Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik I

Tom Noll

Approximation von Traveling Salesman Touren

31. Mai 2007

Seminararbeit im WS 2006/2007

Zusammenfassung

In dieser Ausarbeitung wird ein PTAS Algorithmus vorgestellt, der das Travelling Salesman Problem Approximativ löst. Dabei beträgt die Länge der Lösung des PTAS $(1 + \frac{1}{c})OPT$ der Länge des optimalen Pfades. Die Laufzeit des Algorithmus beträgt dabei $O(n\ (\log n)^{O(c)})$.

Inhaltsverzeichnis

1 Motivation

Das *Traveling Salesman Problem (TSP)*, oder im deutschen *Problem des Handlungsreisenden*, ist ein kombinatorisches Optimierungsproblem. Es geht darum eine Route über eine feste Anzahl von Punkten zu finden, die wieder zum Ausgangspunkt zurückführt und die geringst möglichen Kosten hat. Dieses Problem tritt im Alltag an vielen Stellen auf. Angefangen bei der Planung einer Reise zu verschiedenen Orten, wie sie zum Beispiel ein Handlungsreisender unternehmen muß. Diesem Beispiel verdankt das Problem seinen heute üblichen Namen, der ihm von Hassler Whitney in den 30er Jahren gegeben wurde. Bekannt ist dieses Optimierungsproblem jedoch schon viel länger. Schon im 19. Jahrhundert beschäftigten sich Wissentschaftler mit der Lösung eng verwandter Probleme, die als Vorreiter des heutigen TSP gelten. Die richtige Arbeit an einer Lösung des TSP begann jedoch erst, wie auch seine Benennung, in den 30er Jahren des 20. Jahrhunderts.
Das *Traveling Salesman Problem* ist NP-vollständig. Das bedeutet, dass, wenn $P \neq NP$ gilt, jede deterministische Lösung, die einen optimalen Weg findet, exponentiell viel Zeit in der Zahl der zu besuchenen Punkte braucht, um diesen Weg zu finden. Es gibt daher grundsätzlich zwei Arten der Lösung des TSP. Die eine ist eine exakte Lösung, die jedoch exponentiell lange Laufzeit in der Anzahl der Knoten aufweisen kann, oder eine heuristische Lösung, die jedoch eine beliebig schlechte Annäherung bewirken kann. Die Heuristiken zur Lösung des TSP sind mit den Jahren der Forschung immer besser geworden und erlauben eine sehr gute Approximation in polynomieller Zeit. In dieser Ausarbeitung wird ein Algorithmus von Sanjeev Arora [1] beschrieben, der ein sogenanntes *Polynomimial-Time Approximation Scheme* (PTAS) verwendet, um das *Traveling Salesman Problem* zu lösen. Ein PTAS ist ein Verfahren, welches eine $(1 + \frac{1}{c})$-Approximation eines Problems in polynomieller Zeit liefert. Dabei ist $c > 1$ eine selbst gewählte Konstante. Diese bestimmt die Güte der Approximation. Der hier vorgestellte randomisierte Algorithmus hat eine Laufzeit von $O(n\ (\log\ n)^{O(c)})$, bei n Knoten im $\mathbb{R}^2$. Es existiert auch eine deterministische Version des Algorithmus. Bei dieser erhöht sich die Laufzeit jedoch um den Faktor $O(n^2)$ im $\mathbb{R}^2$.

2 Der PTAS Algorithmus im $\mathbb{R}^2$

Der hier vorgestellte Algorithmus verwendet eine rekursive geometrische Partitionierung der Ausgangssituation um das Traveling Salesman Problem auf ein kleineres Problem zu reduzieren und es dann mittels dynamischer Programmierung zu lösen. Dabei wird, nach einigen Vorarbeiten, welche später genauer betrachtet und analysiert werden, um die beteiligten Knoten ein achsenparalleles umschließendes Quadrat gelegt, welches rekursiv in vier gleichgroße Quadrate zerlegt wird. Jedes dieser Quadrate wird wieder wie die Bounding Box zerlegt. Dies wird durchgeführt, bis in jedem einzelnen Quadrat nur noch maximal ein Knoten vorhanden ist, oder die Größe der Quadrate ≤ 1 wird. Diese Zerlegung wird dann noch verschoben und in einen Quadtree umgewandelt. Mittels dynamische Programmierung wird nun für jedes Quadrat aus jeder Schicht ein optimaler Pfad berechnet.
Im folgenden betrachten wir die einzelnen Schritte des Algorithmus genauer:

1. **Anpassungen**

   Als erstes wird die Anfangssituation so modifiziert, dass spätere Laufzeit- und Komplexitätsberechnungen einfacher werden. Es soll erreicht werden, dass alle Knoten eine ganzzahlige x- und y-Koordinate haben, und das die Distanz zwischen 2 Knoten (solange ungleich 0) größer als 8 und die max. Distanz $O(n)$ ist. Es sei nun L_0 die Größe einer Seite der Bounding Box, OPT die Länge des optimales Pfades, n die Anzahl der Knoten und $c > 1$ eine Konstante. Dabei ist $OPT \geq L_0$, wie leicht gesehen werden kann. Die Anpassung wird nun durchgeführt, indem ein Gitternetz mit Gitterabstand $L_0/8nc$ über die Ausgangssituation gelegt wird, und jeder Knoten auf seinen nächsten Gitterpunkt verschoben wird. Dabei kann es dazu kommen, dass 2 Knoten auf denselben Punkt verschoben werden. Dies ist jedoch kein Problem, da der optimale Pfad auf jedenfall eine Verbindung dieser sehr nahe zusammenliegenden Punkte enthält und die zusätzliche Wegstrecke vernachlässigbar klein ist. Diese Verschiebung ändert natürlich den Abstand der Knoten und kann daher den optimalen Pfad verlängern. Dies muß berücksichtigt werden. Die Kosten des optimalen Pfades erhöhen sich um max. $2n * L_0/8nc$. Dies resultiert aus der Tatsache, dass die längste Strecke, die sich ein Knoten bewegen kann, vom Mittelpunkt eines Gitternetzquadrates zu dessen Ecken ist. Da die Seitenlänge dieser Quadrate $L_0/8nc$ beträgt, ist die zurückgelegte Strecke $\sqrt{(\frac{1}{2}L_0/8nc)^2 + (\frac{1}{2}L_0/8nc)^2}$. Umgeformt ergibt dies $\frac{1}{\sqrt{2}}L_0/8nc$. Da $\frac{1}{\sqrt{2}} < 1$ ist, kann jeder Knoten maximal $L_0/8nc$ zu jedem der beiden Liniensegmente des Pfades, die an ihm enden, hinzufügen. Bei n Knoten ergibt dies obige Abschätzung.

Da $L_0 \leq OPT$ ist, gilt:

$$2n \cdot \frac{L_0}{8nc} = \frac{L_0}{4c} \leq \frac{OPT}{4c} \quad .$$

Also werden die Kosten des optimalen Pfades durch die Anpassung um maximal $OPT/4c$ erhöht. Dies bedeutet, dass von der angestrebten $(1 + \frac{1}{c})OPT$ Approximation schon $(1 + \frac{1}{4c})OPT$ "verbraucht"wurde. Daher muss für die angepasste TSP Instanz eine $(1 + \frac{3}{4c})OPT$ Approximation erzeugt werden. Da jedoch $c > 1$ eine Konstante ist, stellt dies kein Problem da.
Nach dieser Verschiebung werden alle Koordinaten der Knoten durch $L_0/64nc$ dividiert. Dadurch werden alle Koordinaten ganzzahlig und der Abstand der Koordinaten ist mindestens 8, denn

$$\frac{L_0}{8nc} \cdot \frac{64nc}{L_0} = 8 \quad .$$

2. **Konstruktion des Quadtree**

Nachdem die Anpassungen aus Punkt 1 vorgenommen wurden, wird der Quadtree aufgebaut. Dabei wird o.B.d.A angenommen, dass die Seiten der Bounding Box (L_0) eine Potenz von 2 sind. Dabei liegt L_0 in $O(n)$. Eine Zerlegung der Bounding Box ist hier, wie am Anfang des Kapitels bereits beschrieben, eine rekursive Teilung der Bounding Box in 4 gleichgroße Quadrate, welche wiederum geteilt werden, bis jedes Quadrat nur noch einen Knoten enthält, oder ≤ 1 ist. Ein Quadtree wird genauso aufgebaut, mit dem Unterschied, dass hier die rekursive Zerteilung eines Quadrates gestoppt wird, sobald nur noch ein Knoten innerhalb des Quadrates liegt, siehe Abbildung 1.
Der hier vorgestellte Algorithmus benötigt jedoch eine spezielle Art des

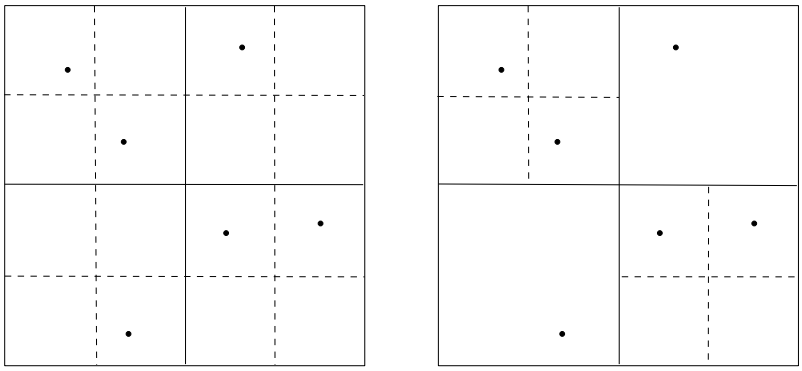

Abbildung 1: Eine Zerlegung (links) und der dazugehörige Quadtree (rechts).

Quadtree. Dabei wird erst eine Zerlegung angefertigt und diese dann verschoben. Hierfür wird zufällig ein Tupel (a, b) $a, b \in [0, L_0) \wedge a, b \in \mathbb{N}$ gewählt und die inneren Ränder der ersten Aufteilung, also die „Mittellinien"der Zerlegung um (a, b) verschoben. Die vertikale Mittellinie wird von der x-Koordinate $L_0/2$ nach $a + L_0/2 \bmod L_0$ und die horizontale von der y-Koordinate $L_0/2$ nach $b + L_0/2 \bmod L_0$ verschoben. Alle weiteren Linien (außer den äußeren Linien der Bounding Box) werden entsprechend mit verschoben, wobei Linien, die über den Rand der Bounding Box wandern, auf der anderen Seite wieder auftauchen. Diese neue Zerlegung wird nun zu einem Quadtree umgewandelt. Dabei sind einige Quadrate umgebrochen, siehe Abbildung 2 und Abbildung 3. Da L_0 in $O(n)$ liegt, ist die Tiefe des Quadtree $O(\log n)$ und die Anzahl der enthaltenen Quadrate ist $T = O(\text{Zahl der Blätter mit Knoten} \cdot \text{Tiefe}) = O(n \log n)$.

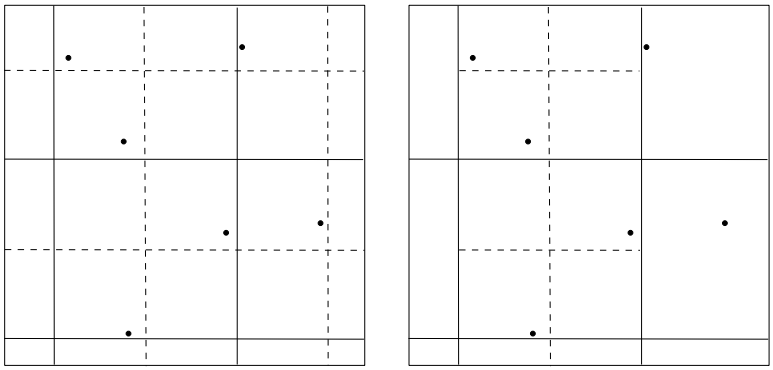

Abbildung 2: Eine um (a, b) verschobene Zerlegung (links) und der dazugehörige Quadtree (rechts).

3. **Dynamische Programmierung**
   Für diesen Abschnitt wird das *Struktur Theorem* und die Definition von (m, r)-light benötigt.

   Definition 1 Seien m, r zwei $\in \mathbb{N}$. Eine *m-reguläre Menge von Portalen* für eine verschobene Zerlegung ist eine Menge von Punkten auf den Kanten ihrer inneren Quadrate. Jedes Quadrat hat ein Portal an jeder seiner 4 Ecken und m weitere gleichweit voneinander entfernte auf jeder Kante.
   Ein *Salesman path* ist ein Weg im $\mathbb{R}^2$, der jeden Knoten und eine Untermenge an Portalen besucht. Dabei kann ein Portal durchaus mehrmals besucht werden.

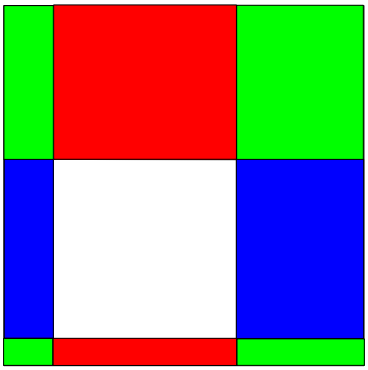

Abbildung 3: Hier sind Quadrate der Tiefe 1 zu sehen, die durch die Verschiebung umgebrochen wurden. Je eine Farbe zeigt ein Quadrat. Diese können aus bis zu 4 Teilen bestehen.

Ein *Salesman path* ist $(m, r) - light$ bezüglich der verschobenen Zerlegung, wenn er jede Ecke jedes Quadrates der Zerlegung maximal r-mal schneidet. Jeder Schnitt passiert dabei an einem Portal.

Theorem 2 *(Struktur Theorem)*
Sei $c > 0$ eine Konstante. Sei die minimale Distanz (ungleich 0) zwischen den Knoten einer TSP Instanz 8 und L die Größe der Kanten der Bounding Box. Seien $0 \leq a, b \leq L$ zufällig ermittelte Verschiebungen. Dann gibt es mit einer Wahrscheinlichkeit von $1/2$ einen Salesman-path *mit Kosten $(1+\frac{1}{c})OPT$ der $(m, r) - light$ bezüglich der verschobenen Zerlegung ist. Dabei ist $m = O(c \log L)$ und $r = O(c)$.*

Der Beweis des Theorems folgt später.

Dabei wird ausdrücklich erlaubt, daß die Kanten des *Salesman path* „geknickt“ sind. Dies entsteht daher, dass ein *Salesman path* nur durch Portale die Kanten der Quadrate kreuzen darf. Nun liegen jedoch die Portale, die 2 Knoten optimal verbinden nicht unbedingt auf einer Linie. Also besteht ein *Salesman path* aus Liniensegmenten, die die einzelnen Portale miteinander verbinden. Die Ecken, welche dabei an den Portalen entstehen, stellen die Knicke dar. Diese lassen sich nach der Konstruktion des *Salesman path* durch die Dreicksungleichung ohne Verlängerung des Pfades „begradigen”.
Nach dem Struktur Theorem gibt es also mit Wahrscheinlichkeit $1/2$ zu einer Verschiebung (a, b) einen *Salesman path*, der maximal $(1 + \frac{1}{c})OPT$ lang ist. Dieser soll im folgenden durch dynamische Programmierung erzeugt werden. Dafür wird der Umstand verwendet, dass

nach dem Struktur Theorem der *Salesman path* $(m, r) - light$ ist. Gilt dies, so kann der optimale *Salesman path* ein beliebiges Quadrat des Quadtree maximal durch $4r$ Portale betreten bzw. verlassen. Jede Kante des Quadrates kann dabei max. r-mal geschnitten werden. Dabei zerfällt der *Salesman path* in p Teilstücke die jeweils 2 Portale des Quadrates miteinander verbinden. Seien nun $a_1, a_2, ..., a_{2p}$ die Portale auf dem Quadrat nummeriert nach dem auftreten auf dem *Salesman path*, dann gilt für jedes Quadrat, dass: (1) das i-te Teilstück $(i = 1, ..., p)$ die Portale a_{2i-1} und a_{2i} verbindet, (2) die Teilstücke zusammen alle Knoten innerhalb des Quadrates besuchen, und (3) die Teilstücke zusammengenommen $(m, r) - light$ sind. Da der *Salesman path* optimal ist, muß er die Menge an Pfaden enthalten, die (1)-(3) erfüllen und die geringesten Kosten haben. Dies führt zu einem sogenannten (m,r)-Multipfadproblem. Dieses erhält als Eingabe folgendes:

i ein nichtleeres Quadrat aus dem verschobenen Quadtree

ii eine Menge aus $\leq r$ Portalen pro Kante des Quadrates, so dass die Summe der Portale $2p \leq 4r$ nicht übersteigt

iii Eine Paarung $\{a_1, a_2\}, \{a_3, a_4\}, \ldots, \{a_{2p-1}, a_{2p}\}$ zwischen den $2p$ Portalen aus ii.

Das Ziel des (m, r)-Multipfadproblems ist es, die $2p$ Pfade so anzuordnen, dass sie die Paarung aus iii erfüllen, alle Knoten innerhalb des Quadrates besuchen und minimale Kosten haben. Im Prinzip ist es ein TSP mit mehreren Salesman, die zusammengenommen alle Knoten innerhalb des Quadrates besuchen müssen. Dabei startet jeder auf einem Portal und muß am Ende seiner Tour an seinem Partnerportal, welches in iii festgelegt wurde, angekommen sein. Ist p für eine Instanz gleich 0, so wird hier der optimale Pfad gesucht, der alle Knoten verbindet.
Mit Hilfe von dynamischer Programmierung wird nun eine Lookup Table aufgebaut, in der die Kosten *aller* Instanzen des (m, r)-Multipfadproblems aufgelistet werden, die in dem Quadtree auftreten können. Dies sind insgesamt $O(T \cdot (m + 4)^{4r} \cdot (4r)!)$, denn für jedes nichtleere Quadrat T gibt es maximal $(m + 4)^{4r}$ Möglichkeiten die Portale auszuwählen und für jede dieser Wahlen gibt es $(4r)!$ Möglichkeiten die Paarungen zwischen diesen zu wählen.
Die Tabelle wird von unten nach oben aufgebaut. Das heißt, dass mit den kleinsten Quadraten begonnen wird. Diese enthalten nur einen einzigen Knoten und somit kann eine einzelne Instanz des (m,r) Multipfadproblems in $O(r)$ Zeit gelöst werden. Und zwar indem jeder der p Pfade einmal so gelegt wird, dass er den einzelnen Knoten enthält. Die anderen Pfade gehen dabei jeweils von ihrem Ausgangspunkt direkt zu ihrem Zielpunkt. Die Lösung mit den minimalsten Kosten wird

dann in der lookup Table eingetragen. Dies wird nun für alle Instanzen des (m, r)-Multipfadproblems mit diesem Quadrat als Eingabe wiederholt. Sind so alle nichtleeren Quadrate der niedrigsten Stufe ausgewertet, wird die nächsthöhere Stufe von Quadraten betrachtet. Hier wird auch wieder jede Instanz des (m, r)-Mulipfadproblems betrachtet. Dabei besteht das betrachtete Quadrat aus 4 Quadraten der nächstkleineren Stufe. Für eine konkrete Instanz wird die optimale Lage der Pfade berechnet, indem für die p Pfade alle Möglichkeiten aufgezählt werden, wie sie die inneren Kanten der 4 Quadrate nächstkleinerer Stufe schneiden. Jede dieser Möglichkeiten ergibt vier (m, r)-Multipfadprobleme, eines für jedes der kleineren Quadrate. Da diese aber schon alle gelöst worden sind, können die optimalen Kosten für diese Probleme aus der lookup Table ausgelesen werden. Dies wird nun für alle Instanzen aller Quadrate des Quadtree durchgeführt. Jede Instanz eines (m, r)-Multipfadproblems eines höheren Quadrates erzeugt dabei maximal $(m + 4)^{4r}$ Möglichkeiten, wie die inneren Kanten der nächstkleineren Quadrate geschnitten werden können und maximal $(4r)^{4r}4!$ Möglichkeiten in welcher Reihenfolge diese Schnitte durchlaufen werden können. Insgesammt ergibt dies eine Laufzeit von:

$$O(T \cdot (m + 4)^{8r} \cdot (4r)^{4r}(4!)^2)$$

was $O(n(\log n)^{O(c)})$ entspricht. Diese Abschätzung folgt aus den folgenden Abschätzungen:

$$T \in O(n \log n),\ m \in O(c \log L),\ r \in O(c),\ r! \in O(c)^O(c).$$

Nun können die Kosten des optimalen *Salesman path* aus der Lookup Table ausgelesen werden. Die Kosten entsprechen dem Eintrag für das größte Quadrat und $p = 0$. Da neben den Kosten auch der tatsächliche Pfad ausgegeben werden soll, muß noch einmal durch die lookup Table gegangen werden und die Pfade jeder Entscheidung die zu den Kosten des optimalen *Salesman path* geführt haben herausgenommen und zu dem eigentlichen Pfad zusammengefügt werden. Dieser kann Selbstschnitte enthalten, doch diese können recht einfach beseitigt werden.

3 Korrektheit des Algorithmus

Durch die dynamische Programmierung wurde ein *Salesman Path* aufgebaut. Dabei wurde bei jedem Schritt darauf geachtet, dass der *Salesman Path* $(m, r) - light$ ist und die geringstmöglichen Kosten hat. Nach dem Struktur Theorem gibt es mit Wahrscheinlichkeit $\frac{1}{2}$ einen *Salesman Path*, der $(m, r) - light$ und eine $(1 + \frac{1}{c})OPT$ Approximation der optimalen Traveling Salesman Tour ist. Da der durch den Algorithmus erzeugte *Salesman*

Path der kürzeste aller möglichen ist, muß dies der im Struktur Theorem genannte sein. Für die Korrektheit des Algorithmus fehlt also nur noch der Beweis des Struktur Theorems.
Für den Beweis werden zwei weitere Lemmata benötigt.

Lemma 3 *(Patching Lemma)*
Es gibt eine Konstante $g > 0$, so dass folgende Aussage wahr ist. Sei S ein Liniensegment der Länge s und π ein geschlossener Weg der S mindestens dreimal schneidet. Dann lassen sich π Liniensegmente auf S mit Gesamtlänge $\leq g \cdot s$ hinzufügen, so dass der resultierende Weg S maximal zweimal schneidet.

Beweis. Angenommen π schneidet S $t \geq 3$-mal und die Schnittpunkte seien $M_1, ..., M_t$. Wird π an diesen Schnittpunkten getrennt, so zerfällt es in t Teile $P_1, ..., P_t$. Wir verdoppeln die Schnittpunkte zu zwei Mengen M' und M'', so dass auf jeder Seite von S ein Schnittpunkt aus M liegt. Dabei zerfällt M_1 in M_1' und M_1'', M_2 in M_2' und M_2'', etc. , siehe hierzu auch Abbildung 4.

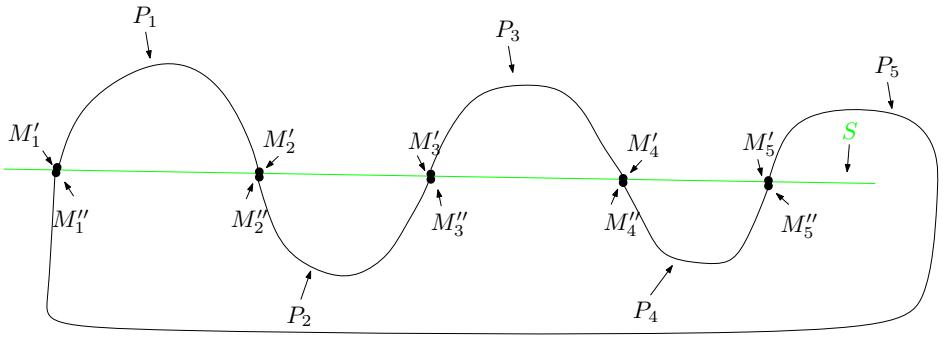

Abbildung 4: Ein geschlossener Pfad der von einem Liniensegment S in $t = 5$ Teilstücke zerlegt wurde. Dabei wurden die Schnittpunkte in je 2 Punkte geteilt. Je einer auf jeder Seite von S.

Sei $2k$ die größte gerade Zahl, die **kleiner** als t ist. (Hierbei ist es wichtig, dass $2k$ nie gleich t sein kann!) Es wird nun eine Gruppe J von Liniensegmenten erzeugt, die einerseits eine minimale Salesman Tour mit den Punkten $M_1, ..., M_t$ und andererseits ein minimales perfektes Matching mit den Punkten $M_1, ..., M_{2k}$ bilden. Diese Liniensegmante liegen alle auf S und sind zusammen maximal $3s$ lang. Dabei ist die minimale Salesman Tour maximal $2s$ und das perfekte Matching maximal s lang. Das perfekte Matching verbindet auch immer M_1 mit M_2, M_3 mit M_4, ..., da dies auf einer Linie und einer graden Anzahl von Punkten das minimale perfekte Matching ist. Nachdem J erzeugt wurde, werden alle Liniensegmente verdoppelt und auf je eine Seite von S geschoben. Siehe hierzu Abbildung 5 .

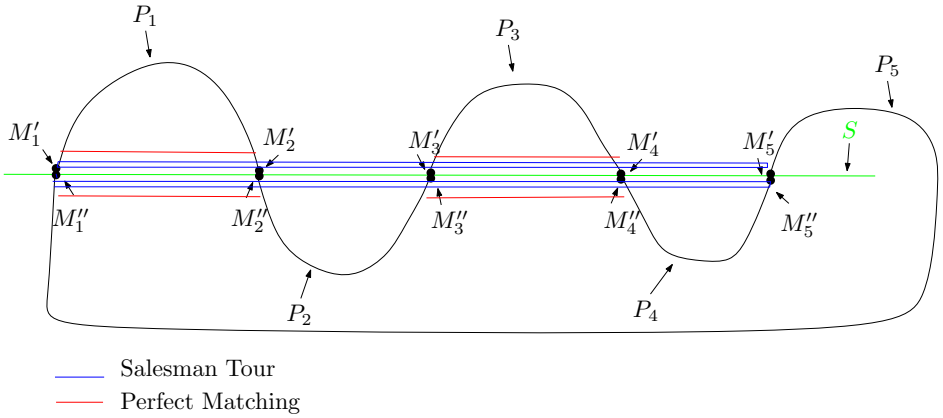

Abbildung 5: Auf S liegen die Liniensegmente des optimalen *Salesman Path* und des *Perfect Matching*. Diese sind leicht abgehoben gezeichnet, damit sie erkennbar sind. Eigentlich liegen sie direkt auf S.

Wenn $t = 2k + 1$ (t also ungerade) ist, dann wird zwischen M'_{2k+1} und M''_{2k+1} eine Verbindung hergestellt, und wenn $t = 2k + 2$ ist, dann wird zwischen M'_{2k+1} und M''_{2k+1} und zwischen M'_{2k+2} und M''_{2k+2} eine Verbindung eingefügt. Diese Verbindungen haben die Länge 0. Zusammengenommen ergeben die Liniensegmente von π, die verdoppelten von J und die eingefügten Verbindungen zwischen den Schnittpunkten einen Graphen, in der jeder Knoten genau den Grad 4 hat, wie auf Abbildung 6 zu sehen ist.

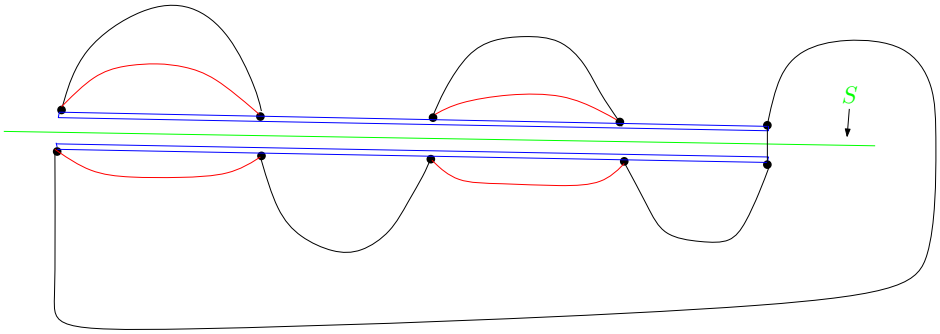

Abbildung 6: Der 4 reguläre Graph aus Liniensegmenten aus j und π. Die Liniensegmente liegen eigentlich direkt auf S, sind hier aber zur Verdeutlichung abgehoben gezeichnet.

Auf diesem Graphen kann eine Eulertour gefunden werden. Diese ist die minimale Tour die $P_1, ..., P_t$ enthält und sie schneidet S maximal 2 mal. Genauer schneidet sie S einmal, wenn t ungerade und zweimal, wenn t gerade

ist □

Lemma 4 *Gegeben sei ein Quadrat, welches durch Giternetzlinien mit Abstand 1 aufgeteilt ist. Sei nun l eine solche Linie und π eine Salesman Tour, dann bezeichne $t(\pi, l)$ die Anzahl der Schnitte von π und l. Wenn die minimale Distanz zwischen 2 Knoten einer TSP Instanz, welche als Bounding Box das oben gennante Quadrat besitzt, maximal 4 beträgt und T die Länge von π ist, dann gilt:*

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

Beweis. Die linke Seite der Gleichung zählt die Schnitte von π mit den horizontalen und vertikalen Linien des Gitternetzes. Da diese den Abstand von 1 zueinander haben, misst die linke Seite praktisch die Länge von π in l_1 Norm und diese beträgt maximal $\sqrt{2}T$. Da jedoch nicht alle Punkte von π auf den Liniensegmenten liegen müssen, ist die Länge von π in l_1 Norm nicht exakt gleich der linken Seite der Ungleichung. Die Betrachtungsweise liefert jedoch einen guten Ansatz zur Abschätzung.
Sei nun s eine Kante von π und u und v die horizontalen und vertikalen Projektionen dieser Kanten, so dass $u^2 + v^2 = s^2$ gilt. u bzw. v können höchstens $\lfloor u \rfloor + 1$ bzw. $\lfloor v \rfloor + 1$ mal π schneiden. Daher erhöht s die linke Seite der Ungleichung um maximal:

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

Für $s \geq 4$ gilt außerdem: $\sqrt{2s^2} + 2 \leq 2s$ □

Beweis.(des Struktur Theorems)
Sei $s = 12gc$, wobei g die Konstante des Patching Lemmas ist. Sei $r = s + 4$, $m \geq 2s \log L$, π eine optimale Salesman Tour und (a, b) eine zufällig ermittelte Verschiebung. Im folgenden wird π so modifiziert, dass daraus ein *Salesman Path* wird, der $(m, r) - light$ ist. Übersteigen die zusätzlichen Kosten des Pfades dabei OPT/c nicht, so ist das Theorem bewiesen. Für die Zählung der zusätzlichen Kosten wird ein Gitternetz mit Gitterabstand 1 in die Bounding Box gelegt. Sämtliche zusätzliche Kosten werden einer der Gitternetzlinien l „angerechnet". Dabei gilt:

$$E_{a,b}[\text{Kosten für } l \text{ bei Verschiebung } (a,b)] \leq \frac{3g * t(\pi, l)}{s} \qquad (1)$$

Da die zusätzlichen Kosten linear auf die einzelnen Gitternetzlinien addiert werden, folgt für alle Linien eine Kostenzunahme von

$$\sum_{l:vertikal} \frac{3g \cdot t(\pi, l)}{s} + \sum_{l:horizontal} \frac{3g \cdot t(\pi, l)}{s}$$

$$
\begin{aligned}
&= \frac{3g}{s}\Big(\sum_{l:vertikal} t(\pi,l) + \sum_{l:horizontal} (\pi,l)\Big) \\
&\leq \frac{3g}{s} \cdot 2T \quad \text{nach Lemma 4} \\
&\leq \frac{6g}{s} OPT
\end{aligned}
$$

Mit $s \geq 12gc$ ergibt das einen maximalen Kostenzuwachs von $OPT/2c$. Nach der Markov Ungleichung [1] gilt dann auch, dass mit einer Wahrscheinlichkeit von $\frac{1}{2}$ die Kostenzunahme maximal OPT/c beträgt.
Das heißt, dass mit einer Wahrscheinlichkeit von $\frac{1}{2}$ ein *Salesman Path* existiert, der $(m,r) - light$ ist und maximal die Länge $(1 + 1/c)OPT$ hat, solange Gleichung 1 gilt.
Es muß also noch gezeigt werden, dass Gleichung 1 gilt. Dafür wird die Modifikation der optimalen Salesman Tour genauer betrachtet. Als Ausgangssituation sei gegeben, dass um die Knoten der optimalen Salesman Tour eine Bounding Box gelegt wurde, die o.b.d.A eine Seitenlänge hat, die eine Potenz von 2 ist. Außerdem sei eine Verschiebung um (a,b) und eine darauf basiernde Zerlegung vorgenommen worden. Da $(a,b) \in \mathbb{N}$ sind, liegen die Linien der Zerlegung alle auf den Linien eines Gitternetzes mit Abstand 1. Die einzelnen Quadrate der Zerlegung seien mit einer Tiefe versehen, die ihrer Entstehung beim Aufstellen der Zerlegung entspricht. Das heißt, dass ein Quadrat, welches vor einem anderem in der Zerlegung entsteht eine kleinere Tiefe besitzt, als eines, was später ensteht. Die Bounding Box hat zum Beispiel Tiefe 0, die 4 Quadrate in die sie aufgeteilt wird haben Tiefe 1, etc. Jede Gitternetzlinie besitze nun die kleinste Tiefe von allen Quadraten, die eine Kante mit ihr teilen. Liegt also auf einer Gitternetzlinie zum Beispiel eine Kante eines Quadrates der Tiefe 3, aber auch eine Kante eines Quadrates der Tiefe 1, so besitzt die Gitternetzlinie die Tiefe 1. Da bei der Zerlegung ein Quadrat in weitere Quadrate zerlegt wird, wird auch jede Kante in 2 Kanten der nächstgrößeren Tiefe zerlegt. Dies bedeutet, dass eine Kante der Tiefe i auch Kanten der Tiefe $i+1, ...,$ *maximale Tiefe* enthält. Die Zerlegung besitzt pro Tiefe $i \geq 1$ 2^i horizontale und vertikale Linien. Da a zufällig gewählt wurde, beträgt die Wahrscheinlichkeit, dass eine vertikale Gitternetzlinie l Tiefe i besitzt:

$$P_a[l \text{ hat Tiefe } i] = \frac{2^i}{L} \tag{2}$$

Eine ähnliche Aussage gilt auch für b.
Um die optimale Salesman Tour in einen Salesman Path der $(m,r) - light$ ist zu überführen, muss als erstes sichergestellt werden, dass jedes der 2^i Segmente jeder Gitternetzlinie durch π maximal r-mal geschnitten wird.

[1] Mark. Ungl.: $P[h(x) \geq a] \leq \frac{E[h(X)]}{a}$ Hier: $P[\text{Kost.ZN.} \geq \frac{OPT}{c}] \leq \frac{OPT}{2c} \cdot \frac{c}{OPT} \leq \frac{1}{2}$

(Für jede Gitternetzlinie wird i wie oben beschrieben bestimmt.) Um die Schnitte eines Segmentes zu reduzieren wird das Patching Lemma verwendet. Die einfachste Methode wäre es, für jede Gitternetzlinie die 2^i Segmente zu testen und bei mehr als r Schnitten das Patching Lemma anzuwenden. Dies würde jedoch die Länge des Salesman Path zu sehr erhöhen, da die Kosten, die durch das Patching Lemma enstehen, von der Größe der Liniensegmente abhängt. Daher wird das Patching Lemma für ein Liniensegment mit minimaler Tiefe i, im folgenden L_i genannt, nicht direkt angewendet, sondern es wird ersteinmal auf alle Liniensegmente j der maximalen Tiefe, die innerhalb von L_i liegen und mehr als s mal geschnitten werden, angewendet. Es wird hier s als Grenze für die Anzahl der Schnitte verwendet, da durch das Anwenden des Patching Lemmas auf ein Liniensegment bei einem othogonal liegenden weitere Schnitte auftreten können. Darauf wird später näher eingegangen. Sind alle Liniensegmente der Tiefe j innerhalb von L_i $(m, s) - light$, so wird für die Liniensegmente der Tiefe $j - 1$ getestet, ob diese $(m, s) - light$ sind. Sind sie es nicht, wird nun auch hier das Patching Lemma angewendet. Dies wird solange fortgesetzt, bis $j = i$ ist. Bei jedem Anwenden des Patching Lemmas wird die Zahl der Schnitte durch dieses Liniensegment auf maximal 4 reduziert. Es sind 4, und nicht 2, wie eigentlich im Patching Lemma angegeben, da die einzelnen Kanten der Quadrate durch die Verschiebung um (a, b) umgebrochen sein können, und so für jeden der beiden Teile das Patching Lemma angewendet werden kann.
Im folgenden wird nun eine Abschätzung der Kosten für die iterative Anwendung des Patching Lemmas gegeben. Da die Abschätzung für die vertikalen und horizontalen Linien analog ist, wird hier nur die Abschätzung für die vertikalen Linien betrachtet. Sei dafür $c_{l,j}(a)$ die Anzahl der Liniensegmente auf die das Patching Lemma angewendet wurde, wobei l ein Liniensegment, j die Tiefe von l, auf der das Patching Lemma angewendet wurde, und a die Verschiebung in x-Richtung bezeichne. Für die Summe aller Segmente aller Iterationen, auf die das Patching Lemma angewendet wurde, gilt:

$$\sum_{j \geq 1} c_{l,j}(a) \leq \frac{t(\pi, l)}{s - 3}$$

Dies gilt, da π l nach Definition nur $t(\pi, l)$ mal schneidet und bei jedem Anwenden des Patching Lemmas die mindestens $s + 1$ Schnitte (es müssen so viele sein, da sonst das Patching Lemma nicht angewendet würde) auf 4 reduziert. Also werden bei jedem Anwenden des Patching Lemmas mindestens $s - 3$ der $t(\pi, l)$ Schnitte eliminiert. Dies bedeutet, dass, wenn das Patching Lemma $\frac{t(\pi,l)}{s-3}$ mal angewendet wurde, die Schnitte von π mit L_i weniger als s sind. Nach dem Patching Lemma gilt dann für das Erhöhen der Kosten von π:

$$\text{Erhöhen der Kosten von } \pi \leq \sum_{j \geq i} c_{l,j}(a) \cdot g \cdot \frac{L}{2^j}$$

wobei $\frac{L}{2^j}$ die Länge des Liniensegmentes, also s im Patching Lemma ist. Dies muß natürlich für jedes der 2^i Segmente von L_i durchgeführt werden. Dies führt zu dieser Abschätzung der Kosten für ein Liniensegment l:

$$\begin{aligned}
E_a \quad & [\text{Kosten die } l \text{ zugeschrieben werden bei horizontaler Verschiebung } a] \\
= \quad & \sum_{i \geq 1} \frac{2^i}{L} \cdot \text{Kosten für die iterative Anwendung des Patching Lemma s} \\
\leq \quad & \sum_{i \geq 1} \frac{2^i}{L} \sum_{j \geq i} c_{l,j}(a) \cdot g \cdot \frac{L}{2^j} \\
= \quad & g \cdot \sum_{j \geq 1} \frac{c_{l,j}(a)}{2^j} \cdot \sum_{i \leq j} 2^i \\
\leq \quad & g \cdot \sum_{j \geq 1} 2 \cdot c_{l,j}(a) \\
\leq \quad & \frac{2g \cdot t(\pi, l)}{s-3}
\end{aligned}$$

Dabei gibt $\frac{2^i}{L}$ nach Gleichung 2 die Wahrscheinlichkeit an, dass l die minimale Tiefe i hat. Die Kosten für jede mögliche Tiefe i werden hier mit ihrer Wahrscheinlichkeit gewichtet und addiert, so dass eine Abschätzung der wahrscheinlichen Kosten für ein Liniensegment ensteht.

Damit ist π so modifiziert worden, dass er $(m,s)-light$ ist. Das Struktur Lemma verlangt jedoch außerdem, dass die Schnitte von π mit den Kanten der Quadrate nur an Portalen erfolgen. Damit π diese Anforderung erfüllt, wird einfach jedes Liniensegment von π, welches ein Quadrat schneidet an diesem Schnittpunkt geteilt. Die beiden dadurch enstandenen Teilstücke werden wieder verbunden, indem parallel zur Kante des Quadrates von beiden Seiten aus eine Verbindung zum nächsten Portal erzeugt wird. An dem Portal wird schließlich die Verbindung wieder hergestellt. Dadurch erhöhen sich die Kosten für π um $2 \cdot \frac{1}{2} \frac{L}{2^i m}$ pro Schnitt in einem Liniensegment der Tiefe i. Insgesamt erhöhen sich die erwarteten Kosten für l um:

$$\sum_{i=1}^{\log L} \frac{2^i}{L} \cdot t(\pi, l) \cdot \frac{L}{2^i m} = \frac{t(\pi, l) \log L}{m}$$

Mit $m \geq 2s \log L$ ergibt dies $t(\pi, l)/2s$.
Zusammengenommen erhöhen die beiden Modifikationen an π diesen also um:

$$\frac{2g \cdot t(\pi, l)}{s-3} + \frac{t(\pi, l)}{2s} \leq \frac{3g \cdot t(\pi, l)}{s}$$

Wobei für die letzte Abschätzung $s > 15$ vorausgesetzt wird.
Damit wäre gezeigt, dass die Kosten ein Liniensegment (m,s)-light zu ma-

chen $\leq \frac{3g \cdot t(\pi,l)}{s}$ sind. Dabei wurde jedoch vorausgesetzt, dass durch das Anwenden des Patching Lemmas die Zahl der Schnitte auf den horizontalen Linien gleich bleibt. Dies ist jedoch nicht unbedingt der Fall. Jedoch treten dadurch nur konstant viele zusätzliche Schnitte pro Liniensegment auf. Denn wenn durch das Anwenden des Patching Lemmas auf die vertikalen Liniensegmente auf einem horizontalen Liniensegment mehr als 2 Schnitte hinzugefügt werden, können diese durch Anwenden des Patching Lemmas auf 2 reduziert werden. Da die Teilstücke der vertikalen Liniensegmente, die die horizontalen schneiden übereinander liegen, besitzen sie keine horizontale Ausdehnung und daher erhöhen sich die Kosten des Pfades durch das Anwenden des Patching Lemmas auf diese auch nicht. Somit können nicht mehr als 4 Schnitte pro Liniensegment durch das Anwenden des Patching Lemmas hinzukommen (wiederum wegen der Möglichkeit, dass die Liniensegmente umgebrochen sind) und der erzeugte *Salesman path* ist $(m, r) - light$, denn es gilt nach Vorraussetzung $r = s + 4$. □