Distanzerhaltende Approximation von Kantenzügen

Nikolas Klug

27. März 2018

Zusammenfassung

asdf

1 Einführung und Definitionen

Sei $d \geq 1$ und $p_1, p_2, \ldots, p_n$ ein Kantenzug P, dessen Knoten alle in $\mathbb{R}^d$ liegen. Ein solcher Pfad kann mitunter eine hohe Zahl von Knoten besitzen, die sehr nah beieinander liegen und leicht durch deutlich weniger Kanten approximiert werden können, wobei sich wichtige Parameter nicht stark ändern. In der Fachliteratur werden dabei zahlreiche Kriterien aufgeführt, einige davon sind die Fläche, die der Pfad einschließt und die Distanz bzw. die Länge des Pfades. In dieser Arbeit soll es um Algorithmen gehen, die einen solchen Pfad unter Einhaltung der Länge approximieren. Art der Approximationen: - Flächenerhaltende Approximation + Anwendungen
- Distanzerhaltende Approximatioin + Anwendungen
- tolerance zone, infinite beam, uniform measure...
(- Vor- und Nachteile)

Kantenzug, Knoten, Kanten, analog zu Graphen "KÜRZESTEt-distanzerhaltende Approximation

Seien $u, v \in \mathbb{R}^d$, dann definieren wir $|uv|$ als den euklidischen Abstand (bzw. als die $L2 - Norm$) dieser beiden Punkte. Sind $v_i, v_j \in P$ dann sei $\delta(v_i, v_j) \coloneqq \sum_{i=1}^{n-1} |v_i v_{i+1}|$, also der euklidische Abstand dieser beiden Punkte entlang des Pfades P

Definition 1.1 (t-distanzerhaltend)**.** *Sind $p_i, p_j \in P$, dann ist die Kante (p_i, p_j) genau dann* t-distanzerhaltend*, wenn $\delta(p_i, p_{i+1}) \leq t \cdot |p_i p_j|$.*

Lemma 1.2. *Für alle Knoten $p_i, p_j \in P$ gilt $|p_i p_j| \leq \delta(p_i, p_j)$*

Beweis. Dies folgt direkt aus der Dreiecksungleichung in $\mathbb{R}$. □

Definition 1.3 (t-distanzerhaltende Approximation)**.** *Ein Kantenzug $Q = (p_{i_1}, p_{i_2}, \ldots, p_{i_j})$ ist genau dann eine* t-distanzerhaltende Approximation von P*, wenn*

(1) $1 = i_1 < i_2 < \ldots < i_j = n$

(2) Für alle $1 \leq l < j$ ist die Kante $(p_{i_l}, p_{i_{l+1}})$ des Pfades t-distanzerhaltend

Wir nennen eine t-distanzerhaltende Approximation eines Pfades am *kürzesten* bzw. eine *Kürzeste*, falls sie die geringst mögliche Zahl an Knoten besitzt.

Aus Definition 1.3 geht direkt hervor, dass falls $1 \leq t < t'$ jede t-distanzerhaltende Approximation eines Pfades, auch eine t'-distanzerhaltende Approximation dieses Pfades ist.

Lemma 1.4. *Sei* $1 \leq t < t'$

- Definition t-distanzerhaltend
- Definition t-distanzerhaltende Approximation
- Definition MVPS
- Definition MDPS

Beispiel Bild

2 Exakte Algorithmen für MVPS und MDPS

Zu Beginn betrachten wir zwei einfache exakte Algorithmen. Sei wieder $d \geq 1$, $t > 1$ und $P = (p_1, p_2, \ldots, p_n)$ ein Pfad in $\mathbb{R}^d$. Sei weiter P^* die Menge der t-distanzerhaltenden Approximationen, die die geringste Knotenzahl besitzen.

Wir konstruieren jetzt den gerichteten Graphen $G_t = (V, E_t)$, wobei V genau aus den Knoten des Pfades P besteht und $E_t = \{(p_i, p_j) \in V \times V |\, i < j \text{ und } (p_i, p_j) \text{ ist } t\text{-distanzerhaltend}\}$. E_t ist also gerade die Menge aller t-distanzerhaltenden Kanten zwischen Knoten aus V. Zunächst beobachten wir, dass jede t-distanzerhaltende Approximation von P einem Pfad in G_t entspricht, da G_t alle t-distanzerhaltenden Kanten zwischen Knoten von P enthält. Andererseits ist auch jeder Pfad $Q = (p_{i_1}, p_{i_2}, \ldots, p_{i_k})$ mit $1 = i_1 < i_2 < \ldots < i_k = n$ in G_t eine t-distanzerhaltende Approximation von P, da nur t-distanzerhaltende Kanten verwendet werden. Daraus folgt, dass auch P^* in G_t liegt. Jetzt müssen wir also nur noch ein Element aus P^* ermitteln. Das ist leicht: Wir führen eine Breitensuche in G_t mit Startknoten p_1 durch, bei der wir jeden Knoten mit der Nummer des Knotens beschriften, von dem aus er zum ersten Mal entdeckt wurde. Am Ende lesen wir diese Beschriftung bei p_n beginnend solange aus, bis wir p_1 erreichen. Der dadurch enstandene Pfad ist dann aufgrund der Eigenschaften der Breitensuche in P^*. Nun betrachten wir noch die Laufzeit: Die Konstruktion von G_t gelingt uns in $O(n^2)$, da wir für maximal $\binom{n}{2} = O(n^2)$ Kanten überprüfen müssen, ob diese t-distanzerhaltend sind. Sei m die Zahl der Kanten in G_t, dann wissen wir aus [1], dass die Breitensuche $O(n+m)$ Zeit dauert. In unserem Fall ist $O(m) = O(n^2)$, und somit dauert die Breitensuche auch $O(n^2)$. Insbesondere haben wir:

Satz 2.1. *Das Minimum Vertex Path Simplification Problem kann für Pfade mit n Knoten $O(n^2)$ gelöst werden.*

Als nächstes wollen wir uns überlegen, wie man das MDPS-Problem für eine feste Anzahl von Knoten k lösen kann. Sei im Folgenden κ_t die geringst mögliche Zahl von Knoten für eine t-distanzerhaltende Approximation von P.

Lemma 2.2. *Sind $t, t' \in \mathbb{R}$ und $t < t'$, dann ist $\kappa_t \geq \kappa_{t'}$.*

Beweis. Wäre $\kappa_{t'} < \kappa_t$, hätte eine kürzeste t'-distanzerhaltende Approximation weniger Knoten als eine kürzeste t-distanzerhaltende. Aber jede t-distanzerhaltende Approximation von P ist nach LEMMA XX auch eine t'-distanzerhaltende Approximation. Das ist ein Widerspruch. □

Da G_t maximal $O(n^2)$ Kanten enthält, gibt es eine endliche Zahl von t-Werten. Wir müssen also nur noch aus diesen Werten den geringsten Wert t^* ermitteln, der gerade noch k Knoten oder weniger hat. Dazu definieren wir zunächst für $1 \leq i < j \leq n$ $t^*_{ij} \coloneqq \frac{\delta(p_i,p_j)}{|p_i p_j|}$, also als die Abweichung der Kante (p_i, p_j) vom Pfad. Sei nun $M \coloneqq \{t^*_{ij} \mid 1 \leq i < j \leq n\}$. Wir wissen, dass $t^* \in M$, da die gesuchte Approximation eine Kante mit maximalen t-Wert hat, und M gerade alle diese enthält. Wegen Lemma 2.2 wissen wir, das sich die κ_t umgekehrt proportional zu den t-Werten verhalten. Sortieren wir jetzt M zu M', können wir in M' nach t^* suchen. Da M $O(n^2)$ Elemente enthält, können wir M nach [1] in $O(n^2 \log n^2) = O(n^2 \log n)$ sortieren. Für die Suche verwenden wir eine Binärsuche, bei der wir jeweils für den aktuell betrachteten t-Wert das MVPS lösen und dann abhängig vom Ergebnis entweder im rechten oder linken Teilbereich weitersuchen. Eine gewöhnliche Binärsuche dauert bekanntermaßen $O(\log n)$ und mit Satz 2.1 ergibt sich auch hier eine Laufzeit von $O(n^2 \log n)$. Wir halten fest:

Satz 2.3. *Das Minimum Dilation Path Simplification Problem kann für Pfade mit n Knoten in $O(n^2 \log n)$ gelöst werden.*

Damit beschließen wir das Kapitel über exakte Algorithmen für das MVPS- und das MDPS-Problem und wenden uns approximativen Lösungen zu.

3 Approximative Algorithmen

3.1 Well-separated Pair Decomposition

Definition 3.1 (wohl-separiert)**.** *Sei $s > 0$ und A und B zwei endliche Mengen von Punkten im $\mathbb{R}^d$. A und B heißen wohl-separiert in Bezug zu s (engl. well-separated), falls es zwei disjunkte Bälle C_A und C_B gibt, die denselben Radius R haben, wobei $A \subseteq C_A$ und $B \subseteq C_B$ und die euklidische Distanz zwischen C_A und C_B mindestens $s \cdot R$ beträgt.*

Das folgende Lemma hält zwei wichtige Eigenschaften von zwei wohl-separierten Mengen A und B fest.

Lemma 3.2. *Seien $a, a' \in A$ und $b, b' \in B$. Dann gilt:*

(1) $|aa'| \leq \frac{2}{s} \cdot |a'b'|$

(2) $|a'b'| \leq (1 + \frac{4}{s}) \cdot |ab|$

Beweis. Zu 1. Ist r der Radius von C_A und C_B, so gilt $|aa'| \leq 2 \cdot r$. Da A und B wohl-separiert sind, gilt $|a'b'| \geq s \cdot r$, was äquivalent ist zu $r \leq \frac{|a'b'|}{s}$. Durch Einsetzen folgt dann die Behauptung. Zu 2. Da A und B wohl-separiert in Bezug zu s sind, und C_A und C_B beide denselben Radius r haben, gilt $|a'b'| \leq s \cdot r + 4 \cdot r$. Ausklammern rechts ergibt $(1 + \frac{4}{s}) \cdot s \cdot r$. Da ja auch $s \cdot r \leq |ab|$, folgt durch Einsetzen die Behauptung. □

Definition 3.3 (well-separated pair decomposition)**.** *Sei $S \subseteq \mathbb{R}^d$ und $s > 0$. Eine Folge $(A_i, B_i)_{1 \leq i \leq m}$ von nicht-leeren Teilmengen von S ist genau dann eine* Zerlegung in wohl-separierte Paare (engl. well-separated pair decomposition; WSPD)*, wenn gilt:*

(1) $A_i \cap B_i = \emptyset$

(2) Für alle $p, q \in S$ gibt es genau einen Index $1 \leq i \leq m$, sodass entweder $p \in A_i$ und $q \in B_i$ oder $q \in A_i$ und $p \in B_i$.

(3) A_i *und* B_i *sind wohl-separiert in Bezug zu* s

m nennen wir dabei die *Größe* der WSPD.

Die WSPD bildet eine wichtige Grundlage für die beiden Algorithmen, die wir im Folgenden betrachten werden. . . . haben gezeigt, dass man eine WSPD der Größe $m = O(n)$ in $O(n \log n)$ Zeit berechnen kann. Dabei wird zunächst ein sogenannter *fairer Split Tree* berechnet, aus dem dann in $O(s^d n)$ Zeit eine WSPD erstellt werden kann. Wir werden sehen, dass es für unseren Anwendungsfall genügt, eine WSPD für Mengen von Punkten aus $\mathbb{R}$ zu erstellen. Für diesen 1-dimensionalen Fall kann der faire Split Tree mit Hilfe eines einfachen Algorithmus berechnet werden.

Sei S' eine endliche Teilmenge von $\mathbb{R}$ und $|S'| = n$. Wir können davon ausgehen, dass uns diese Menge sortiert in einem Array $S[1..n]$ vorliegt und werden später sehen, dass das bei unserem Algorithmus auch tatsächlich der Fall ist. ABBILDUNG X stellt einen Algorithmus dar, der diesen Split Tree T erstellt. Bei T handelt es sich um einen Binärbaum, an dessen Blättern die Werte von S in von links nach rechts aufsteigend sortierter Reihenfolge gespeichert sind. Für jeden inneren Knoten wird zusätzlich das Intervall in dem die Blätter des von ihm induzierten Teilbaumes liegen gepsiechert. Da T n Blätter hat, erstellen wir $O(n)$ Knoten. Dabei müssen wir aber in ZEILE 88 jedesmal eine Binärsuche durchführen, die $O(\log n)$ Zeit kostet. Somit ergibt sich für das Erstellen von T eine Gesamtlaufzeit von $O(n \log n)$. Betrachten wir jetzt zwei innere Knoten p und q von T. Seien $[i, j]$ und $[k, l]$ die Intervalle, die wir mit p und q gespeichert haben und

$$R \coloneqq \max(i - j, k - l)$$

Nach Definition 3.1 sind die beiden Intervalle genau dann wohl-separiert, wenn

$$k - j \geq R \cdot s \text{ oder } i - l \geq R \cdot s$$

Der in ABBILDUNG XX dargestellte Algorithmus berechnet dann aus dem fairen Split-Tree eine WSPD. Wie XY und ZA gezeigt haben, läuft ALG in $O(s \cdot n)$ und gibt tatsächlich eine WSPD aus. Der Beweis dafür ist allerdings so langwierig und kompliziert, dass wir hier darauf verzichten, ihn explizit aufzuführen. Interessierte können ihn aber auf SEITE 88 in CITE X nachlesen. Insgesamt erhalten wir für das Erstellen der WSPD eine Gesamtlaufzeit von $O(n \log n + s \cdot n)$.

3.2 Algorithmus für das MVPS

Nach dem wir jetzt einige Vorarbeit geleistet haben, werden wir in diesem Kapitel sehen, wie man das MVPS-Problem mit Hilfe einer WSPD bis auf ein ϵ genau approximieren kann - und dass (für festes t und ϵ) in $O(n \log n)$.

Sei $P = (p_1, p_2, \ldots, p_n)$ ein Kantenzug in $\mathbb{R}^d$. Für unseren Anwendungsfall genügt es, eine eindimensionale Version $S = (x_1, x_2, \ldots, x_n)$ dieses Pfades zu betrachten. Diese erhalten wir so: Für alle $1 \leq i \leq n$ ist $x_i = \delta(p_1, p_i)$. Als nächstes berechnen wir für ein festes $s > 0$ zunächst den Split Tree T und danach eine zugehörige WSPD $\{A_i, B_i\}_{1 \leq i \leq m}$ von S. Wegen Eigenschaft (3) der WSPD sind für alle i entweder alle Elemente in A_i kleiner als die in B_i oder umgekehrt. Wir werden o.B.d.A annehmen, dass alle Elemente, die in A_i enthalten sind, kleiner sind, als alle Elemente in B_i, da wir einfach bei der Erstellung der WSPD die beiden Mengen passend benennen.

Lemma 3.4. *Seien $p, p', q, q' \in P$ und sei i ein solcher Index, dass für $x = \delta(p_1, p)$, $x' = \delta(p_1, p')$, $y = \delta(p_1, q)$ und $y' = \delta(p_1, q')$ $x, x' \in A_i$ und $y, y' \in B_i$ sind. Ist weiter $1 \leq t < \frac{s^2}{4s+16}$ und ist das Tupel (p, q) t-distanzerhaltend, dass ist (p', q') t'-distanzerhaltend, wobei t' gegeben ist durch*

$$t' = \frac{(1 + \frac{4}{s}) \cdot t}{1 - 4(1 + \frac{4}{s}) \cdot \frac{t}{s}}$$

Beweis.

$$
\begin{aligned}
\delta(p',q') &= |x'y'| \\
&\leq (1+\frac{4}{s})\cdot|xy| && \text{(Lemma 3.2 (2))} \\
&= (1+\frac{4}{s})\cdot\delta(p,q) \\
&\leq (1+\frac{4}{s})t\cdot|pq| && \text{((p,q) ist } t\text{-distanzerhaltend)} \\
&\leq (1+\frac{4}{s})t\cdot(|pp'|+|p'q'|+|q'q|) && \text{(Dreiecksungleichung in } \mathbb{R}) \\
&\leq (1+\frac{4}{s})t\cdot(\delta(p,p')+|p'q'|+\delta(q',q)) && \text{(Lemma 1.2)} \\
&= (1+\frac{4}{s})t\cdot(|xx'|+|p'q'|+|yy'|) \\
&\leq (1+\frac{4}{s})t\cdot(\frac{2}{s}\cdot|x'y'|+|p'q'|+\frac{2}{s}\cdot|x'y'|) && \text{(Lemma 3.2 (1))} \\
&= (1+\frac{4}{s})t\cdot(\frac{4}{s}\cdot\delta(p',q')+|p'q'|) \\
&= 4(1+\frac{4}{s})\frac{t}{s}\cdot\delta(p',q')+(1+\frac{4}{s})t\cdot|p'q'|
\end{aligned}
$$

Also ist

$$
\begin{aligned}
& \delta(p',q') \leq 4(1+\frac{4}{s})\frac{t}{s}\cdot\delta(p',q')+(1+\frac{4}{s})t\cdot|p'q'| \\
\Leftrightarrow\quad & \delta(p',q')\cdot(1-4(1+\frac{4}{s})\frac{t}{s}) \leq (1+\frac{4}{s})t\cdot|p'q'| \\
\Leftrightarrow\quad & \delta(p',q') \leq t'\cdot|p'q'|
\end{aligned}
$$

□

Sei jetzt $0<\epsilon<\frac{1}{3}$ und $1\leq t$. Sei

$$s=\frac{12+24(1+\frac{\epsilon}{3})\cdot t}{\epsilon}$$

Lemma 3.5. *Seien $p,p',q,q'\in P$ wie in Lemma 3.4. Dann gilt*

(1) Ist (p,q) t-distanzerhaltend, dann ist (p',q') $(1+\frac{\epsilon}{3})t$-distanzerhaltend.

(2) Ist (p,q) (p',q') $(1+\frac{\epsilon}{3})t$-distanzerhaltend, dann ist (p',q') $(1+\epsilon)t$-distanzerhaltend.

Beweis. Zu 1. □

... Wir nennen das Paar (A_i,B_i) (t, epsilon)-distanzerh. falls ... Zur Erinnerung: Die Mengen A_i und B_i enthalten die Elemente $x_k=\delta(p_1,p_k)$. Als nächstes konstruieren wir aus der WSPD einen gerichteten Graphen H. Dabei sind die Knoten von H genau die $2m$ Mengen A_i und B_i und die Kanten sind wie folgt definiert:

(1) Für alle $1 \leq i \leq m$ ist (A_i, B_i) genau dann eine Kante, wenn (A_i, B_i) (t, ϵ)-distanzerhaltend ist und $x_n \in B_i$

(2) Für alle $1 \leq i < j \leq m$ ist (A_i, A_j) genau dann eine Kannte, wenn (A_i, B_i) (t, ϵ)-distanzerhaltend ist und $A_j \cap B_i \neq \emptyset$

Satz 3.6. *Jede t-distanzerhaltende Approximation $Q = (q_1, q_2, \ldots, q_k)$ von P entspricht einem Pfad R der Länge k in H von einer Menge A_i, die x_1 enthält, zu einer Menge B_j, die x_n enthält*

Beweis. Sei y_i das Element der Menge S, für das $y_i = \delta(p_1, q_i)$ gilt. Da nach Bedingung ja $q_1 = p_1$ gilt, ist also auch $y_1 = x_1$. Sei weiter i_1 ein solcher Index, für den $y_1 \in A_{i_1}$ und $y_2 \in B_{i_1}$. Dann hat der Pfad R A_{i_1} als ersten Knoten.

Nehmen wir jetzt an, dass wir bereits für ein l mit $1 \leq l < k-1$ den Kantenzug $(q_1, \ldots, q_l)$ zu dem Teilpfad $(A_{i_1}, \ldots, A_{i_l})$ von R umgewandelt haben, sodass $y_l \in A_{i_l}$ und $y_{l+1} \in B_{i_l}$. Wir wählen jetzt i_{l+1} als den Index, für den $y_{l+1} \in A_{i_{l+1}}$ und $y_{l+2} \in B_{i_{l+1}}$ ist. Solch ein Index existiert nach Definition der WSPD. Wir wissen, dass (q_l, q_{l+1}) t-distanzerhaltend ist und $y_l \in A_{i_l}$ und $y_{l+1} \in B_{i_l}$ liegt. Aus LEMMA XX.1 folgt dann, dass das Tupel (A_i, B_i) (t, ϵ)-distanzerhaltend ist. Des Weiteren ist der Schnitt von $A_{i_{l+1}}$ mit B_{i_l} nicht leer, da y_{l+1} in beiden Mengen liegt. Es folgt, dass (A_{i_l}, A_{i_l+1}) eine Kante in H ist. Wir haben damit also $(q_1, \ldots, q_l, q_{l+1})$ zu dem Pfad $(A_{i_1}, \ldots, A_{i_l}, A_{i_{l+1}})$ umgewandelt, sodass $y_{l+1} \in A_{i_{l+1}}$ und $y_{l+2} \in B_{i_{l+1}}$.

Nehmen wir an, dass wir bereits $(q_1, \ldots, q_l, q_{k-1})$ zu dem Pfad $(A_{i_1}, \ldots, A_{i_{k-1}})$ umgewandelt haben, wobei $y_{k-1} \in A_{i_{k-1}}$ und $y_k \in B_{i_{k-1}}$. Es ist $y_k = \delta(p_1, q_k) = \delta(p_1, p_n) = x_n \in B_{i_{k-1}}$. Da die Kante (q_{k-1}, q_k) t-distanzerhaltend ist, ist wieder wegen LEMMA XX.1 $(A_{i_{k-1}}, B_{i_{k-1}})$ (t, ϵ)-distanzerhaltend und es folgt, dass $(A_{i_{k-1}}, B_{i_{k-1}})$ eine Kante in H ist. Wir fügen $B_{i_{k-1}}$ zum Pfad hinzu, und erhalten als Gesamtergebnis $R = (A_{i_1}, \ldots, A_{i_{k-1}}, B_{i_{k-1}})$. □

Wir haben also gezeigt, dass jede t-distanzerhaltende Approximation von P einem Pfad mit der gleichen Zahl von Knoten in H entspricht. Der nächste Satz zeigt, dass dies auch umgekehrt der Fall ist, unter der Einschränkung, dass die Approximation um einen kleinen Teil länger sein kann, als gewünscht.

Satz 3.7. *Jeder Pfad $R = (A_{i_1}, \ldots, A_{i_{k-1}}, B_{i_{k-1}})$ in H mit $x_1 \in A_{i_1}$ und $x_n \in B_{i_{k-1}}$ entspricht einer $(1+\epsilon)t$-distanzerhaltenden Approximation Q von P, die k Knoten besitzt.*

Beweis. Sei wieder y_i das Element der Menge S, für das $y_i = \delta(p_1, q_i)$ gilt. Da x_1 in A_{i_1}, können wir als ersten Knoten von Q $q_1 = p_1$ wählen.

Nehmen wir an, dass wir bereits für ein l mit $1 \leq l < k-1$ den Teilpfad $(A_{i_1}, \ldots, A_{i_l})$ zu dem Kantenzug $(q_1, \ldots, q_l)$ umgewandelt haben, sodass $y_1 (= x_1) \in A_{i_1}$ und für alle $1 < j \leq l$ $y_j \in A_{i_j} \cap B_{i_{j-1}}$. Betrachten wir jetzt die Kante $(A_{i_l}, A_{i_{l+1}})$, gibt es ein $y_{l+1} \in A_{i_{l+1}} \cap B_{i_l}$, da der Schnitt nach der Konstruktion von H nicht leer ist. Dann fügen wir das zu y_{l+1} gehörende q_{l+1} zum Kantenzug hinzu und erhalten $(q_1, \ldots, q_l, q_l + 1)$.

Nehmen wir an, dass wir bereits $(A_{i_1}, \ldots, A_{i_k-1})$ bereits zu $(q_1, \ldots, q_{k-1})$ konvertiert haben. Nach Voraussetzung ist $x_n \in B_{k-1}$. Wir wählen dann $q_k = p_n$ und fügen q_k zum Pfad Q hinzu. Insgesamt haben wir also gezeigt, wie man R zu einem Pfad Q mit gleich vielen Knoten umwandeln kann. Jetzt bleibt zu zeigen, dass Q auch tatsächlich $(1+\epsilon)t$-distanzerhaltend ist.

Dazu betrachten wir ein j mit $1 \leq j < k$. Nach unserer Konstruktion von Q ist $y_j \in A_{i_j}$ und $y_{j+1} \in B_{i_j}$. Die Kante A_{i_j} und B_{i_j} ist zudem (t, ϵ)-distanzerhaltend. Mit LEMMA XX.2 folgt dann, dass alle Tupel (a, b) mit $a \in A_{i_j}$ und $b \in B_{i_{j+1}}$ $(1 + \epsilon)t$-distanzerhaltend sind, insbesondere ist also die Kante zwischen den zu y_j und y_{j+1} gehörigen Knoten (q_j, q_{j+1}) $(1 + \epsilon)t$-distanzerhaltend. Diese Eigenschaft gilt für alle aufeinanderfolgenden Knoten in Q, woraus folgt, dass der von uns konstruierte Pfad Q also $(1 + \epsilon)t$-distanzerhaltend ist. □

3.3 Algorithmus für das MDPS

Heuristischer Algorithmus aus Paper mit allen Erklärungen

4 Fazit

Literatur

[1] Torben Hagerup. Vorlesungsskript Informatik III, WS 17/18.