Es gibt Zahlen $m = O(c \log L)$ und $r = O(c)$, sodass mit Wahrscheinlichkeit $\geq \frac{1}{2}$ in dem (a, b) verschobenen Quadtree ein (m, r) -leichter Salesman Pfad existiert, der höchstens Kosten von $(1 + \frac{1}{c})opt_I$ hat.

Für die Beschreibung des Algorithmus gehen wir von der Gültigkeit des Struktur-Theorems aus.

2.1 Normierung der Eingabe

Bei der Normierung der TSP-Instanz I will man drei Dinge erreichen:

1. Alle Punktkoordinaten sind ganzzahlig
2. Der Abstand zweier Punkte ist genau 0 oder mindestens 8
3. Der maximale Punktabstand ist $O(n)$

Sei L_0 die Seitenlänge der Bounding-Box von I und opt_I die optimalen Rundreisekosten. Es gilt $opt_I > L_0$. Ein Gitter mit der Auflösung $d := \frac{L_0}{8nc}$ wird in die Bounding-Box gelegt und jeder Punkt (x_i, y_i) auf den naheliegenden Gitterpunkt verschoben: $(x'_i, y'_i) = (\lfloor x_i + \frac{1}{2} \cdot \frac{L_0}{8nc} \rfloor, \lfloor y_i + \frac{1}{2} \cdot \frac{L_0}{8nc} \rfloor) = (d \lfloor \frac{x_i}{d} + \frac{1}{2} \rfloor, d \lfloor \frac{y_i}{d} + \frac{1}{2} \rfloor)$. Es kann passieren, dass mehrere Punkte auf einen Gitterpunkt zusammenfallen. Dadurch haben diese Punkte eine Zwischenpunktdistanz von 0, dies ist durch das Struktur-Theorem erlaubt.

Die verschobenen Punkte werden durch $\frac{L_0}{64nc}$ geteilt (oder mit $\frac{8}{d}$ multipliziert). Hieraus ergibt sich, dass der Abstand zwischen zwei Punkten, die auf unterschiedlichen Gitterpunkten liegen mindestens 8 ist und die Punktkoordinaten ganzzahlig sind. Das Verschieben der Punkte, die auf der oberen und rechten Seite der exakten Bounding-Box liegen, verschiebt die obere und die linke Seite der Bounding-Box um maximal $\frac{d}{2}$. Die Skalierung ergibt die neue Bounding-Box mit Seitenlänge L .

$$L \leq (L_0 + \frac{d}{2}) \cdot \frac{8}{d} = 64nc + 4 = O(nc) = O(n)$$

Damit erfüllt die Normierung die drei geforderten Punkte. Durch die Normierung muss eine $(1 + \frac{3}{4c})$ -Approximation berechnet werden, dies ist jedoch zu vernachlässigen, wenn $c > 1$ eine beliebige Konstante repräsentiert.

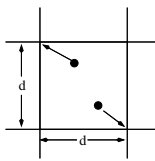


Abb. 2.5: Verschieben auf den naheliegenden Gitterpunkt

2.2 Konstruktion des verschobenen Quadrees

Aus der Dissection kann ein zufällig (a, b) -geshielter Quadtree berechnet werden, zum Beispiel mit einem *Sorting-Base*-Algorithmus, der Laufzeit $O(n \log^2 n)$ hat. Die Bounding-Box der normierten TSP-Instanz hat die Länge $L = O(n)$. Daraus ergibt sich, dass die Tiefe des Quadrees $O(\log n)$ und die Anzahl der Quadrate des Quadrees durch $T = O(n \log(n))$ beschränkt ist. Die Beschränkung der Anzahl der Quadrate ergibt sich dadurch, dass in jedem Level des verschobenen Quadrees höchstens $2n$ Quadrate sind.

2.3 Dynamisches Programm

Durch dynamische Programmierung soll ein (m, r) -leichter Salesman Pfad im verschobenen Quadtree gefunden werden. Es gilt, $m = O(c \log n)$ und $r = O(c)$. Mit Wahrscheinlichkeit $\geq \frac{1}{2}$ sind die Kosten von diesem Pfad maximal $O(1 + \frac{3}{4c}) \text{opt}_T$. Die Laufzeit des dynamischen Programms beträgt $O(T \cdot (m)^{O(r)})$, also $O(n \log n)^{O(c)}$.

Das Programm nutzt folgende Beobachtung: Sei Q ein Quadrat des verschobenen Quadrees und der optimale (m, r) -leichte Salesman Pfad π schneidet die Grenzen von Q insgesamt $2p \leq 4r$ mal. Sei a_1, a_2, \dots, a_{2p} eine Sequenz von Portalen, in der die Seiten von Q durchkreuzt werden. Der Teil des optimalen Salesman Pfades in Q ist dann eine Sequenz von p Pfaden für die gilt:

- i) Für $i = 1, 2, \dots, p$ verbindet der i -te Pfad die Portale a_{2i-1} mit a_{2i}
- ii) Alle p Pfade zusammen besuchen alle Punkte die in Q liegen
- iii) Die Zusammenfassung der p Pfade ist (m, r) -leicht, d.h. dass sie insgesamt jede Kante jedes Unterquadrates in Q höchstens r -mal schneiden und dass diese Schnitte in Portalen passieren.

Da π optimal ist, muss die beschriebene Sequenz von Pfaden kostenminimal, unter allen Pfaden, die i), ii) und iii) erfüllen, sein. Durch diese Beobachtung lässt sich das (m, r) -Multitour Problem definieren. Eine Instanz dieses Problems ist definiert wie folgt:

- a) Ein nicht leeres Quadrat des verschobenen Quadrtrees
- b) Eine Multimenge von r Portalen an jeder der vier Seiten des Quadrates, sodass die Summe der Kardinalitäten der vier Multimengen eine gerade Zahl $2p \leq 4r$ ist
- c) Eine Paarung $(a_1, a_2), (a_3, a_4), \dots, (a_{2p-1}, a_{2p})$ aus den $2p$ Portalen aus b)

Ziel des (m, r) -Multitour Problems ist es, eine Verbindung aus p Pfaden in dem Quadrat zu finden, die (m, r) -leicht und kostenminimal sind. Der i -te Pfad, $0 \leq i \leq p$, soll dabei Portal a_{2i-1} mit Portal a_{2i} verbinden und alle Pfade zusammen sollen jeden Punkt des Quadrates besuchen. Für $p = 0$ wird ein optimaler (m, r) -leichter Salesman Pfad für die Punkte innerhalb des Quadrates gesucht. Daraus folgt, dass der gesuchte optimale (m, r) -leichte Salesman Pfad der Lösung des (m, r) -Multitour Problems mit Eingabe Bounding-Box und $p = 0$ entspricht. Die Lösung des (m, r) -Multitour Problems für einen Level i des Quadrtrees wird aus den Lösungen der Level $i + 1$ berechnet. Das dynamische Programm erstellt eine *Lookup-Tabelle*, welche die optimalen Kosten aller Instanzen des (m, r) -Multitour Problems aus dem verschobenen Quadtree enthält. Das Programm terminiert, sobald der optimale (m, r) -leichte Salesman Pfad in der Tabelle enthalten ist. Dies ist der Wert der Wurzel des Quadrtrees mit $p = 0$. Die Anzahl der Einträge in einem Quadtree mit T nicht leeren Quadraten ist beschränkt durch $O(T \cdot (m + 4)^{4r} \cdot (4r)!)$, da es für jedes der T nicht leeren Quadrate $(m + 4)^{4r}$ Möglichkeiten gibt eine Multimenge von Portalen zu wählen und maximal $(4r)!$ Möglichkeiten Portalpaare zu bilden.

Die Tabelle wird beginnend beim höchsten Level, also beginnend bei den kleinsten Quadraten, zum Level 0 aufgebaut. Blätter des Quadtree enthalten höchstens einen Punkt und $O(r)$ Portale. Um eine (m, r) -Multitour in einem Blatt mit genau einem Punkt zu bestimmen, muss dieser probeweise auf jeden der $O(r)$ Pfade platziert werden, um den Pfad zu finden, der die niedrigsten Kosten hat. Hat der Algorithmus alle (m, r) -Multitour Probleme des Levels $i + 1$ gelöst, verwendet er diese Ergebnisse, um das (m, r) -Multitour Problem des Levels i zu lösen. Sei Q ein Quadrat des Levels i , das kein Blatt des verschobenen Quadrtrees ist. Seien Q_1, Q_2, Q_3, Q_4 die Kindquadrate von Q , deren Level $i + 1$ ist. Bei jeder Wahl der Eingabe für b) und c) zu Q zählt der Algorithmus alle Möglichkeiten auf, in denen eine (m, r) -Multitour die vier inneren Kanten von Q_1, \dots, Q_4 durchkreuzen kann. Das beinhaltet alle Wahlen der folgenden Möglichkeiten:

- a') Eine Multimenge von $\leq r$ Portalen auf jeder der vier inneren Seiten der Kindquadrate von Q
- b') Eine Reihenfolge in der die Portale, die in a') bestimmt wurden, von den Pfaden, die durch b) und c) festgelegt worden sind, durchkreuzt werden

Die Anzahl der möglichen Wahlen in a') ist höchstens $((m + 4)^r)^4$ und in b') maximal $(4r)^{4r}(4r)!$. Für b') entspricht $(4r)^{4r}$ der oberen Schranke der Wegwahlmöglichkeiten der Wege die in a') ausgesucht wurden.

Jede Wahl in a') und b') führt zu (m, r) -Multitour Problemen der vier Kindquadrate, deren optimale Lösungen in der Lookup-Tabelle gespeichert sind.

Die Laufzeit ist somit:

$$\begin{aligned} O(T \cdot (m+4)^{8r} \cdot (4r)^{4r} \cdot ((4r)!)^2) &= O(T \cdot (m+4)^{8r} \cdot (4r)^{4r} \cdot (4r)^{2r}) \\ &= O(T \cdot O(c \log n)^{O(c)} \cdot O(c)^{O(c)} \cdot O(c)^{O(c)}) = O(n(\log n)^{O(c)}) \end{aligned}$$

Es muss noch betrachtet werden, wie der Algorithmus Quadrate verarbeitet, die durch das zufällige Verschieben nicht zusammenhängend sind. Bei solchen Quadraten ist zu beobachten, dass die Anzahl der zu untersuchenden Wahlen geringer ist, als bei zusammenhängenden Quadraten, da sich die (m, r) -Multitour nur in einem der zwei oder vier Teilquadrate befinden kann. Somit brauchen nicht alle Wahlen aus b) und c) betrachtet werden, sondern nur die, die zu gültigen Eingabeinstanzen führen.

3 Beweis des Struktur-Theorems

Um das Struktur-Theorem zu beweisen, benötigen wir zwei Lemma. Das Patching Lemma zeigt, dass die Anzahl der Schnittpunkte eines Pfades mit einem Liniensegment auf maximal zwei Schnitte reduziert werden kann. Das zweite Lemma beschreibt, wie die Länge einer Rundreise durch die Anzahl der Schnitte mit einem Gitter ausgedrückt werden kann.

3.1 Patching-Lemma

3.1.1 Lemma

1. Sei S ein beliebiges Liniensegment der Länge l und π ein beliebiger geschlossener Pfad, der S mindestens dreimal durchkreuzt. Dann gibt es auf S Linienstücke, deren Gesamtlänge höchstens $2l$ ist, sodass die Erweiterung von π um diese Linienstücke einen geschlossenen Pfad π' ergibt, der S höchstens zweimal durchkreuzt.
2. Es gibt einen Pfad $\tilde{\pi}$ mit Eigenschaften aus 1), zu dem die Erweiterung mit Linienstücken auf S mindestens Gesamtlänge $2l$ haben muss.

Beweis: Sei t die Anzahl der Schnitte von π mit S und seien M_1, M_2, \dots, M_t diese Schnittpunkte. Eine Zerlegung von π an diesen Punkten ergibt die Pfadabschnitte P_1, P_2, \dots, P_t . Wir konstruieren einen Hilfsgraph G mit $2t$ Knoten, $M1 = \{M'_1, M'_2, \dots, M'_t\}$ und $M2 = \{M''_1, M''_2, \dots, M''_t\}$. Die Schnittpunkte M_i aus π werden dazu durch Kanten mit je zwei Knoten M'_i und M''_i ersetzt. Die Knotenmengen $M1$ und $M2$ sind dabei durch S getrennt, d.h. dass zum Beispiel alle Elemente aus $M1$ links und alle Elemente aus $M2$ rechts von S liegen. Die Knoten M'_i und $M''_j, i \neq j$, werden miteinander verbunden, wenn in π eine entsprechende Kante vorhanden ist. Durch den Hilfsgraph G wird gezeigt werden, dass π so verändert werden kann, dass S höchstens zweimal geschnitten wird.

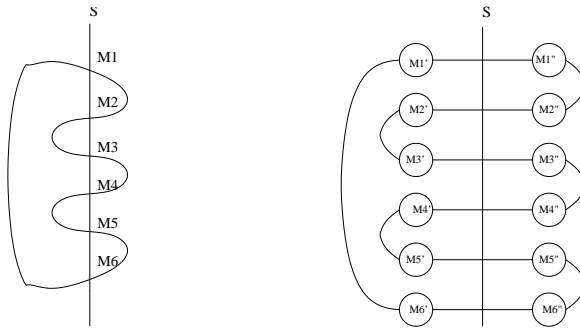


Abb. 3.1: Ursprungsgraph Abb. 3.2: Hilfsgraph mit eingefügten Punkten

Da π ein zusammenhängender und geschlossener Graph ist, ist auch der Hilfsgraph G zusammenhängend und geschlossen. Da jeder Knoten Grad 2 hat, sind G und π *Eulerkreise*, die das Liniensegment S genau t -mal schneiden (siehe Abb. 3.1 und Abb. 3.2). Durch den Hilfsgraph lässt sich ein Pfad π' erstellen, der ein geschlossener Eulerkreis ist und der das Liniensegment S höchstens zweimal schneidet. Hierzu lässt sich ein Algorithmus angeben.

1. $i := 1$
2. Entferne Kante $\{M'_i, M''_i\}$ aus G
3. Füge die Kanten $\{M'_i, M'_{i+1}\}$ und $\{M''_i, M''_{i+1}\}$ in G ein
4. Entferne Kante $\{M'_{i+1}, M''_{i+1}\}$ aus G
5. Verbinde die maximal zwei Komponenten durch die Doppelkante $\{M'_{i+1}, M'_{i+2}\}$ bzw. $\{M''_{i+1}, M''_{i+2}\}$
6. $i := i + 2$
7. Falls $i \leq t - 2$ gehe zu 2), sonst Stop

Zunächst wird gezeigt, dass der Algorithmus einen Eulerkreis berechnet, der S höchstens zweimal schneidet und π um maximal $2l$ verlängert.

Nach Durchführung von Schritt 2 ist der Pfad immer noch zusammenhängend, da er vorher ein Eulerkreis war und nur eine Kante entfernt wurde. Die Knoten M'_i und M''_i sind jetzt von ungeradem Grad. Durch das Einfügen der Kanten $\{M'_i, M'_{i+1}\}$ und $\{M''_i, M''_{i+1}\}$ haben die beiden Knoten wieder geraden Grad, aber M'_{i+1} und M''_{i+1} haben jetzt einen ungeraden Grad. Durch das Entfernen der Kante zwischen M'_{i+1} und M''_{i+1} haben beide Knoten wieder geraden Grad, jedoch zerfällt der Graph dabei in zwei Teilgraphen, in denen jeder Knoten geraden Grad hat. Durch das Einfügen einer Doppelkante zwischen den zwei Komponenten bleibt der Grad der neu verbundenen Knoten gerade und der Pfad bildet wieder einen Eulerkreis. Die Doppelkante schneidet nicht S . Daraus folgt, dass in jedem Schleifendurchlauf die Anzahl der Schnitte mit S um zwei reduziert wird, der Graph zusammenhängend bleibt und einen Eulerkreis bildet. Das Programm terminiert, sobald S nur noch maximal zweimal geschnitten wird.

Der Algorithmus fügt nur im dritten und im fünften Schritt jeweils zwei Kanten ein. In Schritt drei werden Liniensegmente eingefügt, die den Abstand M_i, M_{i+1}

haben und im fünften Schritt entspricht die Länge der eingefügten Liniensegmente M_{i+1}, M_{i+2} . Es werden immer zwei Kanten eingefügt, daraus ergibt sich eine maximale Gesamtlänge der eingefügten Kanten von $2l$. Es ist zu beachten, dass im Pfad π' die Kanten P_1, P_2, \dots, P_t aus π enthalten sind.

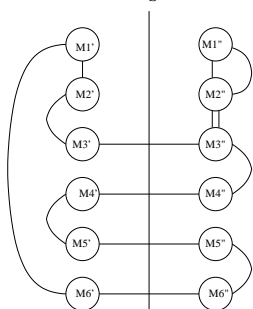


Abb. 3.3: Graph nach dem ersten Schleifendurchlauf

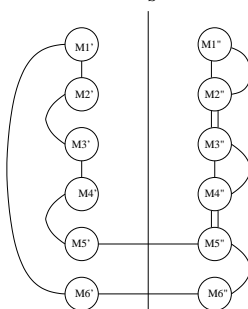


Abb. 3.4: Graph nach dem zweiten Schleifendurchlauf

Im Folgenden wird die zweite Aussage gezeigt. Es wird ein Hilfsgraph mit Pfad $\tilde{\pi}$ verwendet. Der neue Hilfsgraph ist dem ersten Hilfsgraph ähnlich, $\tilde{\pi}$ fehlen aber die Kanten $\{M'_i, M''_i\}$, also die Schnittkanten mit S . Es muss gezeigt werden, dass Liniensegmente der Mindestgesamtlänge $2l$ eingefügt werden müssen um einen Eulerkreis $\tilde{\pi}'$ zu konstruieren. Es ist zu beachten, dass laut Algorithmus nur ein Schnitt von S erlaubt ist, wenn der Ursprungsgraph eine ungerade Anzahl von Schnitten mit S hat. Wären die beiden durch S getrennten Seiten durch keine oder durch zwei Kanten verbunden, dann gäbe es auf beiden Seiten von S eine ungerade Anzahl von Knoten mit ungeradem Grad. Daraus folgt, diese Knoten können nicht miteinander verbunden werden, sodass jeder Knoten geraden Grad hat und S ein weiteres Mal geschnitten wird. Bei einer geraden Schnittanzahl mit S berechnet der Algorithmus zwei Schnitte.

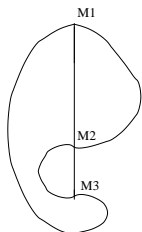


Abb. 3.5: Pfad $\tilde{\pi}$

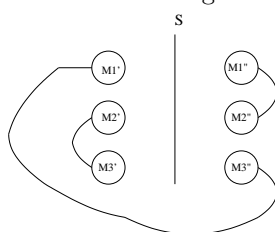


Abb. 3.6: Hilfsgraph zu $\tilde{\pi}$

Daher müssen in einen Graph Liniensegmente von mindestens $2l$ Gesamtlänge eingefügt werden, um einen Eulerkreis zu erstellen, der S höchstens zweimal schneidet.

q.e.d.

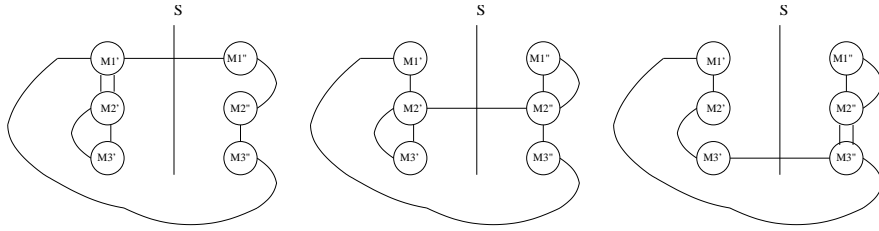


Abb. 3.7: Drei Möglichkeiten einen Eulerkreis $\tilde{\pi}$ zu berechnen.

Als nächstes soll gezeigt werden, wie die Länge einer Rundreise zu den Schnitten mit senkrechten und waagerechten Linien eines Gitters im Zusammenhang steht. In die Bounding-Box einer TSP-Instanz wird ein Gitter mit Körnung 1 gelegt. Sei π eine Rundreise und l eine beliebige feste Gitterlinie, dann beschreibt $t(\pi, l)$ die Anzahl der Schnitte von π mit l .

3.1.2 Lemma:

Wenn der Abstand zweier beliebiger Punkte mindestens 4 ist und π eine Rundreise der Länge T , dann gilt:

$$\sum_{l_{horizontal}} t(\pi, l) + \sum_{l_{vertikal}} t(\pi, l) \leq 2T$$

Der Summenteil der Ungleichung kann als ein Pfad in der l_1 -Norm aufgefasst werden, der maximal $\sqrt{2}$ mal länger ist, als der Pfad in der l_2 -Norm. In der l_1 -Norm wird die Zwischenknotendistanz bestimmt, indem der Pfad über die Kanten eines Gitters abgegangen wird. In der l_2 -Norm wird der direkte Weg betrachtet.

Sei e eine Kante aus π mit Länge s . Seien u und v die horizontale und vertikale Projektion von e . $u + v$ ist die l_1 -Länge und $\sqrt{u^2 + v^2}$ die l_2 -Länge.

Es gilt: $0 \leq (u - v)^2 = u^2 - 2uv + v^2$

$$\Leftrightarrow 2uv \leq u^2 + v^2$$

$$\Leftrightarrow u^2 + 2uv + v^2 = (u + v)^2 \leq 2(u^2 + v^2)$$

$$\Leftrightarrow |e|_{l_1} = u + v \leq \sqrt{2} \cdot \sqrt{u^2 + v^2} = \sqrt{2} \cdot |e|_{l_2}$$

Die linke Seite dieser Ungleichung wird für jede Kante e aus π um maximal $(u + 1) + (v + 1) = u + v + 2$ erhöht, daraus ergibt sich:

$$u + v + 2 \leq \sqrt{2} \cdot (u^2 + v^2) + 2 \leq \sqrt{2} \cdot s + 2 \leq 2s$$

q.e.d.

Nun wird das Struktur-Theorem bewiesen. Sei π eine optimale Rundreise mit Kosten opt_I zu einer beliebigen TSP-Instanz I , seien (a, b) beliebige Verschiebungen und sei $t(\pi, l)$ die Anzahl der Schnitte des Pfades π mit fester Linie l . Der Beweis erfolgt, indem der optimale Pfad π in einen (m, r) -leichten Salesman Pfad transformiert wird. Die Transformation wird in Bezug auf die verschobene Dissection vorgenommen. Diese kann anstelle des verschobenen Quadrees verwendet werden, da die Dreiecksungleichung gilt. Um die Transformation durchzuführen wird ein deterministisches Programm verwendet, das die Tourkosten leicht erhöht. Die Kostensteigerung ist nach oben beschränkt, durch:

$$E_{a,b}(\text{Belastung von } l) \leq \frac{t(\pi, l)}{4c}$$

Um den durch die Umwandlung gegebenen Kostenzuwachs leichter zu bestimmen, wird in die Bounding-Box von I ein Gitter mit Körnung 1 gelegt. Die Gitterlinien werden mit jeder Kostenerhöhung der Tour „belastet“. Für Zufallsvariablen X, Y gilt $E(X, Y) = E(X) + E(Y)$. Daraus folgt für die Ungleichung, bezogen auf die gesamte normierte Rundreise in Verbindung mit Lemma 3.1.2.:

$$\sum_l \frac{t(\pi, l)}{4c} = \frac{1}{4c} \sum_l t(\pi, l) = \frac{1}{4c} \left(\sum_{l_{horizontal}} t(\pi, l) + \sum_{l_{vertikal}} t(\pi, l) \right) \leq \frac{opt_I}{2c}$$

Da die Zufallsvariablen nur positive Werte annehmen können, kann die Markoffsche Ungleichung, $P(X \geq \lambda \cdot E(X)) \leq \frac{1}{\lambda}$ mit $\lambda = 2$ anwenden werden:

$$P(\text{Kostenerhöhung} \geq \frac{opt_I}{c}) \leq \frac{1}{2} \Leftrightarrow P(\text{Kostenerhöhung} \leq \frac{opt_I}{c}) \geq \frac{1}{2}$$

Daher ist der Kostenzuwachs mit Wahrscheinlichkeit von mindestens $\frac{1}{2}$ nicht größer als $\frac{opt_I}{c}$. Für die Kosten des (m, r) -leichten Salesman Pfades in der verschobenen Dissection folgt daraus, dass diese mit Wahrscheinlichkeit $\frac{1}{2}$ höchstens $(1 + \frac{1}{c})opt_I$ sind.

Für den Beweis muss nur noch angegeben werden, wie jeder optimale Pfad π in einen (m, r) -leichten Salesman Pfad transformiert werden kann und dabei die zusätzlich entstehenden Kosten auf die Kanten verteilt werden. Sei L die Größe der Bounding-Box o.B.d.A. eine Zweierpotenz. Dies kann durch Skalierung auf die nächst höhere Zweierpotenz erreicht werden. Die Bounding-Box wird in eine Dissection unterteilt. Die Unterteilung wird solange fortgesetzt, bis Quadrate mit Kantenlänge 1 entstehen. Das Partitionieren wird nicht gestoppt auch wenn schon alle Punkte separiert sind. Diese Dissection deckt sich mit dem Gitter, das zu Beginn beschrieben wurde. Da die zufällige (a, b) -Verschiebung ganzzahlig ist, ist auch die verschobene Dissection deckungsgleich mit diesem Gitter. Den Linien der Dissection kann, genau wie den Quadraten, ein Level zugeordnet werden, in diesem Zusammenhang wird von *Level- i -Linien* gesprochen. Dabei ist zu beachten, dass ein Liniensegment von Level i zwei Liniensegmente von Level $i + 1$ sind. Für alle $i \geq 0$ existieren 2^i vertikale und horizontale Level- i -Linien. Für die verschobene Dissection werden die vertikalen Koordinaten einer Linie bestimmt durch: $x_p = (a + p \cdot \frac{L}{2^i}) \pmod{L}$ für $p = \{0, 1, \dots, (2^i - 1)\}$. Für horizontale Linien erfolgt die Koordinatenbestimmung analog mit b .

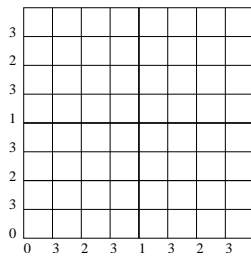


Abb. 3.8: Gitterlinien mit max-Level Werten

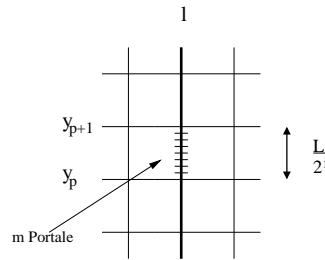


Abb. 3.9

Für jede Linie kann ihr *max-Level* bestimmt werden. Der max-Level einer Linie ist das Kleinstmögliche i , für das die Linie noch eine Level- i -Linie ist (siehe Abb. 3.8).

$ML_l : \{0, 1, \dots, L - 1\} \rightarrow \{0, 1, \dots, \log L\}$ ist die Definition einer Zufallsvariable für eine feste vertikale Gitterlinie l , die den max-Level von l bei horizontaler a -Verschiebung angibt.

max-Level	0	1	2	3	4	...	i	...	$\log L$
Anzahl möglicher a -Verschiebungen für feste Linie l	1	1	2	4	8	...	2^{i-1}	...	$\frac{L}{2}$

Es folgt, dass $Pr_a(ML_l = i) = \begin{cases} \frac{2^{i-1}}{L} & i \geq 1 \\ \frac{1}{L} & i = 0 \end{cases}$ die Wahrscheinlichkeit ist,

mit der eine feste vertikale Linie l nach einer a -Verschiebung max-Level i hat. Analog sind die Wahrscheinlichkeiten für das Erlangen von einem max-Level fester horizontaler Linien mit b -Verschiebung.

Im Folgenden wird die Umwandlung von π in einen (m, r) -leichten Salesman Pfad entlang der vertikalen Linien der Dissection beschrieben. Die Umwandlung entlang der horizontalen Linien erfolgt analog. Dass ein Pfad bezogen auf eine vertikale Linie (m, r) -leicht ist, bedeutet:

1. Eine vertikale Linie l mit max-Level i darf auf einem Segment y_p, y_{p+1} höchstens r -mal geschnitten werden.
2. Schnitte auf einem Segment y_p, y_{p+1} der Linie l dürfen nur in m vorgegebenen Portalen stattfinden.

Abbildung 3.9 zeigt ein Liniensegment y_p, y_{p+1} mit max-Level i und m Portalen. Schnitte des modifizierten Pfades mit diesem Segment dürfen nur höchstens r -mal an Portalen passieren.

Mit dem Patching-Lemma, angewendet auf die einzelnen Segmente einer Linie l mit max-Level i , können die Abschnitte (m, r) -leicht gemacht werden, wenn das Liniensegment öfter als r -mal vom optimalen Pfad π geschnitten wird. Dies kann jedoch die Gesamtkosten zu stark erhöhen. Das Patching-Lemma reduziert die Anzahl der Schnitte mit einem Segment auf höchstens zwei Schnitte, obwohl eine Höchstanzahl von $r > 2$ gefordert wird. Da es in einzelnen Linienabschnitten zu Häufungen von Schnittpunkten kommen kann, wird das Patching-Lemma *bottom-up* für alle Ebenen $j \leq i$ der Gitterlinie l angewendet. Der folgende Algorithmus realisiert dies deterministisch, wobei die zusätzlich entstehenden Kosten für vertikale Linien geringer sind:

MODIFY(l, i, b)

(l ist eine vertikale Gitterlinie, b ist die vertikale Verschiebung der Dissection und i ist der max-Level der Linie l)

FOR $j = \log L$ DOWNTO i DO:

FOR $p = 0$ TO $2^j - 1$ DO:

Sei l_p das Liniensegment auf l zwischen den y -Koordinaten

$(b + p \cdot \frac{L}{2^j}) \pmod{L}$ und $(b + (p + 1) \cdot \frac{L}{2^j}) \pmod{L}$;

IF Anzahl Schnitt der Rundreise mit $l_p > r$

THEN Wende Patching-Lemma auf l_p an;

END DO

END DO

Da Linien mit max-Level Null nicht von der Rundreise geschnitten werden, müssen für die Analyse von **Modify** Linien mit max-Level > 0 betrachtet werden. Da das Programm von der niedrigsten Ebene zu höheren Ebenen arbeitet, wird versucht, das Patching-Lemma in einem niedrigeren Level anzuwenden, also die Anzahl der Schnitte auf möglichst kleinen Liniensegmenten auf maximal zwei Durchkreuzungen zu reduzieren. Bei Segmenten, die durch die vertikale Verschiebung in zwei Teile zerfallen, müssen die beiden Teile separat betrachtet werden. Dadurch können durch die Anwendung des Patching-Lemmas maximal vier Schnitte entstehen. Die Anwendung des Patching-Lemmas kann die Anzahl der Schnitte mit horizontalen Liniensegmenten erhöhen, dies wird zunächst vernachlässigt.

Im Folgenden wird die Kostenerhöhung durch **Modify** untersucht. Sei l eine vertikale Linie mit max-Level i und $c_{l,j}(b)$ die Anzahl der Level- j -Liniensegmente, bei denen das Patching-Lemma angewendet wird; es gilt $j \geq i$. Für $j < i$ ist $c_{l,j}(b) = 0$, das unabhängig von i ist. Damit das Patching-Lemma angewendet wird, muss ein Liniensegment mindestens $(r + 1)$ -mal geschnitten werden. Daraus folgt, dass bei Anwendung des Patching-Lemmas mindestens $r - 3$ Schnitte entfernt werden, da nur maximal 4 Schnitte bestehen bleiben. Da die optimale Rundreise π die Gitterlinie l genau $t(\pi, l)$ -mal kreuzt, gilt für jede Gitterlinie l mit einer b -Verschiebung:

$$\sum_{j=i}^{\log L} c_{l,j}(b) \leq \frac{t(\pi, l)}{r - 3}$$

Die Kostenerhöhung durch **Modify** kann abgeschätzt werden mit:

$$\text{cost}_l(i, b) := \text{Kostenerhöhung durch } \text{MODIFY}(1, i, b) \leq \sum_{j=i}^{\log l} (c_{l,j}(b) \cdot 2 \cdot \frac{L}{2^j})$$

Die Zufallsvariable $K_l : 0, 1, \dots, L - 1 \rightarrow \mathbb{R}$ beschreibt den Kostenzuwachs einer festen, vertikalen, a -verschobenen Linie l , der dadurch entsteht, dass der optimale Pfad π so transformiert wird, dass ein beliebiges Liniensegment von l höchstens r -mal geschnitten wird (Bedingung 1). Die Belastung von l ist $\text{cost}(i, b)$ für jede Verschiebung bei der l max-Level i hat. Dadurch lässt sich die Belastung einer vertikalen Linie für diese Verschiebungen gruppieren, da die Belastung der vertikalen Linie unabhängig ist von dem horizontalen b -Shift.

E_a (Belastung von l durch Schritt 1)

$$\begin{aligned} &= \sum_{a=0}^{L-1} \frac{1}{L} \cdot K_l(a) = \sum_{i=0}^{\log L} P_a(ML_l = i) \cdot \text{cost}_l(i, b) \\ &= \sum_{i=1}^{\log L} \frac{2^{i-1}}{L} \cdot \text{cost}_l(i, b) \leq \sum_{i=1}^{\log L} 2^{i-1} \cdot \sum_{j=i}^{\log L} \frac{\text{cost}_l(j, b)}{2^{j-1}} \\ &= \sum_{j=1}^{\log L} \frac{\text{cost}_l(j, b)}{2^{j-1}} \cdot \sum_{i=1}^j 2^{i-1} = \sum_{j=1}^{\log L} \frac{\text{cost}_l(j, b)}{2^{j-1}} \cdot (2^j - 1) \\ &\leq 2 \cdot \sum_{j=1}^{\log L} c_{l,j}(b) = 2 \cdot \sum_{j=i}^{\log L} c_{l,j}(b) \leq \frac{2t(\pi, l)}{r-3} \end{aligned}$$

Die zweite Modifikation, das Führen des Pfades durch Portale, bewirkt auch eine Kostenerhöhung. Die Länge eines Liniensegments mit max-Level i ist $\frac{L}{2^i}$. Jedes Segment hat $m + 2$ Portale, deren Abstand $\frac{L}{2^i(m+2)}$ ist. Der maximale

Abstand zweier Schnittpunkte auf einem Segment ist $\frac{L}{2^{i(m+1)}}$. Schneidet π ein Segment, wird der Schnittpunkt auf das nächste Portal verschoben und nach Passieren des Portales wieder auf den Pfad zurückgeführt.

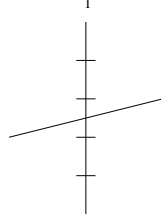


Abb. 3.10: Vor der Verschiebung durch ein Portal

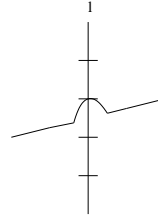


Abb. 3.11: Nach der Verschiebung durch ein Portal

Daraus folgt, dass die Kostenerhöhung pro Schnittpunkt nicht größer als $\frac{L}{2^{i(m+1)}}$ ist. Sei $K'_l : \{0, 1, \dots, L-1\} \rightarrow \mathbb{R}$ eine Zufallsvariable für eine feste vertikale Linie, die den Kostenzuwachs durch das Führen des Pfades durch Portale für eine a -Verschiebung angibt.

$$\begin{aligned} E_a(\text{Belastung von } l \text{ durch Schritt 2}) &= \sum_{a=0}^{L-1} \frac{1}{L} \cdot K'_l(a) \\ &\leq \sum_{i=0}^{\log L} \frac{2^{i-1}}{L} \cdot t(\pi, l) \cdot \frac{L}{2^{i(m+1)}} \\ &= \frac{t(\pi, l) \cdot \log L}{2^{(m+1)}} \leq \frac{t(\pi, l)}{kr} \end{aligned}$$

Die letzte Ungleichung gilt, falls $m > \frac{k}{2}r \log L$ für eine Konstante $k > 0$.

Nun kann die Gesamtkostenerhöhung nach oben abgeschätzt werden, indem die Ergebnisse kombiniert werden.

$$\begin{aligned} E_a(\text{Belastung von } l) &\leq \frac{2t(\pi, l)}{r-3} + \frac{t(\pi, l)}{kr} = \frac{(2kr+r-3)t(\pi, l)}{kr(r-3)} \leq \frac{(19k+8)(r-3)t(\pi, l)}{8kr(r-3)} \\ &= \frac{(19k+8)t(\pi, l)}{8kr} \leq \frac{t(\pi, l)}{4c} \end{aligned}$$

Die zweite Ungleichung setzt für alle $k > 0$ voraus, dass $r \geq 19$ und die letzte Ungleichung gilt nur, wenn $r \geq \frac{(19k+8)c}{2k}$. Da $c \geq 2$, gilt $r \geq 19 + \frac{8}{k} > 19$ für alle $k > 0$. Daher ist die Ungleichung für jedes k erfüllt. Eine untere Schranke für r ist daher $9\frac{1}{2} \cdot c$.

Damit der Beweis des Struktur-Theorems abgeschlossen werden kann, muss noch gezeigt werden, in wie weit das Anwenden des **Modify**-Algorithmus die Schnitte einer festen horizontalen Linie l' beeinflussen kann. Durch das Anwenden des Patching-Lemmas auf eine vertikale Linie l kann die Anzahl der Schnitte des modifizierten Pfades mit einer horizontalen Linie l' größer als $t(\pi, l')$ sein. O.B.d.A kann angenommen werden, dass sich die Anzahl der Schnitte nur um maximal zwei erhöht.

Ingesamt wird also jede Linie maximal $(r+2)$ -mal geschnitten und nicht r -mal, wie gefordert. Daher wird r durch $(r+2)$ ersetzt.

□

4 Effizienz des Algorithmus

Eine direkte Implementierung des Algorithmus ist nicht zu empfehlen, da schon für kleine c sehr viele Berechnungen vorgenommen und die Ergebnisse gespeichert werden müssen. Eine Kombination des Verfahrens mit einer *branch-and-cut* Implementierung ist empfehlenswerter. Eine solche Verbindung kann genutzt werden, um nahezu optimale Rundreisen in Graphen mit 5000 bis 50000 Knoten zu berechnen. Die Idee hierbei ist, eine solche TSP-Instanz mittels *divide and conquer* in kleinere Teilinstanzen zu zerlegen. Aus einer TSP-Instanz mit k Knoten werden so 2^i Teilgraphen, wobei jeder dieser Teilgraphen nur $\frac{k}{2^i}$ Knoten besitzt. Diese Teile können dann mit dem gegebenen Algorithmus einzeln gelöst werden.

5 Bibliographie

- *Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems*, Sanjeev Arora, Princeton University
- *Approximationsalgorithmen Sommersemester 2004*, Marco Lübbecke, TU-Berlin, 2004
- *Untersuchung zu einem polynomiellen Approximationsschema für das Problem des Handlungsreisenden im Euklidischen Raum*, Axel Thümmler, Prof. Dr. H. U. Simon, Universität Dortmund, 1998
- *Einführung in die Wahrscheinlichkeitstheorie und Statistik*, Ulrich Krenzel, Vieweg Verlag, 2003