Distanzerhaltende Approximation von Kantenzügen

Nikolas Klug

Universität Augsburg

17. Mai 2018

Grenze zwischen Deutschland und Österreich. Quelle: *Google Maps*, Stand 8. Mai 2018, https://www.google.de/maps/@47.5869372,11.6674982,9.06z

Grenze zwischen Deutschland und Österreich. Quelle: *Google Maps*, Stand 8. Mai 2018, https://www.google.de/maps/@47.5869372,11.6674982,9.06z

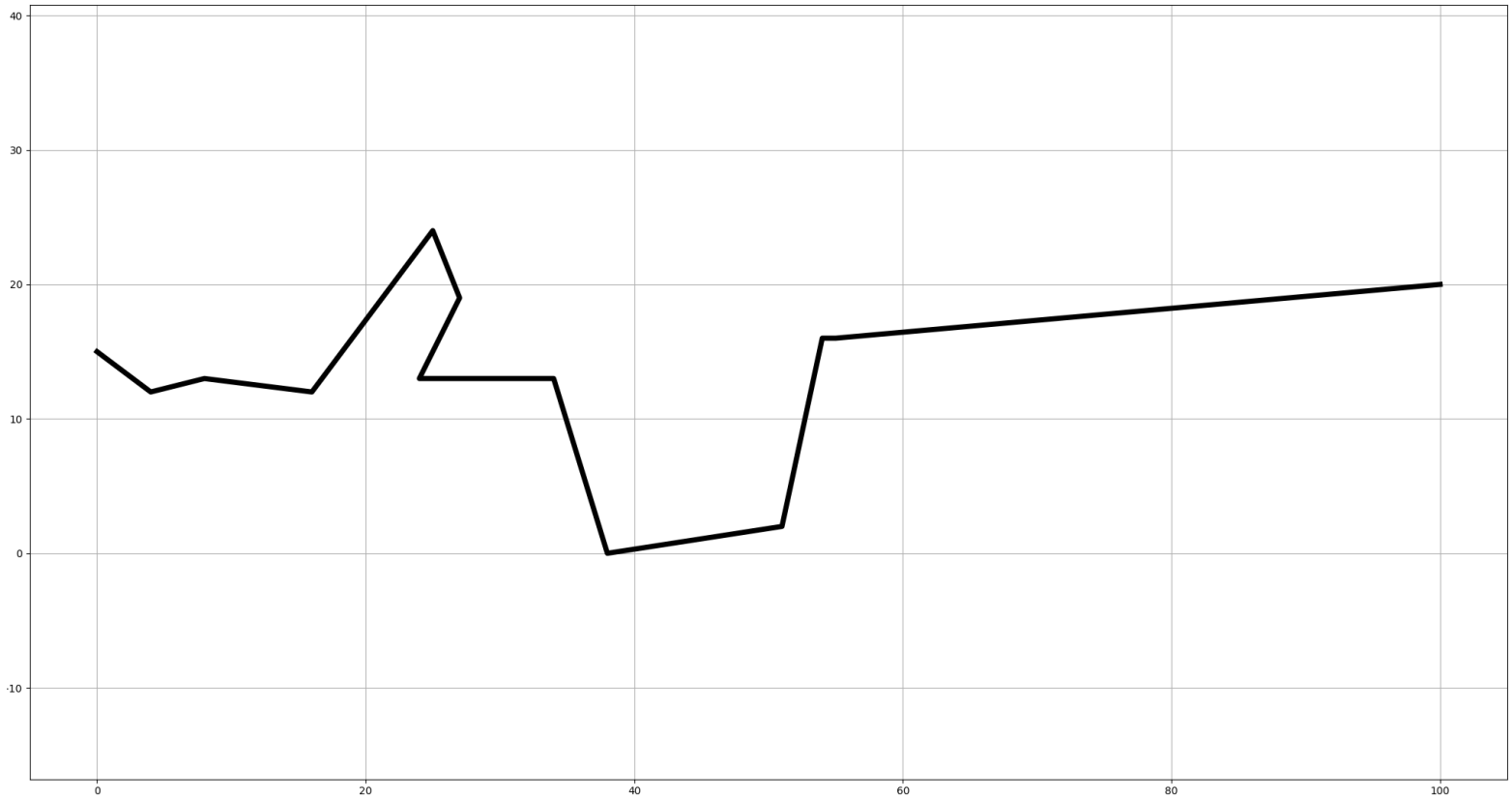
Grenze zwischen Deutschland und Österreich. Quelle: *Google Maps*, Stand 8. Mai 2018, https://www.google.de/maps/@47.5869372,11.6674982,9.06z

Grenze zwischen Deutschland und Österreich. Quelle: *Google Maps*, Stand 8. Mai 2018, https://www.google.de/maps/@47.5869372,11.6674982,9.06z


472
8
8
13
11
Bad Tölz
Schliersee
Tegernsee
Rottach-Egern
307
93
307
Kufstein
E45
181
178


Poland
Netherlands
London
Belgium
Paris
Vienna
Slovakia
Austria
Budapest
Hungary
France
Croatia
Serbia
Italy
Barcelona
Rome

Definition Kantenzug



Definition

Sei $n, d \in \mathbb{N}$ und für $1 \leq i \leq n$ $p_i \in \mathbb{R}^d$. Ein *(polygonaler) Kantenzug* $P = (p_1, p_2, \ldots, p_n)$ ist eine Aneinanderreihung von Geradensegmenten, die für $1 \leq i < n$ jeweils die Punkte p_i und p_{i+1} verbinden.

Definition t-distanzerhaltend

Sei $P = (p_1, \ldots, p_n)$ ein Kantenzug und $p_i, p_j \in P$.

- $|p_i p_j|$ ist die euklidische Distanz zwischen p_i und p_j.

Definition t-distanzerhaltend

Sei $P = (p_1, \ldots, p_n)$ ein Kantenzug und $p_i, p_j \in P$.

- $|p_i p_j|$ ist die euklidische Distanz zwischen p_i und p_j.
- $\delta(p_i, p_j) := \sum_{k=i}^{j-1} |p_k p_{k+1}|$ ist die Distanz entlang des Pfades.

Definition t-distanzerhaltend

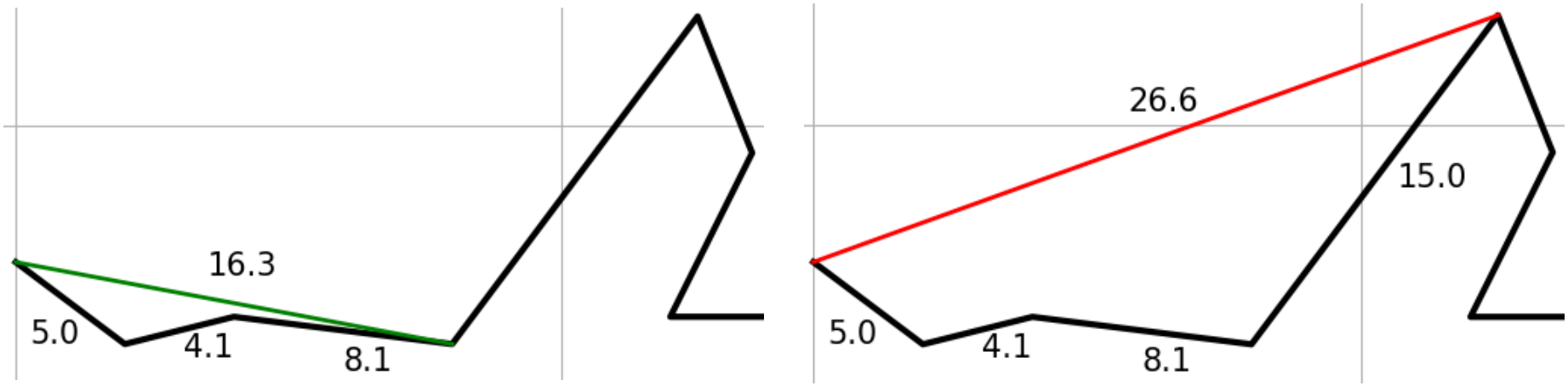
Sei $P = (p_1, \ldots, p_n)$ ein Kantenzug und $p_i, p_j \in P$.

- $|p_i p_j|$ ist die euklidische Distanz zwischen p_i und p_j.
- $\delta(p_i, p_j) := \sum_{k=i}^{j-1} |p_k p_{k+1}|$ ist die Distanz entlang des Pfades.

Definition

Seien $t \in \mathbb{R}$, $t \geq 1$ und $p_i, p_j \in P$. Dann ist die Kante (p_i, p_j) genau dann *t-distanzerhaltend*, wenn $\delta(p_i, p_j) \leq t \cdot |p_i p_j|$.

(p_i, p_j) *t-distanzerhaltend* $\Leftrightarrow \delta(p_i, p_j) \leq t \cdot |p_i p_j|$.



(a) t-distanzerhaltend (b) nicht t-distanzerhaltend

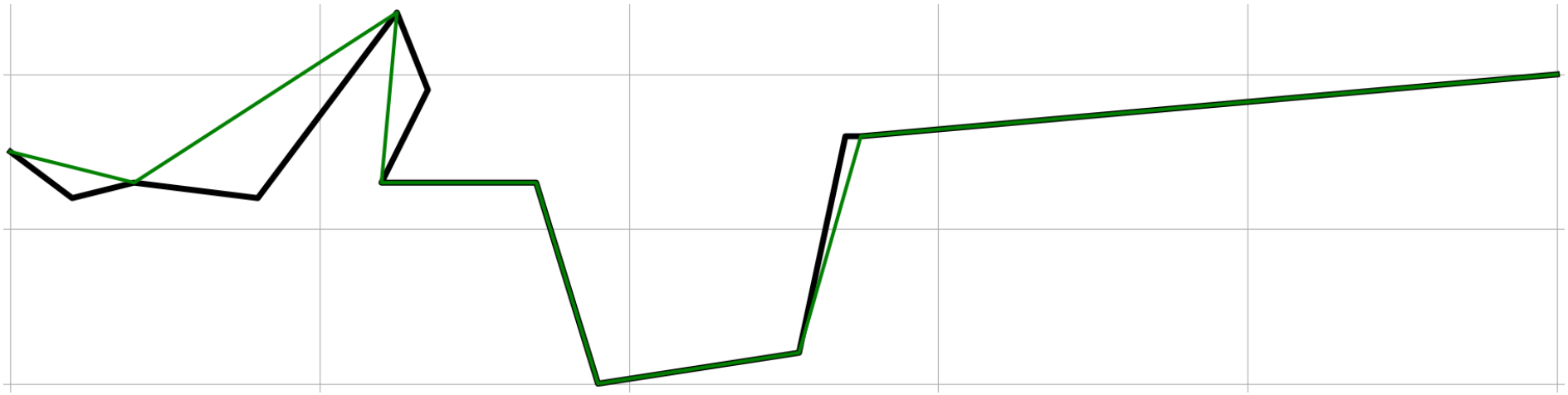
(nicht) t-distanzerhaltende Kanten für $t = 1.2$

Definition t-distanzerhaltende Approximation

Definition

Ein Kantenzug $Q = (p_{i_1}, p_{i_2}, \ldots, p_{i_k})$ ist genau dann eine *t-distanzerhaltende Approximation von* $P = (p_1, p_2, \ldots, p_n)$, wenn beide der folgenden Bedingungen gelten.

1. $1 = i_1 < i_2 < \ldots < i_k = n$.
2. Für alle $1 \leq l < k$ ist die Kante $(p_{i_l}, p_{i_{l+1}})$ des Kantenzugs t-distanzerhaltend.



Problemspezifikation

Definition (Minimum-Vertex-Path-Simplification)

Liegt ein polygonaler Kantenzug P und eine reelle Zahl $t \geq 1$ vor, soll eine minimale t-distanzerhaltende Approximation von P berechnet werden.

Definition (Minimum-Dilation-Path-Simplification)

Liegt ein polygonaler Kantenzug P und eine natürliche Zahl k vor, soll der kleinste Wert t bestimmt werden, für den eine t-distanzerhaltende Approximation von P mit maximal k Knoten existiert.

Exakter Algorithmus für das MVPS-Problem

Sei $P = (p_1, p_2, \ldots, p_n)$ ein Kantenzug und $t \geq 1$.

Schritt 1: Konstruktion des gerichteten Graphen $G_t = (V, E_t)$, wobei:

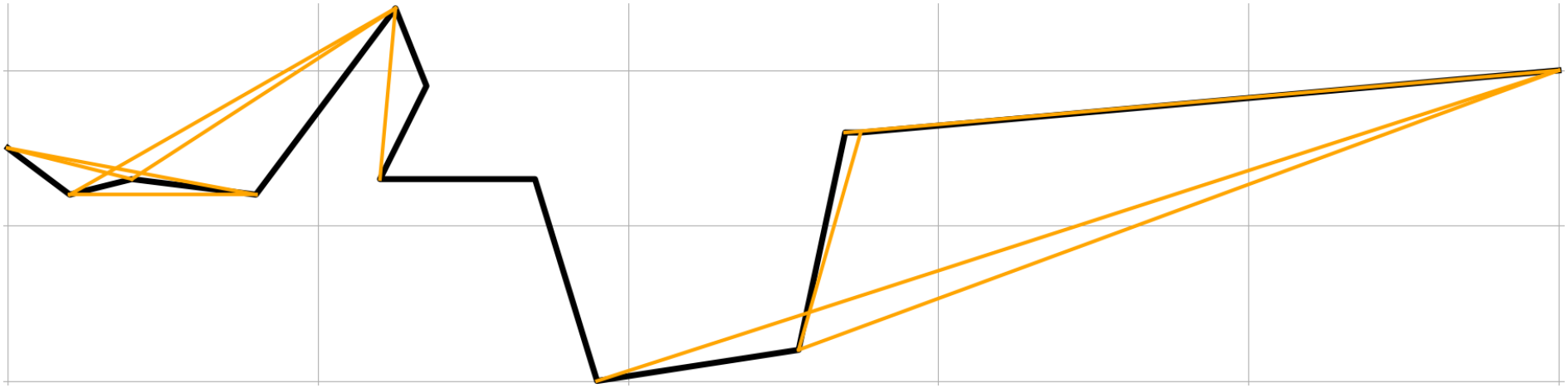
- $V = \{p_1, \ldots, p_n\}$
- $E_t = \{(p_i, p_j) \in V \times V |\ i < j$ und (p_i, p_j) ist t-distanzerhaltend$\}$

Exakter Algorithmus für das MVPS-Problem

Sei $P = (p_1, p_2, \ldots, p_n)$ ein Kantenzug und $t \geq 1$.

Schritt 1: Konstruktion des gerichteten Graphen $G_t = (V, E_t)$, wobei:

- $V = \{p_1, \ldots, p_n\}$
- $E_t = \{(p_i, p_j) \in V \times V \mid i < j$ und (p_i, p_j) ist t-distanzerhaltend$\}$



G_t für $t = 1.2$

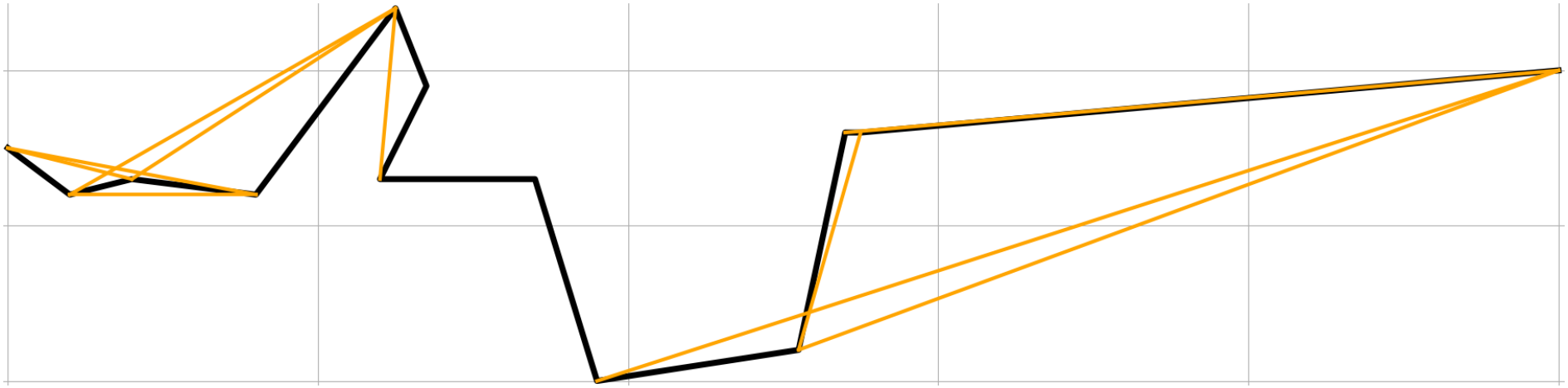
Exakter Algorithmus für das MVPS-Problem

Sei $P = (p_1, p_2, \ldots, p_n)$ ein Kantenzug und $t \geq 1$.

Schritt 1: Konstruktion des gerichteten Graphen $G_t = (V, E_t)$, wobei:

- $V = \{p_1, \ldots, p_n\}$
- $E_t = \{(p_i, p_j) \in V \times V \mid i < j$ und (p_i, p_j) ist t-distanzerhaltend$\}$

Schritt 2: Bestimmen eines kürzesten Pfades in G_t von p_1 nach p_n



G_t für $t = 1.2$

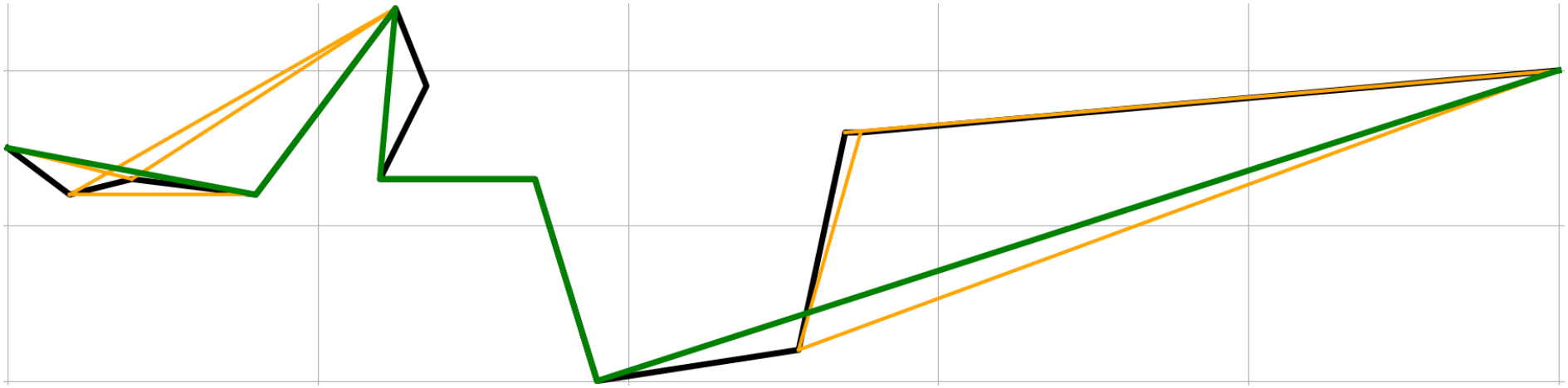
Exakter Algorithmus für das MVPS-Problem

Sei $P = (p_1, p_2, \ldots, p_n)$ ein Kantenzug und $t \geq 1$.

Schritt 1: Konstruktion des gerichteten Graphen $G_t = (V, E_t)$, wobei:

- $V = \{p_1, \ldots, p_n\}$
- $E_t = \{(p_i, p_j) \in V \times V \mid i < j$ und (p_i, p_j) ist t-distanzerhaltend$\}$

Schritt 2: Bestimmen eines kürzesten Pfades in G_t von p_1 nach p_n



Kürzester Pfad in G_t

Laufzeit

Schritt 1: Konstruktion des Graphen G_t $O(n^2)$

Laufzeit

Schritt 1: Konstruktion des Graphen G_t $O(n^2)$

Schritt 2: Breitensuche $O(n+m) = O(n^2)$

Laufzeit

Schritt 1: Konstruktion des Graphen G_t $O(n^2)$

Schritt 2: Breitensuche $O(n+m) = O(n^2)$

Satz

Das Minimum-Vertex-Path-Simplification Problem kann für Kantenzüge mit n Knoten in $O(n^2)$ Zeit gelöst werden.

Exakter Algorithmus für das MDPS-Problem

Sei $P = (p_1, \ldots, p_n)$ und k die gewünschte Knotenzahl.
Sei κ_t die Knotenzahl einer minimalen t-distanzerhaltenden Approximation von P.

Exakter Algorithmus für das MDPS-Problem

Sei $P = (p_1, \ldots, p_n)$ und k die gewünschte Knotenzahl.
Sei κ_t die Knotenzahl einer minimalen t-distanzerhaltenden Approximation von P.

Lemma

Sind $t, t' \in \mathbb{R}$ und $1 \leq t < t'$, dann ist $\kappa_t \geq \kappa_{t'}$.

Exakter Algorithmus für das MDPS-Problem

Sei $P = (p_1, \ldots, p_n)$ und k die gewünschte Knotenzahl.
Sei κ_t die Knotenzahl einer minimalen t-distanzerhaltenden Approximation von P.

Lemma

Sind $t, t' \in \mathbb{R}$ und $1 \leq t < t'$, dann ist $\kappa_t \geq \kappa_{t'}$.

Beweis.

Annahme: $\kappa_t < \kappa_{t'}$.

Exakter Algorithmus für das MDPS-Problem

Sei $P = (p_1, \ldots, p_n)$ und k die gewünschte Knotenzahl.
Sei κ_t die Knotenzahl einer minimalen t-distanzerhaltenden Approximation von P.

Lemma

Sind $t, t' \in \mathbb{R}$ und $1 \leq t < t'$, dann ist $\kappa_t \geq \kappa_{t'}$.

Beweis.

Annahme: $\kappa_t < \kappa_{t'}$.

$\Leftrightarrow$ Eine minimale t-distanzerhaltende Approximation von P hat echt weniger Knoten als eine minimale t'-distanzerhaltende.

Exakter Algorithmus für das MDPS-Problem

Sei $P = (p_1, \ldots, p_n)$ und k die gewünschte Knotenzahl.
Sei κ_t die Knotenzahl einer minimalen t-distanzerhaltenden Approximation von P.

Lemma

Sind $t, t' \in \mathbb{R}$ und $1 \leq t < t'$, dann ist $\kappa_t \geq \kappa_{t'}$.

Beweis.

Annahme: $\kappa_t < \kappa_{t'}$.

$\Leftrightarrow$ Eine minimale t-distanzerhaltende Approximation von P hat echt weniger Knoten als eine minimale t'-distanzerhaltende.

Aber: Jede t-distanzerhaltende Approximation von P ist auch eine t'-distanzerhaltende Approximation von P.
Das ist ein Widerspruch.

Exakter Algorithmus für das MDPS-Problem

Lemma

Sind $t, t' \in \mathbb{R}$ *und* $1 \leq t < t'$, *dann ist* $\kappa_t \geq \kappa_{t'}$.

Sei t^* die Lösung des Problems.

$t^*_{ij} := \frac{\delta(p_i,p_j)}{|p_i p_j|}$ heißt *Abweichung* der Kante (p_i, p_j) vom Kantenzug.

Schritt 1: Berechnen von $M := \{t^*_{ij} \mid 1 \leq i < j \leq n\}$

Exakter Algorithmus für das MDPS-Problem

Lemma

Sind $t, t' \in \mathbb{R}$ *und* $1 \leq t < t'$, *dann ist* $\kappa_t \geq \kappa_{t'}$.

Sei t^* die Lösung des Problems.

$t^*_{ij} := \frac{\delta(p_i,p_j)}{|p_i p_j|}$ heißt *Abweichung* der Kante (p_i, p_j) vom Kantenzug.

Schritt 1: Berechnen von $M := \{t^*_{ij} \mid 1 \leq i < j \leq n\}$

Schritt 2: Sortieren von M

Exakter Algorithmus für das MDPS-Problem

Lemma

Sind $t, t' \in \mathbb{R}$ *und* $1 \leq t < t'$, *dann ist* $\kappa_t \geq \kappa_{t'}$.

Sei t^* die Lösung des Problems.
$t^*_{ij} := \frac{\delta(p_i,p_j)}{|p_i p_j|}$ heißt *Abweichung* der Kante (p_i, p_j) vom Kantenzug.

Schritt 1: Berechnen von $M := \{t^*_{ij} \mid 1 \leq i < j \leq n\}$

Schritt 2: Sortieren von M

Schritt 3: Binäre Suche im M nach t^*:

- Lösen des MVPS-Problems für den aktuellen t-Wert.
- Sei κ_t die Knotenzahl der Lösung. Falls $\kappa_t \leq k$, so ist $t \geq t^*$.
  Sonst ist $\kappa_t > k$ und somit $t < t^*$.

Laufzeit

Schritt 1: Berechnen von M $O(n^2)$

Laufzeit

Schritt 1: Berechnen von M $O(n^2)$

Schritt 2: Sortieren von M $O(n^2 \log n^2) = O(n^2 \log n)$

Laufzeit

Schritt 1: Berechnen von M	$O(n^2)$
Schritt 2: Sortieren von M	$O(n^2 \log n^2) = O(n^2 \log n)$
Schritt 3: Binäre Suche	$O(n^2 \log n^2) = O(n^2 \log n)$

Satz

Das Minimum-Dilation-Path-Simplification Problem kann für Kantenzüge mit n Knoten in $O(n^2 \log n)$ Zeit gelöst werden.

Definition wohl-separiert

Seien $s > 0$ und A und B zwei endliche Mengen von Punkten in $\mathbb{R}^d$.

Definition

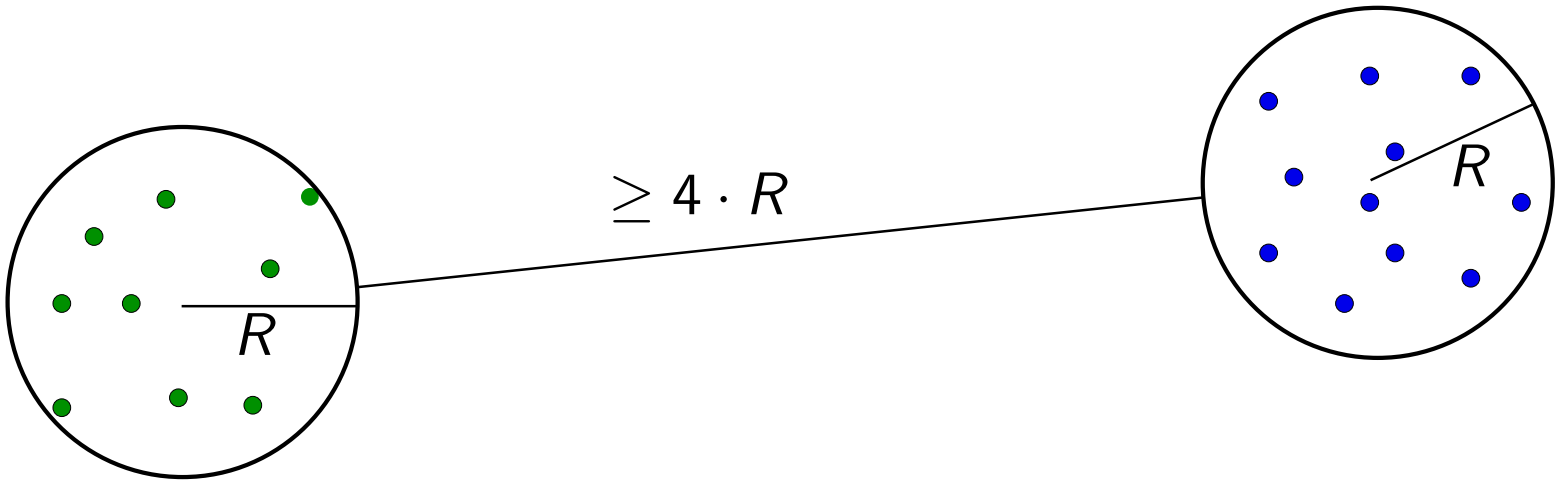
A und B heißen *wohl-separiert bezüglich* s, falls es zwei disjunkte Bälle C_A und C_B gibt, die denselben Radius R haben, sodass $A \subseteq C_A$ und $B \subseteq C_B$ und die euklidische Distanz zwischen den Rändern von C_A und C_B mindestens $s \cdot R$ beträgt.

Definition wohl-separiert

Seien $s > 0$ und A und B zwei endliche Mengen von Punkten in $\mathbb{R}^d$.

Definition

A und B heißen *wohl-separiert bezüglich* s, falls es zwei disjunkte Bälle C_A und C_B gibt, die denselben Radius R haben, sodass $A \subseteq C_A$ und $B \subseteq C_B$ und die euklidische Distanz zwischen den Rändern von C_A und C_B mindestens $s \cdot R$ beträgt.

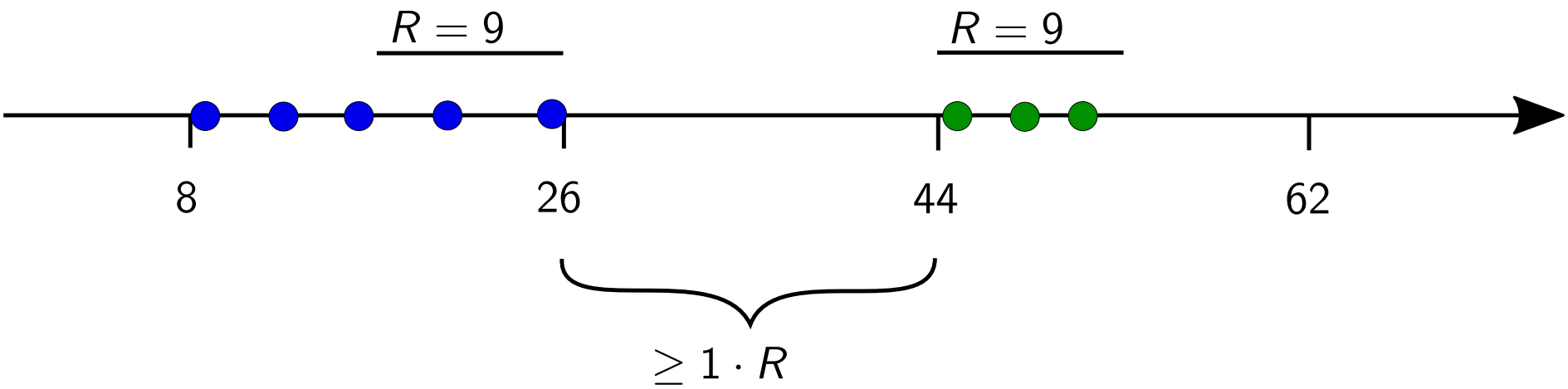


Definition wohl-separiert

Seien $s > 0$ und A und B zwei endliche Mengen von Punkten in $\mathbb{R}^d$.

Definition

A und B heißen *wohl-separiert bezüglich* s, falls es zwei disjunkte Bälle C_A und C_B gibt, die denselben Radius R haben, sodass $A \subseteq C_A$ und $B \subseteq C_B$ und die euklidische Distanz zwischen den Rändern von C_A und C_B mindestens $s \cdot R$ beträgt.



Definition WSPD

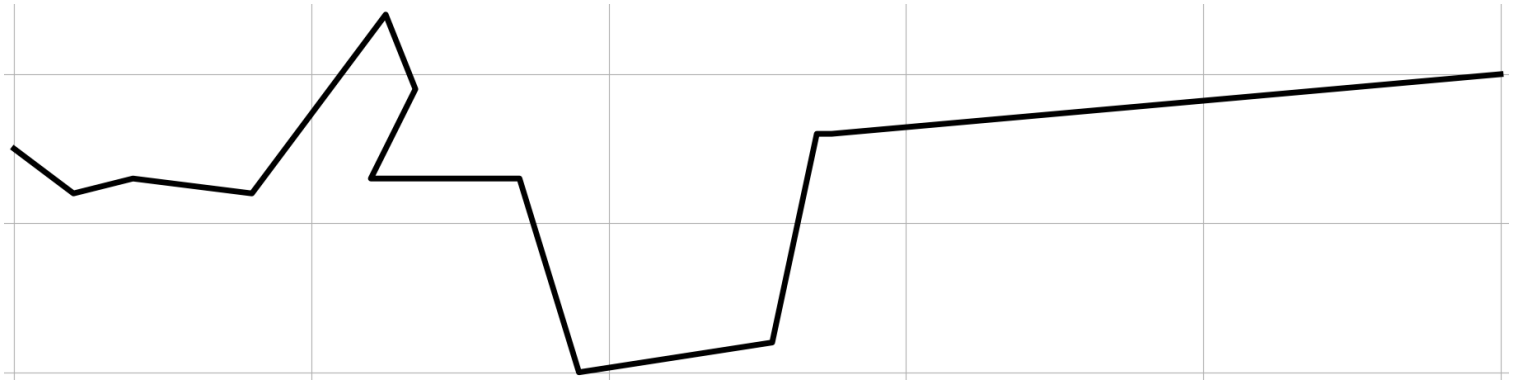
Definition

Sei $S \subseteq \mathbb{R}^d$ und $s > 0$. Eine Menge $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$ von Paaren von nicht-leeren Teilmengen von S ist genau dann eine *Zerlegung in wohl-separierte Paare*, wenn für alle $1 \leq i \leq m$ gilt:

1. $A_i \cap B_i = \emptyset$.
2. Für alle $p, q \in S$ gibt es genau einen Index $1 \leq j \leq m$, sodass entweder $p \in A_j$ und $q \in B_j$ oder $q \in A_j$ und $p \in B_j$.
3. A_i und B_i sind bezüglich s wohl-separiert.

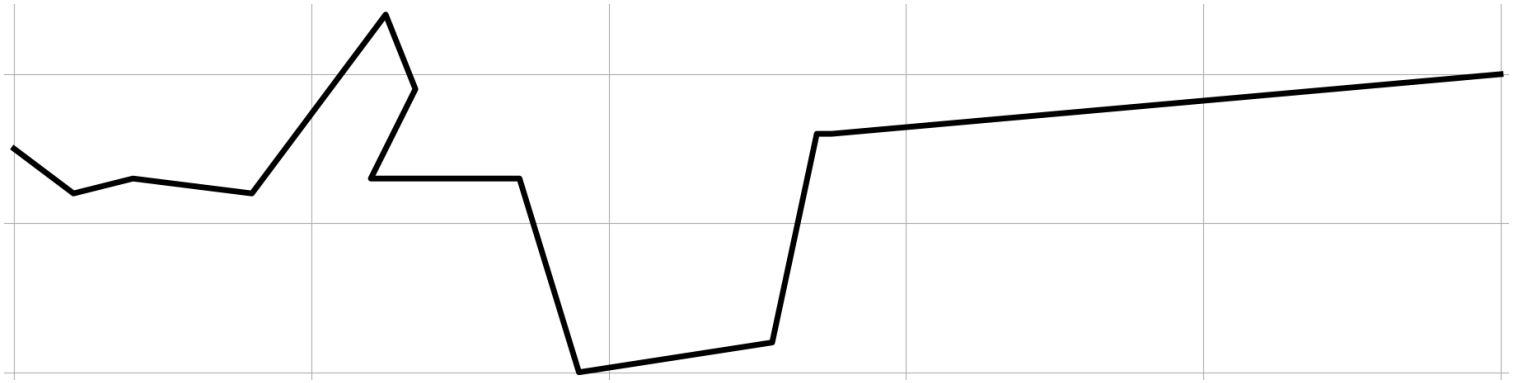
WSPD - Algorithmus für unsere Anwendung

- Transformation des Eingabekantenzuges $P = (p_1, p_2, \ldots, p_n)$ auf eine eindimensionale Folge $S = (x_1, x_2, \ldots, x_n)$, wobei $x_i = \delta(p_1, p_i)$



WSPD - Algorithmus für unsere Anwendung

- Transformation des Eingabekantenzuges $P = (p_1, p_2, \ldots, p_n)$ auf eine eindimensionale Folge $S = (x_1, x_2, \ldots, x_n)$, wobei $x_i = \delta(p_1, p_i)$



$$\Downarrow$$

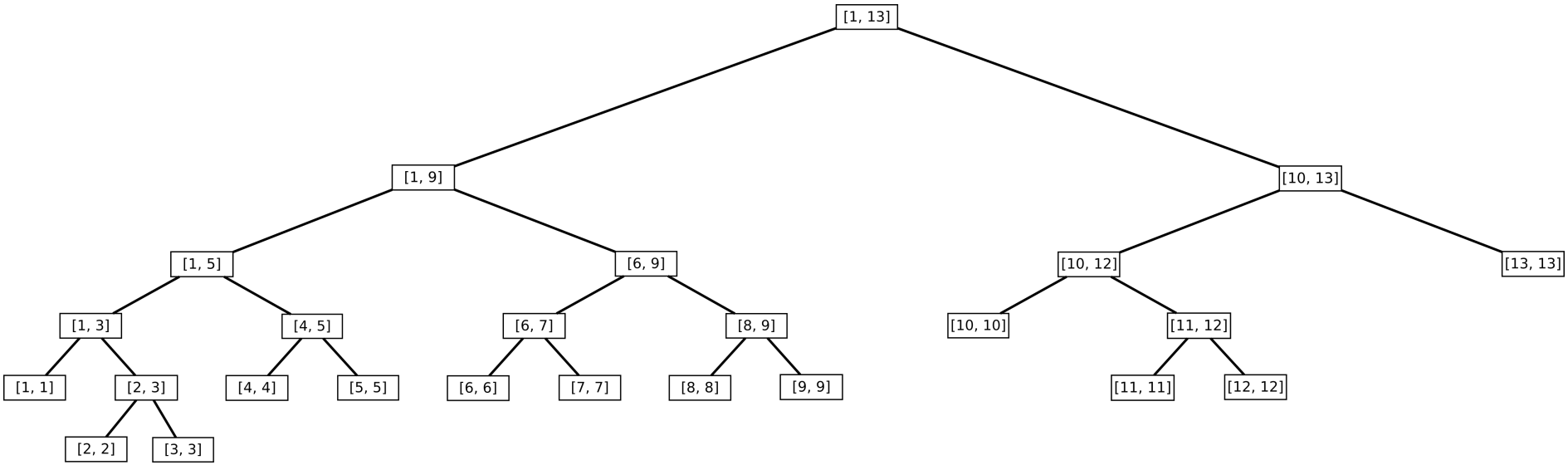
$$S = [0, 5.0, 9.1, 17.2, 32.2, 37.6, 44.3, 54.3, 67.9, 81.0, 95.4, 96.4, 141.5]$$

WSPD - Algorithmus für unsere Anwendung

- Berechnen eines *(fairen) Split-Trees* T aus S

$$S = [0, 5.0, 9.1, 17.2, 32.2, 37.6, 44.3, 54.3, 67.9, 81.0, 95.4, 96.4, 141.5]$$

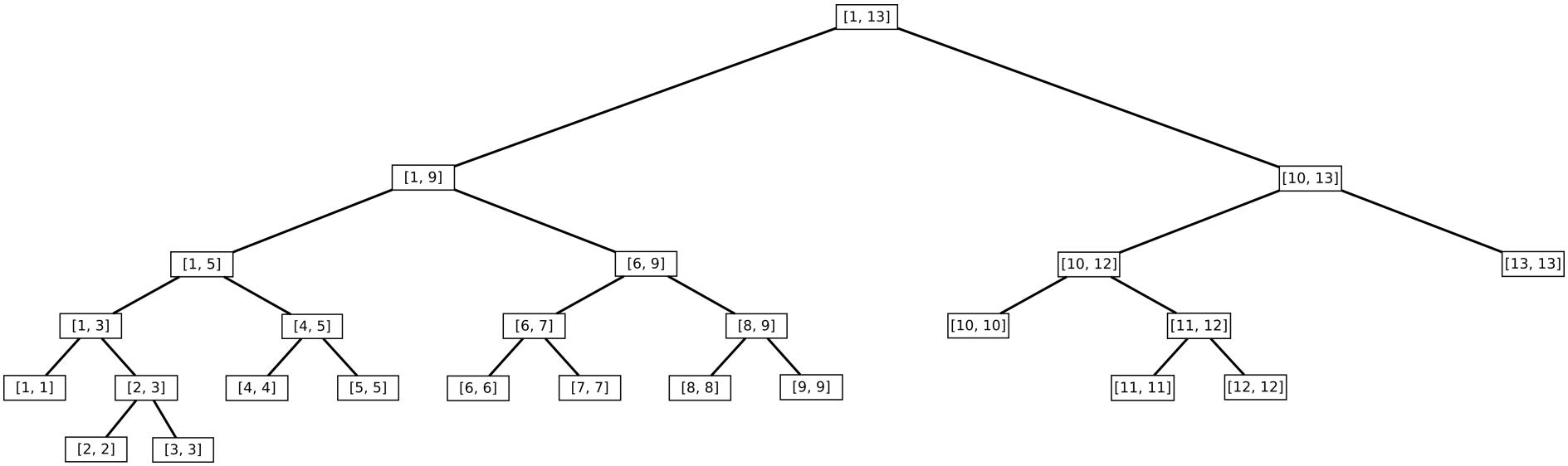
$$\Downarrow$$



WSPD - Algorithmus für unsere Anwendung

$S = [0, 5.0, 9.1, 17.2, 32.2, 37.6, 44.3, 54.3, 67.9, 81.0, 95.4, 96.4, 141.5]$

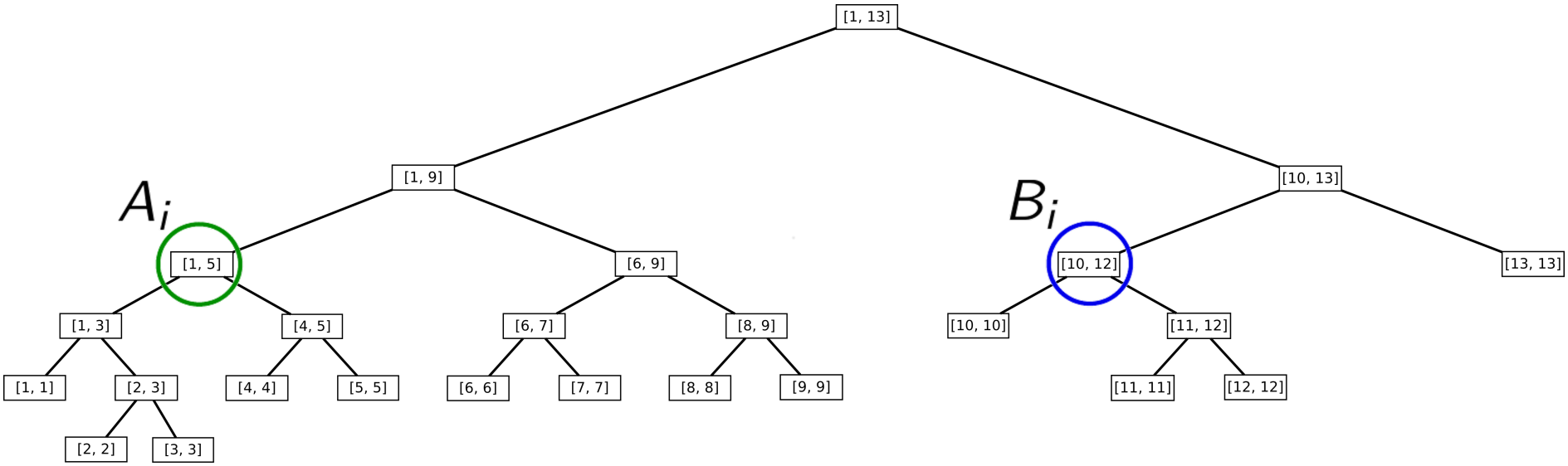
- Berechnen einer WSPD aus dem Split-Tree T



WSPD - Algorithmus für unsere Anwendung

$S = [0, 5.0, 9.1, 17.2, 32.2, 37.6, 44.3, 54.3, 67.9, 81.0, 95.4, 96.4, 141.5]$

- Berechnen einer WSPD aus dem Split-Tree T



Laufzeit

Satz (Callahan/Kosaraju)

Sei $S \subset \mathbb{R}$ endlich und $n := |S|$. Dann kann in $O(n \log n + sn)$ Zeit ein Split-Tree T und eine dazugehörige WSPD $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$ der Größe $m = O(sn)$ berechnet werden.

Sei $0 < \epsilon < \frac{1}{3}$ und $s = \frac{12+24(1+\frac{\epsilon}{3})\cdot t}{\epsilon}$.

Lemma

Seien $p, p', q, q' \in P$, sodass $\rho := \delta(p_1, p) \in A_i$, $\rho' := \delta(p_1, p') \in A_i$, $\varphi := \delta(p_1, q) \in B_i$ und $\varphi' := \delta(p_1, q') \in B_i$. Dann gilt:

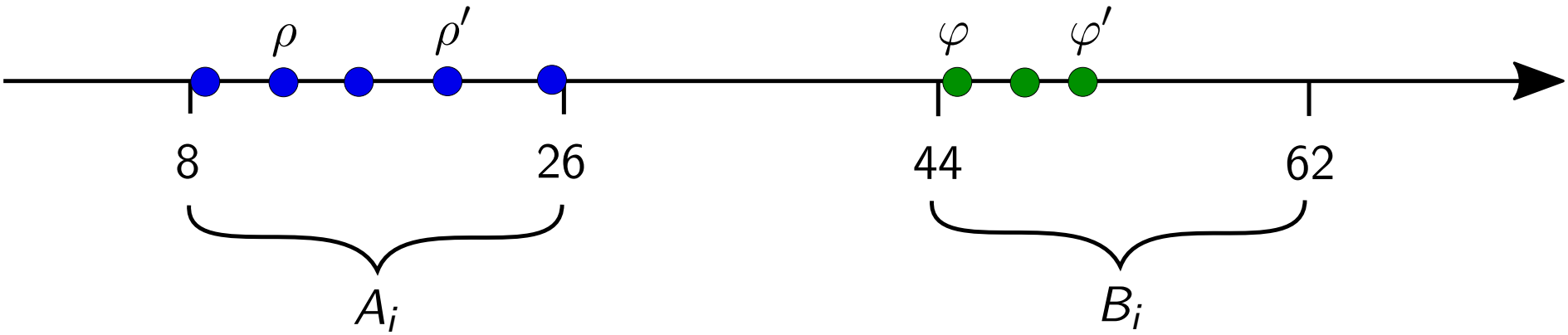
1. *(p, q) t-distanzerhaltend $\Rightarrow$ (p', q') $(1 + \frac{\epsilon}{3})t$-distanzerhaltend.*
2. *(p, q) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend $\Rightarrow$ (p', q') $(1 + \epsilon)t$-distanzerhaltend.*

Sei $0 < \epsilon < \frac{1}{3}$ und $s = \frac{12+24(1+\frac{\epsilon}{3})\cdot t}{\epsilon}$.

Lemma

Seien $p, p', q, q' \in P$, sodass $\rho := \delta(p_1, p) \in A_i$, $\rho' := \delta(p_1, p') \in A_i$, $\varphi := \delta(p_1, q) \in B_i$ und $\varphi' := \delta(p_1, q') \in B_i$. Dann gilt:

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



Konstruktion eines Graphen H

- $S = \{x_1, x_2, \ldots, x_n\}$ mit $x_i = \delta(p_1, p_n)$
- WSPD $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$

- Wahl von festen Elementen $a_i \in A_i$ und $b_i \in B_i$.
- α_i und β_i so, dass $a_i = \delta(p_1, \alpha_i)$ und $b_i = \delta(p_1, \beta_i)$.

Konstruktion eines Graphen H

- $S = \{x_1, x_2, \ldots, x_n\}$ mit $x_i = \delta(p_1, p_n)$
- WSPD $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$

- Wahl von festen Elementen $a_i \in A_i$ und $b_i \in B_i$.
- α_i und β_i so, dass $a_i = \delta(p_1, \alpha_i)$ und $b_i = \delta(p_1, \beta_i)$.
- (A_i, B_i) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend $\Leftrightarrow$ (α_i, β_i) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend

Konstruktion eines Graphen H

- $S = \{x_1, x_2, \ldots, x_n\}$ mit $x_i = \delta(p_1, p_n)$
- WSPD $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$

- Wahl von festen Elementen $a_i \in A_i$ und $b_i \in B_i$.
- α_i und β_i so, dass $a_i = \delta(p_1, \alpha_i)$ und $b_i = \delta(p_1, \beta_i)$.
- (A_i, B_i) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend $\Leftrightarrow$ (α_i, β_i) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend

Konstruktion eines Graphen $H = (V, E)$:

- $V := \{A_i \mid 1 \leq i \leq m\} \cup \{B_i \mid 1 \leq i \leq m\}$

Konstruktion eines Graphen H

- $S = \{x_1, x_2, \ldots, x_n\}$ mit $x_i = \delta(p_1, p_n)$
- WSPD $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$

- Wahl von festen Elementen $a_i \in A_i$ und $b_i \in B_i$.
- α_i und β_i so, dass $a_i = \delta(p_1, \alpha_i)$ und $b_i = \delta(p_1, \beta_i)$.
- (A_i, B_i) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend $\Leftrightarrow$ (α_i, β_i) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend

Konstruktion eines Graphen $H = (V, E)$:

- $V := \{A_i \mid 1 \leq i \leq m\} \cup \{B_i \mid 1 \leq i \leq m\}$
- Kanten E:
  1. Für alle $1 \leq i \leq m$ ist (A_i, B_i) genau dann eine Kante, wenn (A_i, B_i) $(1 + \frac{\epsilon}{3})t$-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 Kante, wenn (A_i, B_i) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend ist und $A_j \cap B_i \neq \emptyset$.

Approximation $\rightarrow H$

$S = \{x_1, x_2, \ldots, x_n\}$ mit $x_i = \delta(p_1, p_n)$

Satz

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. Siehe Aufsatz.

Ergebnis: Pfad $R = (A_{i_1}, A_{i_2}, \ldots, A_{i_{k-1}}, B_{i_{k-1}})$ mit $\delta(p_1, q_j) \in B_{i_{j-1}} \cap A_{i_j}$ für $1 < j < k$

$H \rightarrow$ Approximation

Satz

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. Siehe Aufsatz.

Ergebnis: $(1+\epsilon)t$-distanzerhaltender Kantenzug $Q = (q_1, q_2, \ldots, q_k)$ mit $q_1 = p_1$ und $q_k = p_n$.

Algorithmus

Schritt 1: Berechnen von $S = (x_1, \ldots, x_n)$ mit $x_i = \delta(p_1, p_i)$

Algorithmus

Schritt 1: Berechnen von $S = (x_1, \ldots, x_n)$ mit $x_i = \delta(p_1, p_i)$

Schritt 2: Berechnen des Split-Trees T und einer WSPD $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$ mit der Trennungsrate $s = \frac{12+24(1+\frac{\epsilon}{3})t}{\epsilon}$.

Algorithmus

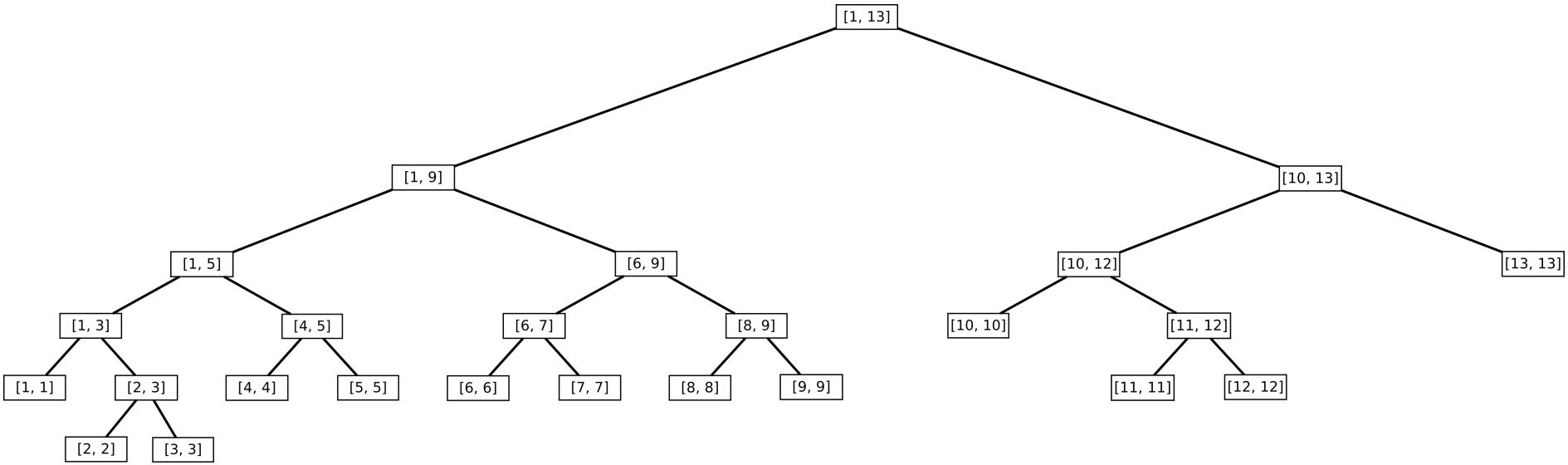
Schritt 1: Berechnen von $S = (x_1, \ldots, x_n)$ mit $x_i = \delta(p_1, p_i)$

Schritt 2: Berechnen des Split-Trees T und einer WSPD $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$ mit der Trennungsrate $s = \frac{12+24(1+\frac{\epsilon}{3})t}{\epsilon}$.

Schritt 3: Wählen von $a_i \in A_i$, $b_i \in B_i$, α_i und β_i die Knoten von P sind, für die $a_i = \delta(p_1, \alpha_i)$ und $b_i = \delta(p_1, \beta_i)$. Falls (α_i, β_i) nicht $(1 + \frac{\epsilon}{3})t$-distanzerhaltend ist, verwirf das korrespondierende Tupel (A_i, B_i), ansonsten behalte es.

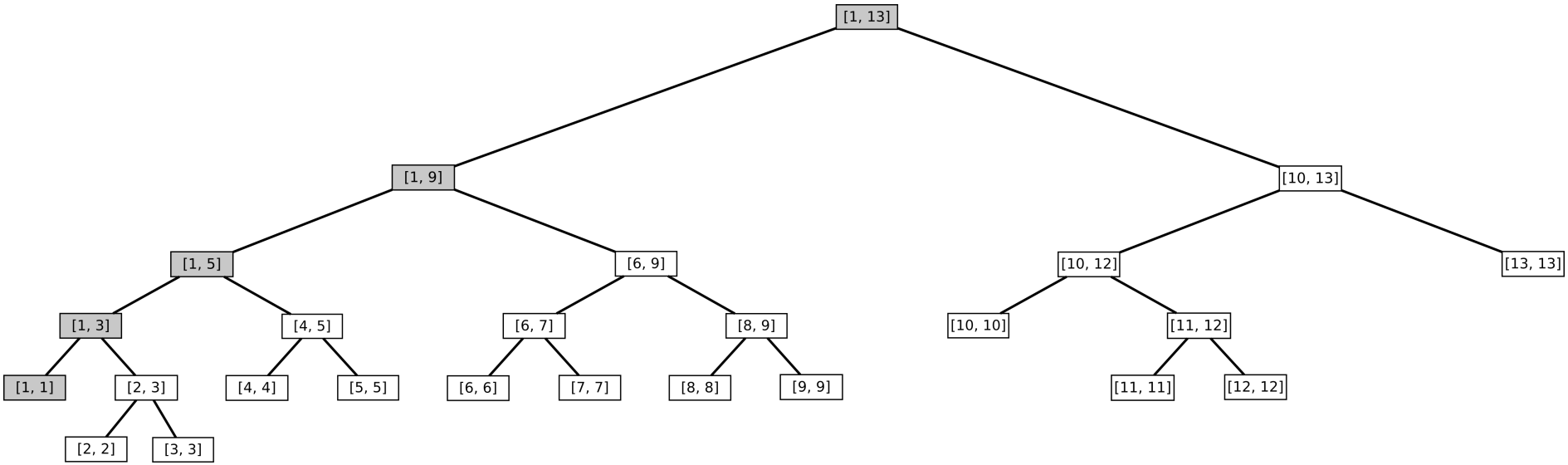
Modifizierte Breitensuche

Sei k Knoten im Baum.
$d[k]$: Distanz von k zum Startknoten



Modifizierte Breitensuche

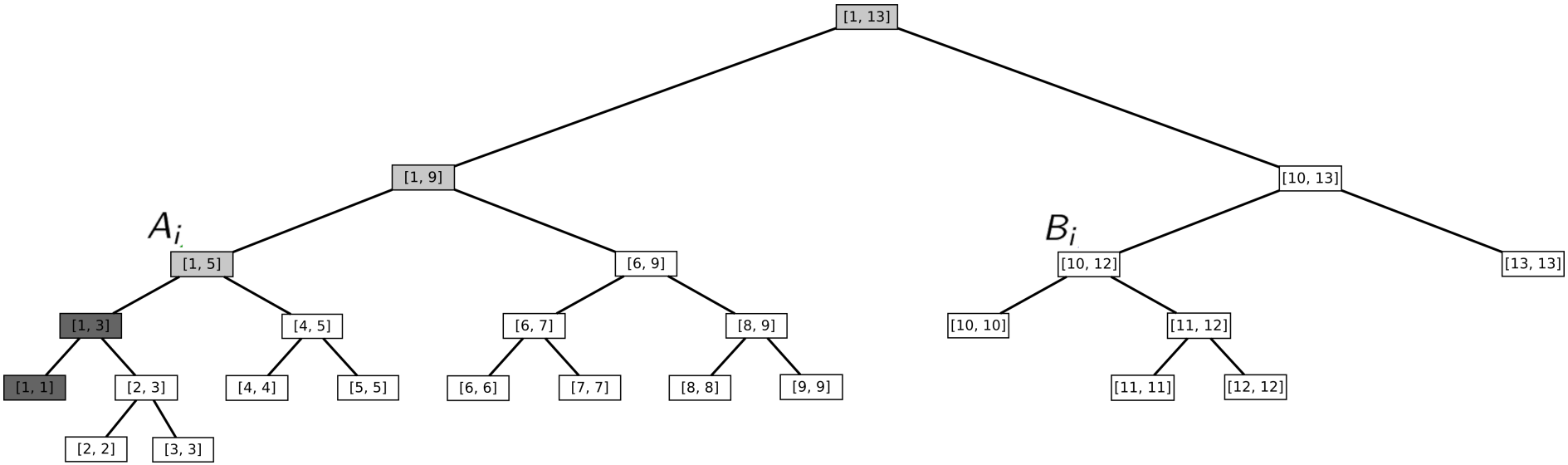
Sei k Knoten im Baum.
$d[k]$: Distanz von k zum Startknoten



Falls A-Knoten k, $d[k] = 0$ und füge k zur Warteschlange hinzu

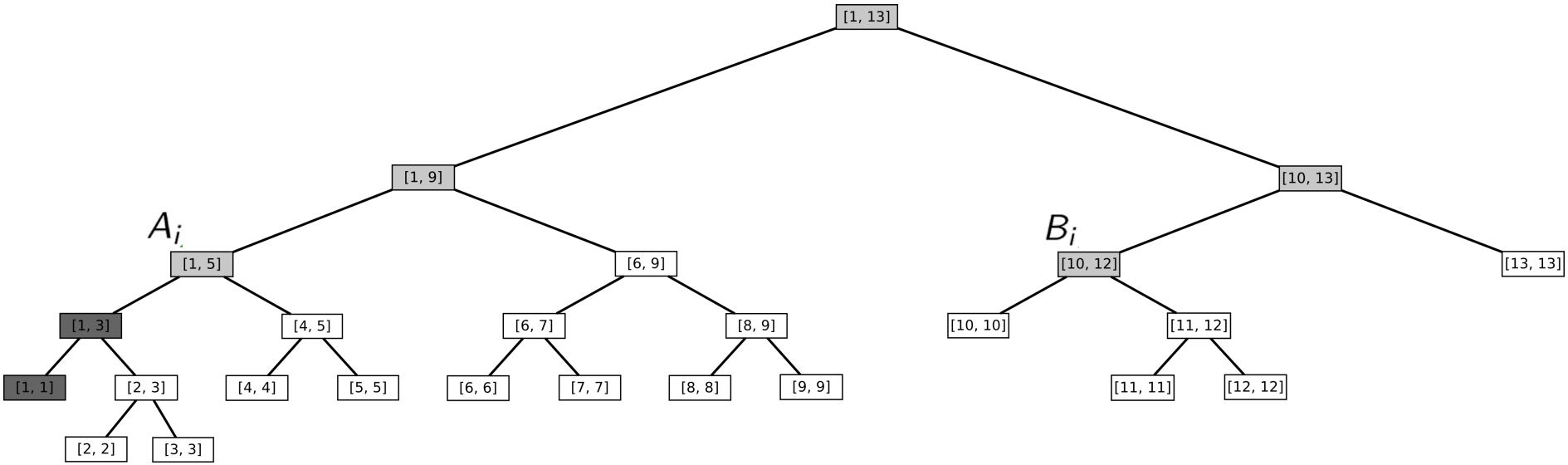
Modifizierte Breitensuche

Sei k Knoten im Baum.
$d[k]$: Distanz von k zum Startknoten



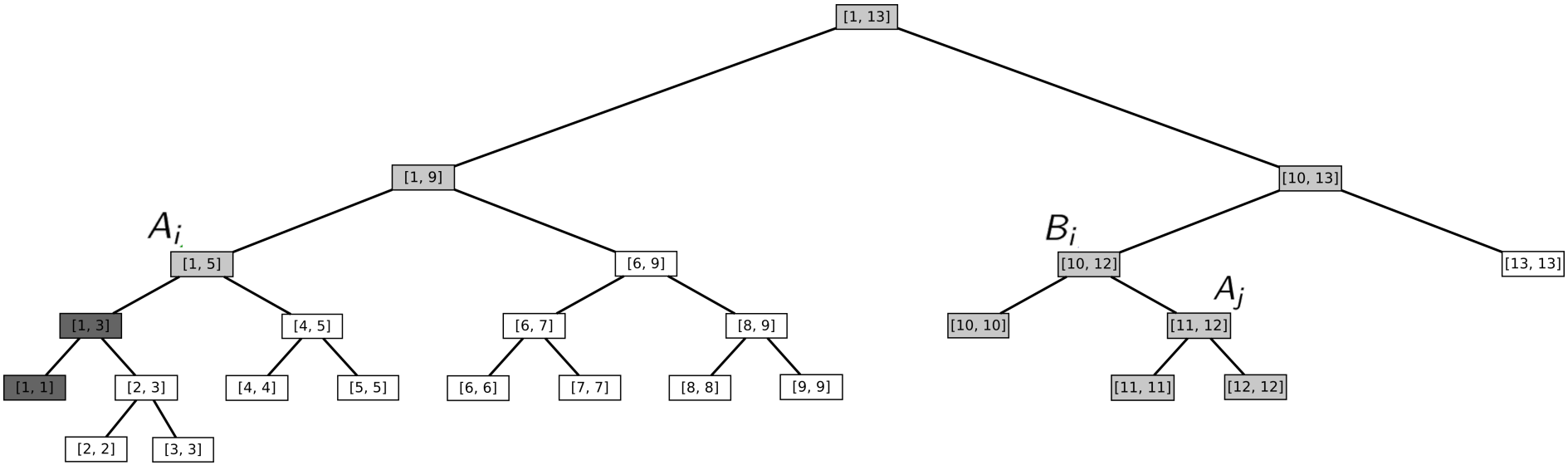
Modifizierte Breitensuche

Sei k Knoten im Baum.
$d[k]$: Distanz von k zum Startknoten



Modifizierte Breitensuche

Sei k Knoten im Baum.
$d[k]$: Distanz von k zum Startknoten

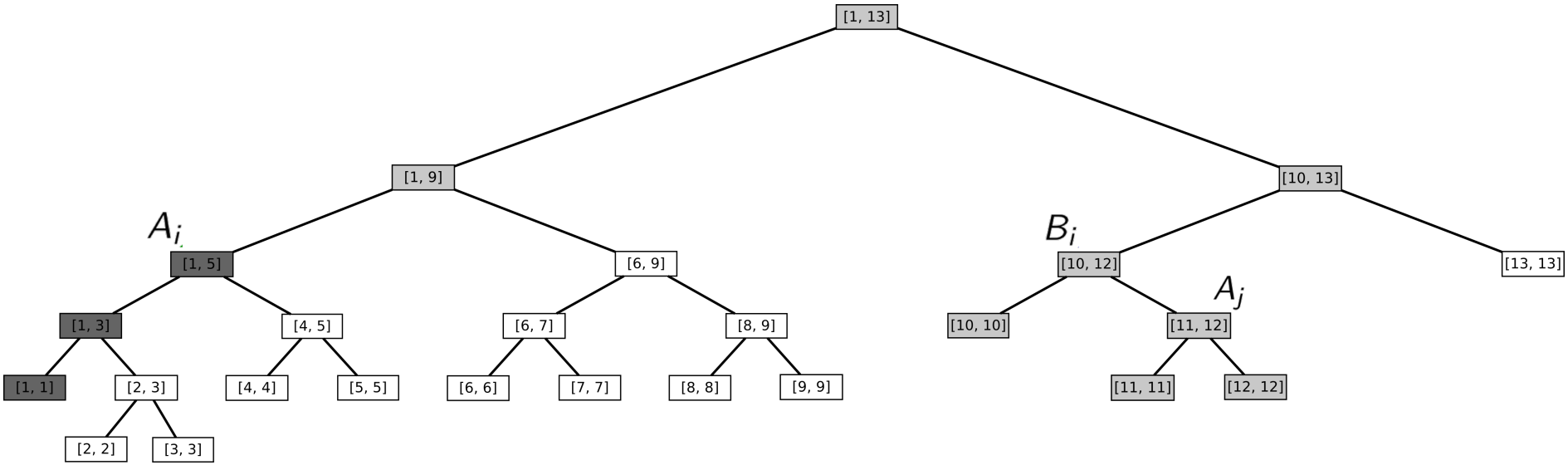


$$d[k'] = d[k] + 1$$

Füge k' zur Warteschlange hinzu

Modifizierte Breitensuche

Sei k Knoten im Baum.
$d[k]$: Distanz von k zum Startknoten



Algorithmus

Schritt 1: Berechnen von $S = (x_1, \ldots, x_n)$ mit $x_i = \delta(p_1, p_i)$

Schritt 2: Berechnen des Split-Trees T und einer WSPD $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$ mit der Trennungsrate $s = \frac{12+24(1+\frac{\epsilon}{3})t}{\epsilon}$.

Schritt 3: Wählen von $a_i \in A_i$, $b_i \in B_i$, α_i und β_i die Knoten von P sind, für die $a_i = \delta(p_1, \alpha_i)$ und $b_i = \delta(p_1, \beta_i)$. Falls (α_i, β_i) nicht $(1 + \frac{\epsilon}{3})t$-distanzerhaltend ist, verwirf das korrespondierende Tupel (A_i, B_i), ansonsten behalte es.

Algorithmus

Schritt 1: Berechnen von $S = (x_1, \ldots, x_n)$ mit $x_i = \delta(p_1, p_i)$

Schritt 2: Berechnen des Split-Trees T und einer WSPD $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$ mit der Trennungsrate $s = \frac{12+24(1+\frac{\epsilon}{3})t}{\epsilon}$.

Schritt 3: Wählen von $a_i \in A_i$, $b_i \in B_i$, α_i und β_i die Knoten von P sind, für die $a_i = \delta(p_1, \alpha_i)$ und $b_i = \delta(p_1, \beta_i)$. Falls (α_i, β_i) nicht $(1 + \frac{\epsilon}{3})t$-distanzerhaltend ist, verwirf das korrespondierende Tupel (A_i, B_i), ansonsten behalte es.

Schritt 4: Ausführen der modifizierte Breitensuche im Split-Tree T.

Algorithmus

Schritt 1: Berechnen von $S = (x_1, \ldots, x_n)$ mit $x_i = \delta(p_1, p_i)$

Schritt 2: Berechnen des Split-Trees T und einer WSPD $\{(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)\}$ mit der Trennungsrate $s = \frac{12+24(1+\frac{\epsilon}{3})t}{\epsilon}$.

Schritt 3: Wählen von $a_i \in A_i$, $b_i \in B_i$, α_i und β_i die Knoten von P sind, für die $a_i = \delta(p_1, \alpha_i)$ und $b_i = \delta(p_1, \beta_i)$. Falls (α_i, β_i) nicht $(1 + \frac{\epsilon}{3})t$-distanzerhaltend ist, verwirf das korrespondierende Tupel (A_i, B_i), ansonsten behalte es.

Schritt 4: Ausführen der modifizierte Breitensuche im Split-Tree T.

Schritt 5: Umwandeln des erhaltenen Pfades $(A_{i_1}, \ldots, A_{i_{k-1}}, B_{i_{k-1}})$ zu einer Approximation von P.

Laufzeit

$$s = \frac{12+24(1+\frac{\epsilon}{3})t}{\epsilon} \Rightarrow O(sn) = O(\frac{t}{\epsilon}n)$$

Schritt 1: Berechnen von S $O(n)$

Laufzeit

$$s = \frac{12+24(1+\frac{\epsilon}{3})t}{\epsilon} \Rightarrow O(sn) = O(\frac{t}{\epsilon}n)$$

Schritt 1:	Berechnen von S	$O(n)$
Schritt 2:	Berechnen der WSPD	$O(n \log n + \frac{t}{\epsilon}n)$

Laufzeit

$$s = \frac{12+24(1+\frac{\epsilon}{3})t}{\epsilon} \Rightarrow O(sn) = O(\frac{t}{\epsilon}n)$$

Schritt 1:	Berechnen von S	$O(n)$
Schritt 2:	Berechnen der WSPD	$O(n \log n + \frac{t}{\epsilon}n)$
Schritt 3:	Aussortieren der WSPD	$O(\frac{t}{\epsilon}n)$

Laufzeit

$$s = \frac{12+24(1+\frac{\epsilon}{3})t}{\epsilon} \Rightarrow O(sn) = O(\frac{t}{\epsilon}n)$$

Schritt 1:	Berechnen von S	$O(n)$
Schritt 2:	Berechnen der WSPD	$O(n \log n + \frac{t}{\epsilon}n)$
Schritt 3:	Aussortieren der WSPD	$O(\frac{t}{\epsilon}n)$
Schritt 4:	Modifizierte Breitensuche	$O(\frac{t}{\epsilon}n)$

Laufzeit

$$s = \frac{12+24(1+\frac{\epsilon}{3})t}{\epsilon} \Rightarrow O(sn) = O(\frac{t}{\epsilon}n)$$

Schritt 1:	Berechnen von S	$O(n)$
Schritt 2:	Berechnen der WSPD	$O(n \log n + \frac{t}{\epsilon}n)$
Schritt 3:	Aussortieren der WSPD	$O(\frac{t}{\epsilon}n)$
Schritt 4:	Modifizierte Breitensuche	$O(\frac{t}{\epsilon}n)$
Schritt 5:	Umwandeln in Approximation	$O(\frac{t}{\epsilon}n)$

Satz

Sei $P = (p_1, p_2, \ldots, p_n)$ ein Kantenzug in $\mathbb{R}^d$, sei $t \geq 1$ und $0 < \epsilon < \frac{1}{3}$ und sei κ die Knotenzahl der minimalen t-distanzerhaltenden Approximationen von P. Dann können wir in $O(n \log n + \frac{t}{\epsilon} n)$ eine $(1+\epsilon)t$-distanzerhaltende Approximation Q von P mit maximal κ Knoten berechnen.

$H \rightarrow$ Approximation

Satz

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 y_i das Element der Menge S, für das $y_i = \delta(p_1, q_i)$ gilt.

- $q_1 := p_1$

Wiederholung

- $y_i = \delta(p_1, q_i)$
- Für alle $1 \leq i < j \leq m$ ist (A_i, A_j) genau dann eine Kante, wenn (A_i, B_i) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend ist und $A_j \cap B_i \neq \emptyset$.

Wiederholung

- $y_i = \delta(p_1, q_i)$
- Für alle $1 \leq i < j \leq m$ ist (A_i, A_j) genau dann eine Kante, wenn (A_i, B_i) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend ist und $A_j \cap B_i \neq \emptyset$.

- Annahme: für ein l mit $1 \leq l < k-1$ wurde der Teilpfad $(A_{i_1}, \ldots, A_{i_l})$ bereits in den Kantenzug $(q_1, \ldots, q_l)$ umgewandelt, sodass für alle $1 < j \leq l$ $y_j \in A_{i_j} \cap B_{i_{j-1}}$. Wir betrachten die Kante $(A_{i_l}, A_{i_{l+1}})$.
  - Es gibt ein $y \in A_{i_{l+1}} \cap B_{i_l}$.
  - q_{l+1} ist der Knoten γ von P, für den $y = \delta(p_1, \gamma)$ gilt

Wiederholung

- $y_i = \delta(p_1, q_i)$
- Für alle $1 \leq i < j \leq m$ ist (A_i, A_j) genau dann eine Kante, wenn (A_i, B_i) $(1 + \frac{\epsilon}{3})t$-distanzerhaltend ist und $A_j \cap B_i \neq \emptyset$.

- Annahme: für ein l mit $1 \leq l < k-1$ wurde der Teilpfad $(A_{i_1}, \ldots, A_{i_l})$ bereits in den Kantenzug $(q_1, \ldots, q_l)$ umgewandelt, sodass für alle $1 < j \leq l$ $y_j \in A_{i_j} \cap B_{i_{j-1}}$. Wir betrachten die Kante $(A_{i_l}, A_{i_{l+1}})$.
  - Es gibt ein $y \in A_{i_{l+1}} \cap B_{i_l}$.
  - q_{l+1} ist der Knoten γ von P, für den $y = \delta(p_1, \gamma)$ gilt
- Annahme: $(A_{i_1}, \ldots, A_{i_{k-1}})$ wurde bereits zu $(q_1, \ldots, q_{k-1})$ umgewandelt haben. Nach Voraussetzung ist $x_n \in B_{i_{k-1}}$.
  $\Rightarrow q_k := p_n$.

Bereits gezeigt: Umwandlung in Kantenzug $Q = (q_1, q_2, \ldots, q_k)$.

Noch zu zeigen: Q ist $(1 + \epsilon)t$-distanzerhaltend.

Sei $1 \leq j < k$.

Bereits gezeigt: Umwandlung in Kantenzug $Q = (q_1, q_2, \ldots, q_k)$.

Noch zu zeigen: Q ist $(1 + \epsilon)t$-distanzerhaltend.

Sei $1 \leq j < k$.

- (q_j, q_{j+1}) ist durch Umwandlung aus $(A_{i_j}, A_{i_{j+1}})$ entstanden, wobei $\delta(p_1, q_i) \in A_{i_j}$ und $\delta(p_1, q_{i+1}) \in B_{i_j}$ $(\cap\, A_{i_{j+1}})$.

Bereits gezeigt: Umwandlung in Kantenzug $Q = (q_1, q_2, \ldots, q_k)$.

Noch zu zeigen: Q ist $(1 + \epsilon)t$-distanzerhaltend.

Sei $1 \leq j < k$.

- (q_j, q_{j+1}) ist durch Umwandlung aus $(A_{i_j}, A_{i_{j+1}})$ entstanden, wobei $\delta(p_1, q_i) \in A_{i_j}$ und $\delta(p_1, q_{i+1}) \in B_{i_j}$ $(\cap A_{i_{j+1}})$.
- Nach Konstruktion von H: (A_{i_j}, B_{i_j}) ist $(1 + \frac{\epsilon}{3})t$-distanzerhaltend.

Bereits gezeigt: Umwandlung in Kantenzug $Q = (q_1, q_2, \ldots, q_k)$.

Noch zu zeigen: Q ist $(1+\epsilon)t$-distanzerhaltend.

Sei $1 \leq j < k$.

- (q_j, q_{j+1}) ist durch Umwandlung aus $(A_{i_j}, A_{i_{j+1}})$ entstanden, wobei $\delta(p_1, q_i) \in A_{i_j}$ und $\delta(p_1, q_{i+1}) \in B_{i_j}$ $(\cap\, A_{i_{j+1}})$.
- Nach Konstruktion von H: (A_{i_j}, B_{i_j}) ist $(1+\frac{\epsilon}{3})t$-distanzerhaltend.
- Also ist $(\alpha_{i_j}, \beta_{i_j})$ $(1+\frac{\epsilon}{3})t$-distanzerhaltend.

Bereits gezeigt: Umwandlung in Kantenzug $Q = (q_1, q_2, \ldots, q_k)$.

Noch zu zeigen: Q ist $(1+\epsilon)t$-distanzerhaltend.

Sei $1 \leq j < k$.

- (q_j, q_{j+1}) ist durch Umwandlung aus $(A_{i_j}, A_{i_{j+1}})$ entstanden, wobei $\delta(p_1, q_i) \in A_{i_j}$ und $\delta(p_1, q_{i+1}) \in B_{i_j}$ $(\cap A_{i_{j+1}})$.
- Nach Konstruktion von H: (A_{i_j}, B_{i_j}) ist $(1+\frac{\epsilon}{3})t$-distanzerhaltend.
- Also ist $(\alpha_{i_j}, \beta_{i_j})$ $(1+\frac{\epsilon}{3})t$-distanzerhaltend.
- $\overset{Lemma}{\Rightarrow}$ (q_j, q_{j+1}) ist $(1+\epsilon)t$-distanzerhaltend.