Methoden der Offline Bewegungsplanung

Trapezzerlegung, BA-Themen

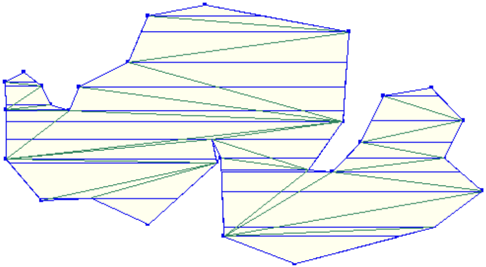
Elmar Langetepe

Universität Bonn, Institut für Informatik

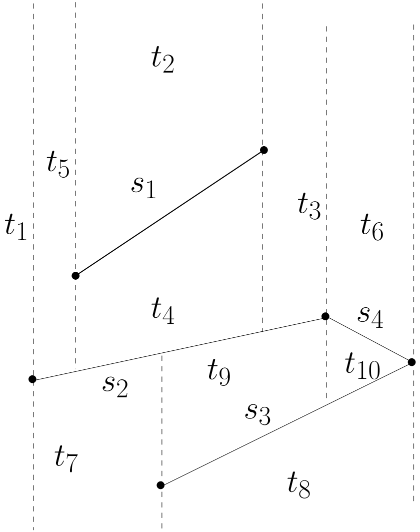
28.01.2015

Datenstruktur: Trapezzerlegung nach Seidel

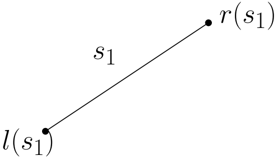
- Lokalisation Black Box
- Polygon: Dreieckszerlegung
- Anfrage: Für $p \in P$ finde Dreieck T mit $p \in T$
- Kürzeste Wege in Polygonen/auf Polyeder
- Aufbau: $O(n \log^* n)$, Query: $O(\log n)$
- Zerlegung in Trapeze, monotone Polygone, danach in Dreiecke
- Hier Trapezzerlegung!



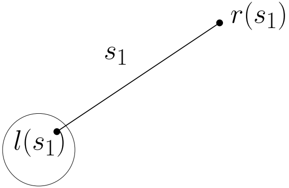
Trapezzerlegung nach Seidel



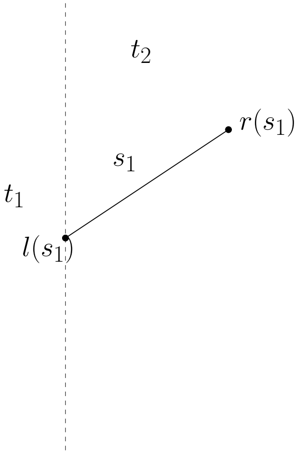
Datenstruktur: Trapezzerlegung nach Seidel

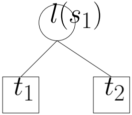


Datenstruktur: Trapezzerlegung nach Seidel

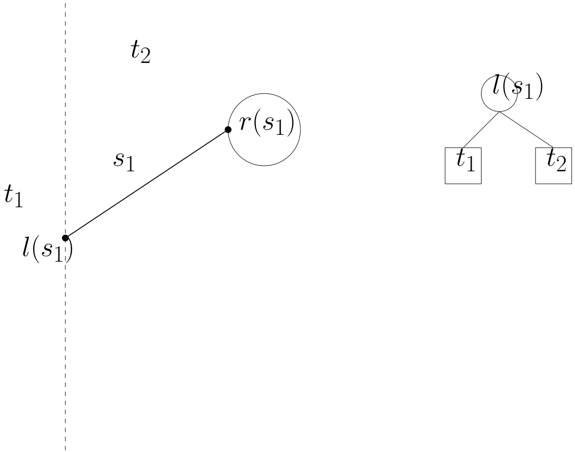


Datenstruktur: Trapezzerlegung nach Seidel

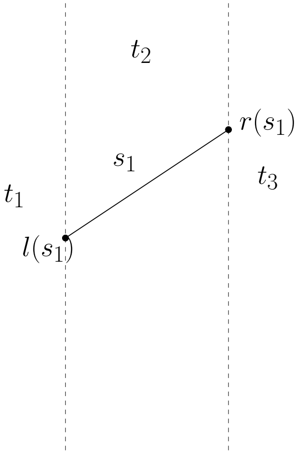


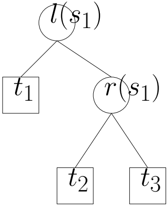


Datenstruktur: Trapezzerlegung nach Seidel

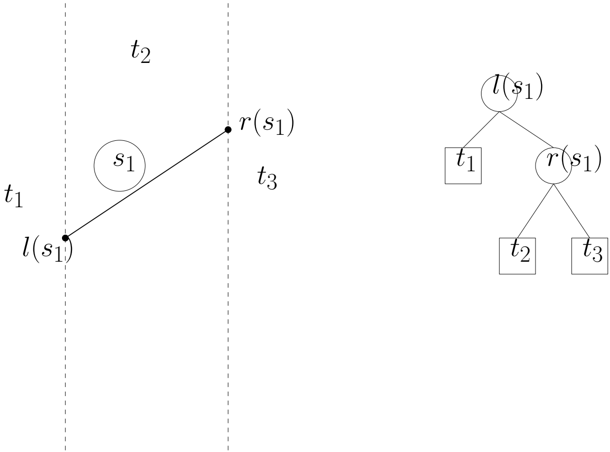


Datenstruktur: Trapezzerlegung nach Seidel

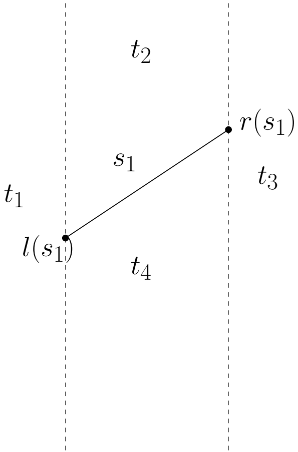


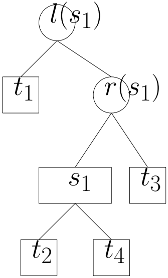


Datenstruktur: Trapezzerlegung nach Seidel

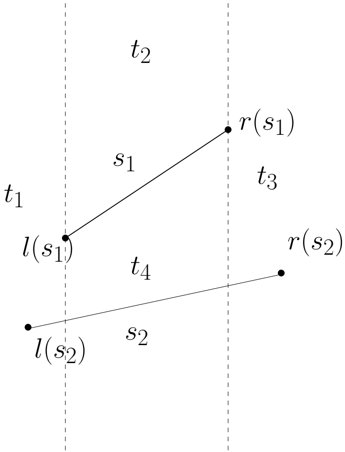


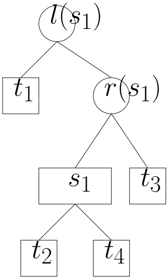
Datenstruktur: Trapezzerlegung nach Seidel



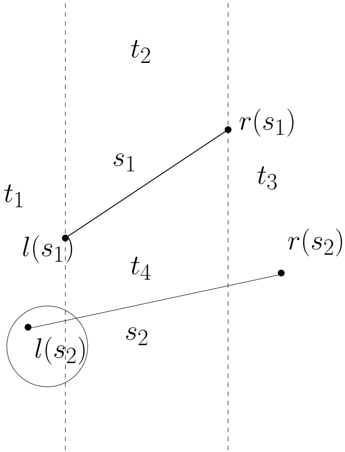


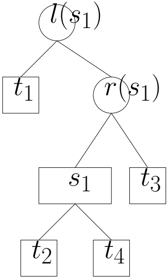
Datenstruktur: Trapezzerlegung nach Seidel



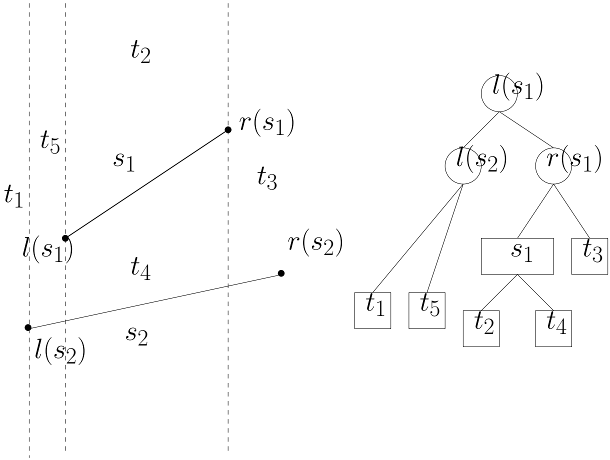


Datenstruktur: Trapezzerlegung nach Seidel

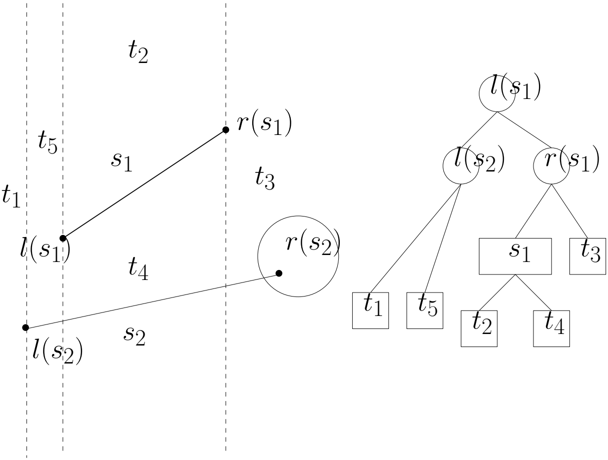




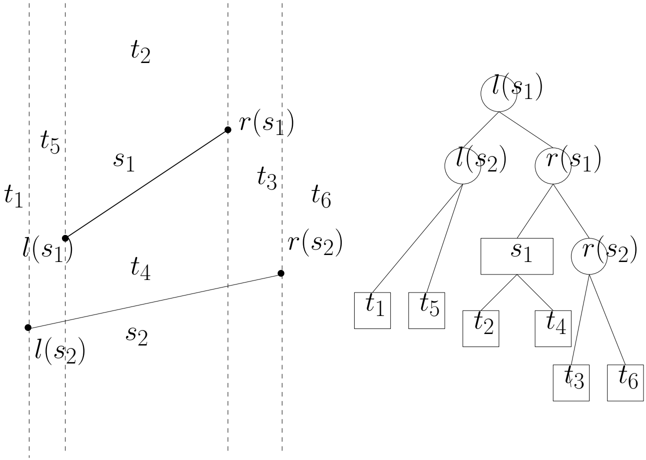
Datenstruktur: Trapezzerlegung nach Seidel



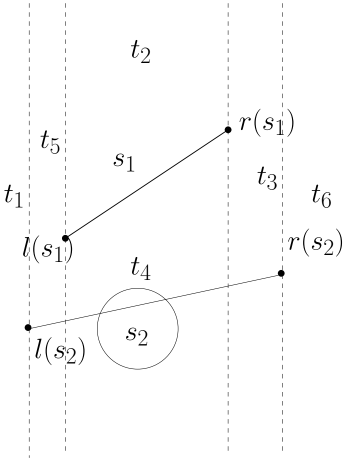
Datenstruktur: Trapezzerlegung nach Seidel

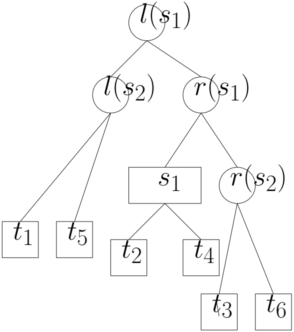


Datenstruktur: Trapezzerlegung nach Seidel

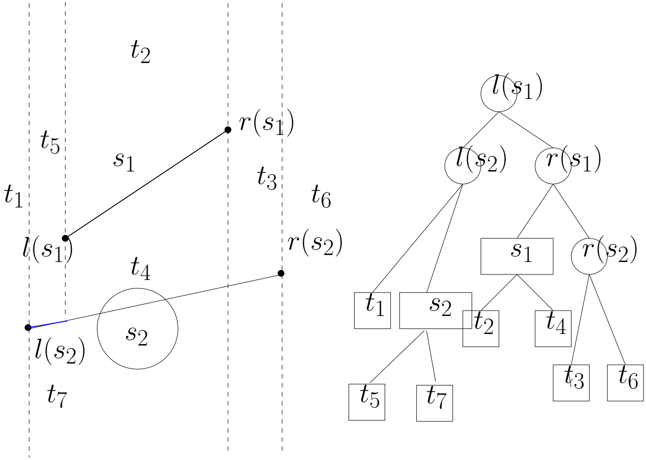


Datenstruktur: Trapezzerlegung nach Seidel

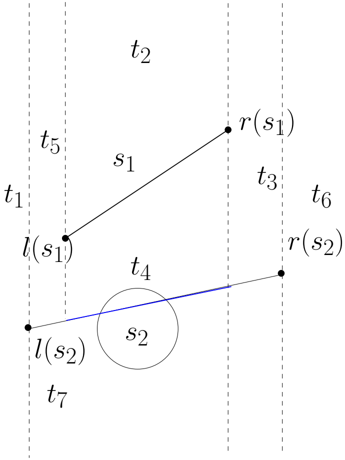


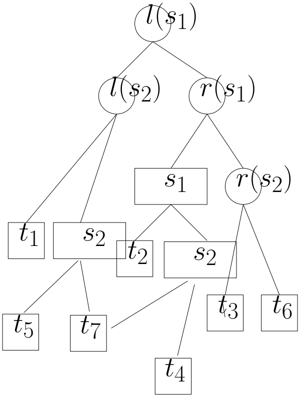


Datenstruktur: Trapezzerlegung nach Seidel

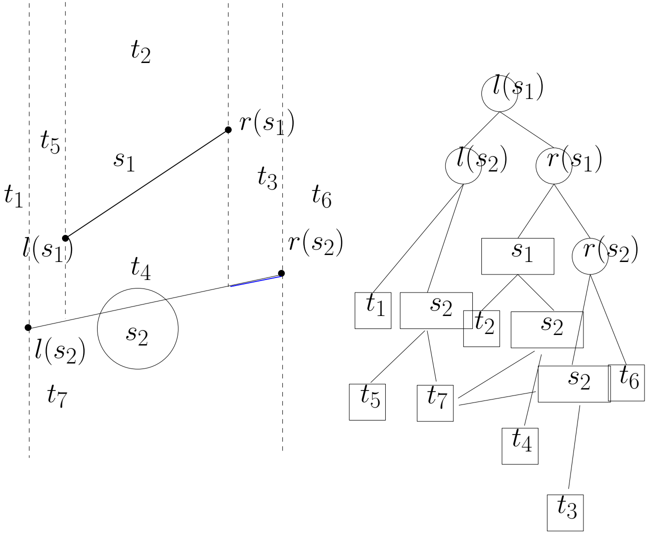


Datenstruktur: Trapezzerlegung nach Seidel

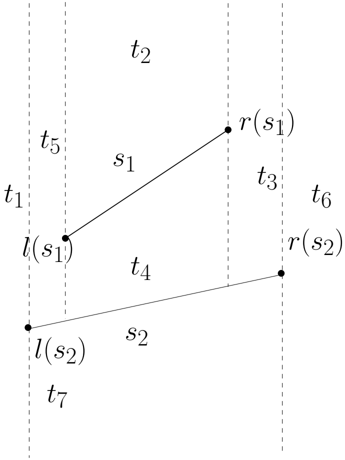


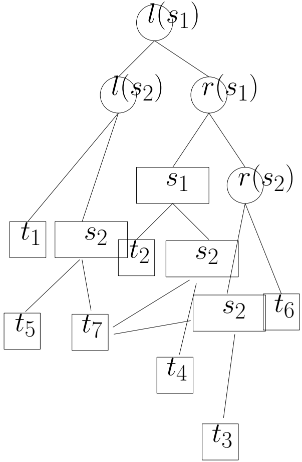


Datenstruktur: Trapezzerlegung nach Seidel



Datenstruktur: Trapezzerlegung nach Seidel





Datenstruktur: Trapezzerlegung nach Seidel

Vorgehensweise:

- Linken Endpunkt lokalisieren
- Rechten Endpunkt lokalisieren
- Strecke s_i durch Trapeze verfolgen
- Trapeze abspeichern mit Nachbarn und Links! $O(1)$
- Wesentlicher Aufwand: Lokalisation
- Query: Lokalisation eines Punktes
- Beweisskizze: Randomisierung! Einfache Analyse!

Datenstruktur: Trapezzerlegung nach Seidel

- Segmente $s_1, s_2, \dots, s_n$ zufällig!
- $P_i = \{s_1, \dots, s_i\}$
- $D(P_i)$ Datenstrucktur nach sukzessiven Einfügen
- $T(P_i)$ (eindeutige) Trapezzerlegung
- Lokalisation der Endpunkte
- Tracen der Segmente, insgesamt $O(n)$ wg. Komplexität

Lemma: Die erwartete Laufzeit bei zufälliger Einfügereihenfolge der Segmente zur Erstellung der Datenstruktur $D(P_n)$ liegt in $O(n \log n)$.

Beweis, Verbesserung durch Phasen, Tracing der polygonalen Kette!

Randomisierte Analyse: Lokalisation der Endpunkte

- Query Punkt q: Sei T_k Trapez von $T(P_k)$, das q enthält
- Betrachte Übergang von $D(P_{j-1})$ nach $D(P_j)$
- Annahme: q liegt in T_{j-1}
- Verändert sich T_{j-1} nicht, gilt $T_{j-1} = T_j$.
  Hier ist kein weiterer Vergleich an dieser Stelle nötig
- Gilt $T_{j-1} \neq T_j$, hat T_j am Rand eines der vier Bestandteile von s_j.
- Jedes Segment hatte die Wahrscheinlichkeit $\frac{4}{j}$, einen Bestandteil zu liefern
- Wahrscheinlichkeit eines weiteren Vergleichs liegt in $O\left(\frac{1}{j}\right)$
- Erwartete Anzahl Vergleiche: $C\sum_{i=1}^{n} \frac{1}{j} \in O(\log n)$

Datenstruktur: Trapezzerlegung nach Seidel

Idee: Schnellere Lokalisation: Polygonale Kette in Phasen in akt. DS lokalisieren.

Theorem: Die erwartete Laufzeit bei zufälliger Einfügereihenfolge der Segmente mit gelegentlicher Lokalisation der polygonalen Kette zur Erstellung der Datenstruktur $D(P_n)$ liegt in $O(n \log^* n)$. Die Zeit pro weiterer Anfrage liegt in $O(\log n)$.

Bewegungsplanung: Feuerbekämpfung

- Wichtiges Problem:
  Schlagen von Feuerschneisen
- Vorhandene Schneisen
  Neue Schneisen
- Einfaches Modell:
  Gleichmäßige Ausbreitung
- Geschwindigkeit: $s \in [0, 1)$

Bewegungsplanung: Feuerbekämpfung

- Wichtiges Problem:
  Schlagen von Feuerschneisen
- Vorhandene Schneisen
  Neue Schneisen
- Einfaches Modell:
  Gleichmäßige Ausbreitung
- Geschwindigkeit: $s \in [0, 1)$

Bewegungsplanung: Feuerbekämpfung

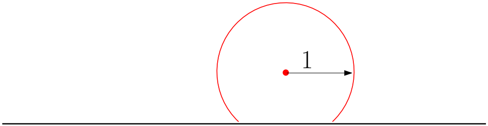
- Wichtiges Problem:
  Schlagen von Feuerschneisen
- Vorhandene Schneisen
  Neue Schneisen
- Einfaches Modell:
  Gleichmäßige Ausbreitung
- Geschwindigkeit: $s \in [0, 1)$

- $s \approx 0.38 \in [0, 1)$
- $s = \cos(\alpha)$,
  $\alpha \in (0, \pi/2]$

Bewegungsplanung: Feuerbekämpfung

- Wichtiges Problem:
  Schlagen von Feuerschneisen
- Vorhandene Schneisen
  Neue Schneisen
- Einfaches Modell:
  Gleichmäßige Ausbreitung
- Geschwindigkeit: $s \in [0,1)$

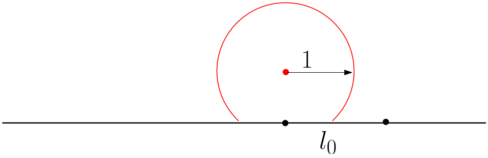
- $s \approx 0.38 \in [0,1)$
- $s = \cos(\alpha)$,
  $\alpha \in (0, \pi/2]$



Bewegungsplanung: Feuerbekämpfung

- Wichtiges Problem:
  Schlagen von Feuerschneisen
- Vorhandene Schneisen
  Neue Schneisen
- Einfaches Modell:
  Gleichmäßige Ausbreitung
- Geschwindigkeit: $s \in [0,1)$

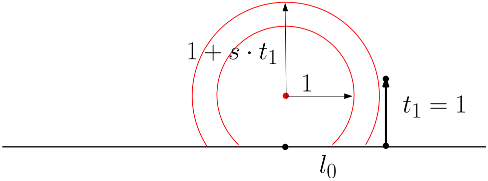
- $s \approx 0.38 \in [0,1)$
- $s = \cos(\alpha)$,
  $\alpha \in (0, \pi/2]$



Bewegungsplanung: Feuerbekämpfung

- Wichtiges Problem:
  Schlagen von Feuerschneisen
- Vorhandene Schneisen
  Neue Schneisen
- Einfaches Modell:
  Gleichmäßige Ausbreitung
- Geschwindigkeit: $s \in [0, 1)$

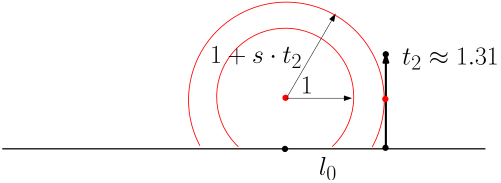
- $s \approx 0.38 \in [0, 1)$
- $s = \cos(\alpha)$,
  $\alpha \in (0, \pi/2]$



Bewegungsplanung: Feuerbekämpfung

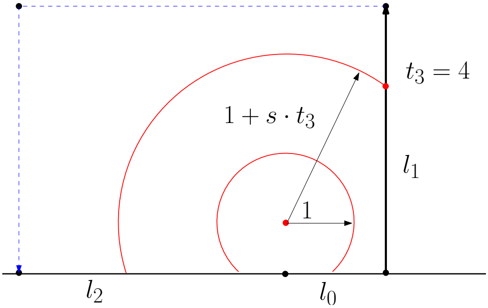
- Wichtiges Problem:
  Schlagen von Feuerschneisen
- Vorhandene Schneisen
  Neue Schneisen
- Einfaches Modell:
  Gleichmäßige Ausbreitung
- Geschwindigkeit: $s \in [0, 1)$

- $s \approx 0.38 \in [0, 1)$
- $s = \cos(\alpha)$,
  $\alpha \in (0, \pi/2]$



Bewegungsplanung: Feuerbekämpfung

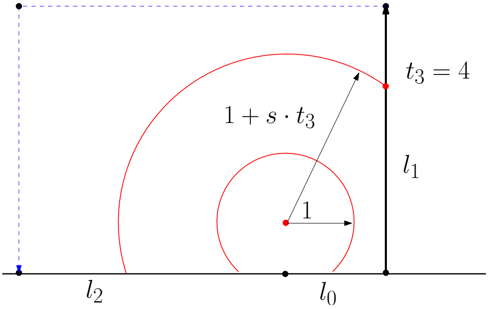
- Wichtiges Problem:
  Schlagen von Feuerschneisen
- Vorhandene Schneisen
  Neue Schneisen
- Einfaches Modell:
  Gleichmäßige Ausbreitung
- Geschwindigkeit: $s \in [0, 1)$



- $s \approx 0.38 \in [0, 1)$
- $s = \cos(\alpha)$,
  $\alpha \in (0, \pi/2]$

Bewegungsplanung: Feuerbekämpfung

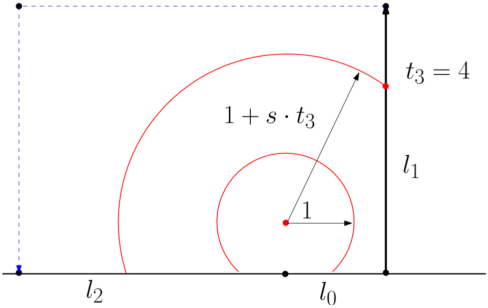
- Wichtiges Problem:
  Schlagen von Feuerschneisen
- Vorhandene Schneisen
  Neue Schneisen
- Einfaches Modell:
  Gleichmäßige Ausbreitung
- Geschwindigkeit: $s \in [0, 1)$



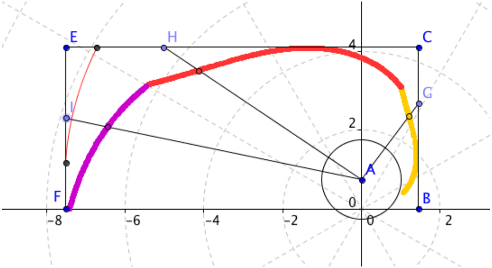
- $s \approx 0.38 \in [0, 1)$
- $s = \cos(\alpha)$,
  $\alpha \in (0, \pi/2]$
- Feuer schnell
  umschließen
  (Zeitminimierung!)

Bewegungsplanung: Feuerbekämpfung

- Wichtiges Problem:
  Schlagen von Feuerschneisen
- Vorhandene Schneisen
  Neue Schneisen
- Einfaches Modell:
  Gleichmäßige Ausbreitung
- Geschwindigkeit: $s \in [0,1)$



- $s \approx 0.38 \in [0,1)$
- $s = \cos(\alpha)$,
  $\alpha \in (0, \pi/2]$
- Feuer schnell
  umschließen
  (Zeitminimierung!)
- l_0, l_1, l_2 *gefahrlos*
  minimieren!

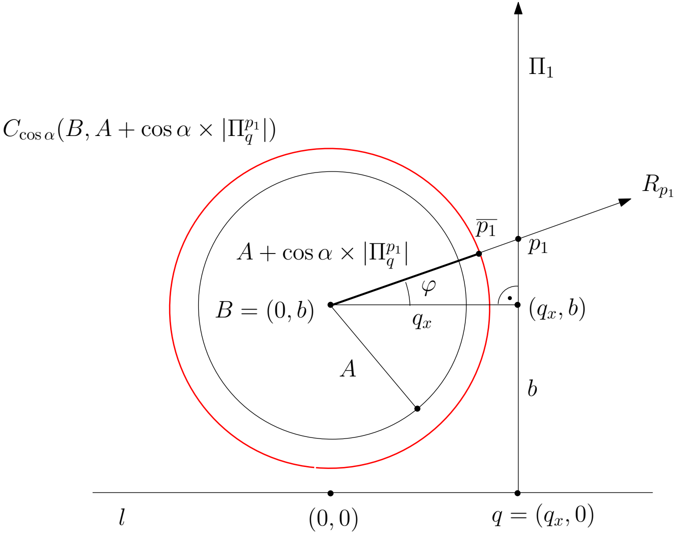


- $s \approx 0.38 = \cos(1.18)$
  Optimale Lösung!
- Minimieren der Werte
- *Passieren* des Feuers
- Zeit *und* Fläche!

Theorem: There exists a unique area and completion time optimal axis-parallel firebreak for all $\alpha \in (\pi/4, \pi/2]$. For $\alpha \in (0, \pi/4]$ no solution exists.

Beweis!

Situation erster Teil!

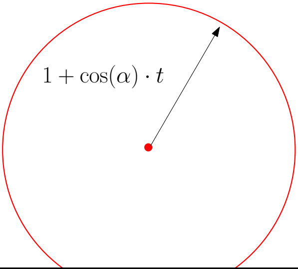


Bewegungsplanung: Feuerbekämpfung

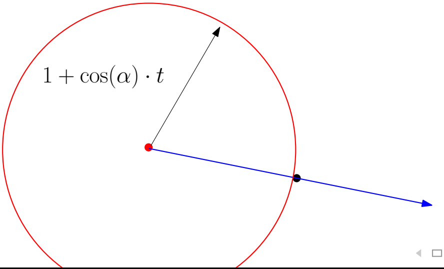
- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit cos(α), $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!

Bewegungsplanung: Feuerbekämpfung

- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit cos(α), $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!

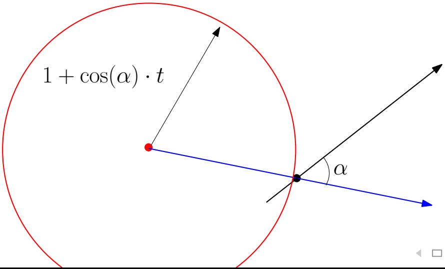


- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit cos(α), $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!

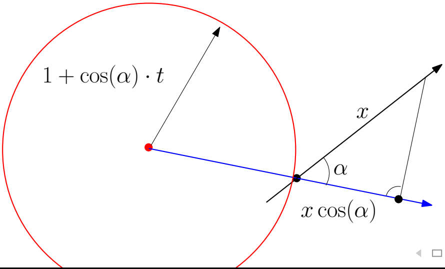


Bewegungsplanung: Feuerbekämpfung

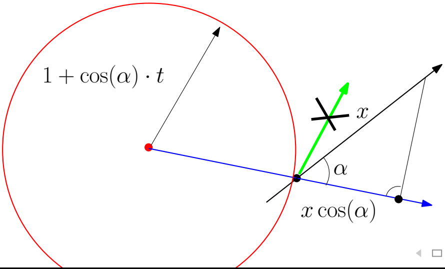
- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit $\cos(\alpha)$, $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!



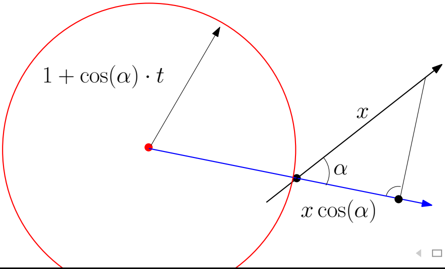
- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit cos(α), $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!



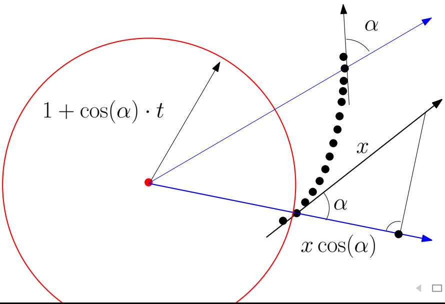
- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit $\cos(\alpha)$, $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!



- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit $\cos(\alpha)$, $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!



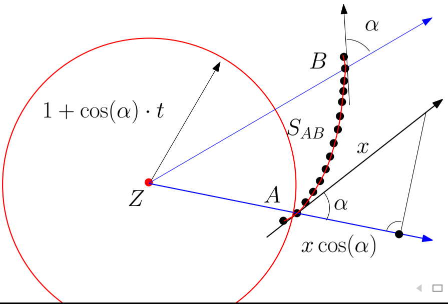
- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit $\cos(\alpha)$, $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!



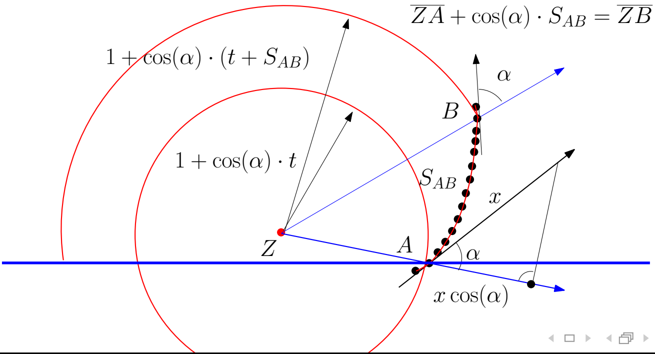
Bewegungsplanung: Feuerbekämpfung

- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit $\cos(\alpha)$, $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!

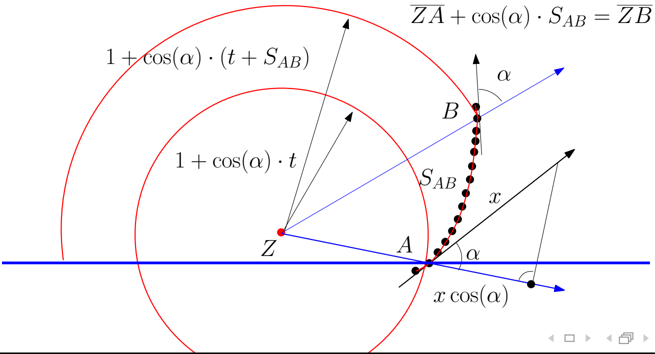
$$\overline{ZA} + \cos(\alpha) \cdot S_{AB} = \overline{ZB}$$



- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit $\cos(\alpha)$, $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!



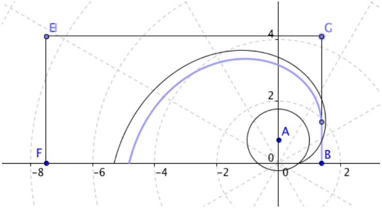
- Für $\alpha \in (0, \pi/4]$ keine Lösung!
- Geht das, wenn die Firefigther beliebig graben dürfen?
- Idee: Starte nah am Feuer und bleibe dran!
- Geschwindigkeit $\cos(\alpha)$, $\alpha \in (0, \pi/2]$
- Kontinuierlich lokal in Richtung α!
- Weg: Logarithmische Spirale mit Exzentrizität α!



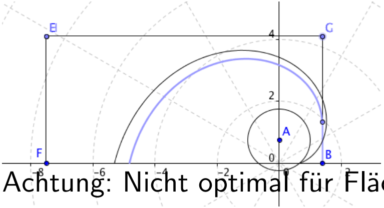
Bewegungsplanung: Feuerbekämpfung

- Mit Spiralen eine bessere/optimale Lösung erstellen!
- Verschiedene Kandidaten!
- Optimale Lösung: Zeitminimierung für jedes α!!!

- Mit Spiralen eine bessere/optimale Lösung erstellen!
- Verschiedene Kandidaten!
- Optimale Lösung: Zeitminimierung für jedes α!!!
- Begründung! (Beweis!) Gilt für alle $\alpha \in (0, \pi/2]$



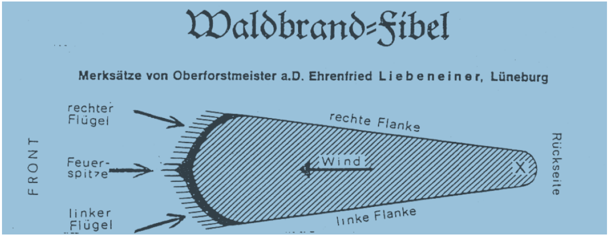
- Mit Spiralen eine bessere/optimale Lösung erstellen!
- Verschiedene Kandidaten!
- Optimale Lösung: Zeitminimierung für jedes α!!!
- Begründung! (Beweis!) Gilt für alle $\alpha \in (0, \pi/2]$



Achtung: Nicht optimal für Flächenminimierung!

Bewegungsplanung: Feuerbekämpfung

- Waldbrände, Grundsätze
- Ausbreitungsgeschwindigkeit klein: 0.5 km/h
- Graben mit Winkel
- Veränderte Ausbreitungskurven
- Modellerweiterungen



Mögliche Aufgabenstellungen

- Kontinuierliches Modell
- Beschränkung der verbrannten Fläche
- Wege mit ein/zwei Agenten, Wege mit wenig Knicken
- Diskretes Modell
- Feuermeldungspfad (auch kontinuierlich)
- Optimale Vorabschneise plus Bekämpfung
- Grid-Graphen oder allgemeine Graphen

Diskrete Problemstellung: Strategic Deployment

- Starting form a basis camp with a set of agents

Diskrete Problemstellung: Strategic Deployment

- Starting form a basis camp with a set of agents
- Take over and control some settlements

Diskrete Problemstellung: Strategic Deployment

- Starting form a basis camp with a set of agents
- Take over and control some settlements
- Resistance in the outback

Diskrete Problemstellung: Strategic Deployment

- Starting form a basis camp with a set of agents
- Take over and control some settlements
- Resistance in the outback
- Enough agents for the movements between the settlements and for controlling the settlements

Diskrete Problemstellung: Strategic Deployment

- Starting form a basis camp with a set of agents
- Take over and control some settlements
- Resistance in the outback
- Enough agents for the movements between the settlements and for controlling the settlements
- Task: Move *efficiently* around and occupy the settlements

Diskrete Problemstellung: Strategic Deployment

- Starting form a basis camp with a set of agents
- Take over and control some settlements
- Resistance in the outback
- Enough agents for the movements between the settlements and for controlling the settlements
- Task: Move *efficiently* around and occupy the settlements
- Historic examples!
  Gaius Julius Ceasar: Conquer of the Gauls (58 to 51 B.C.)
  Alexander the Great (356 B.C. to 323 B.C.): Alexander's campaign

Diskrete Problemstellung: Strategic Deployment

- Starting form a basis camp with a set of agents
- Take over and control some settlements
- Resistance in the outback
- Enough agents for the movements between the settlements and for controlling the settlements
- Task: Move *efficiently* around and occupy the settlements
- Historic examples!
  Gaius Julius Ceasar: Conquer of the Gauls (58 to 51 B.C.)
  Alexander the Great (356 B.C. to 323 B.C.): Alexander's campaign
- Modeled by an edge an vertex-weigthed graph

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s
- Rules for the movement:

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s
- Rules for the movement:
  1. Edge e, weight w_e: *Lower bound* on the number of agents required for traversal of e.

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s
- Rules for the movement:
  1. Edge e, weight w_e: *Lower bound* on the number of agents required for traversal of e.
  2. Vertex v, weight w_v: Number of agents that have to be placed at the vertex.

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s
- Rules for the movement:
  1. Edge e, weight w_e: *Lower bound* on the number of agents required for traversal of e.
  2. Vertex v, weight w_v: Number of agents that have to be placed at the vertex.
  3. First visit of v: Full amount w_v have to be placed, these agents cannot be removed any more.

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s
- Rules for the movement:
    1. Edge e, weight w_e: *Lower bound* on the number of agents required for traversal of e.
    2. Vertex v, weight w_v: Number of agents that have to be placed at the vertex.
    3. First visit of v: Full amount w_v have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s
- Rules for the movement:
  1. Edge e, weight w_e: *Lower bound* on the number of agents required for traversal of e.
  2. Vertex v, weight w_v: Number of agents that have to be placed at the vertex.
  3. First visit of v: Full amount w_v have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly
- Interesting computational questions:

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s
- Rules for the movement:
  1. Edge e, weight w_e: *Lower bound* on the number of agents required for traversal of e.
  2. Vertex v, weight w_v: Number of agents that have to be placed at the vertex.
  3. First visit of v: Full amount w_v have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly
- Interesting computational questions:
  How many agents are required in total?

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s
- Rules for the movement:
  1. Edge e, weight w_e: *Lower bound* on the number of agents required for traversal of e.
  2. Vertex v, weight w_v: Number of agents that have to be placed at the vertex.
  3. First visit of v: Full amount w_v have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly
- Interesting computational questions:
  How many agents are required in total?
  How long does this take?

Model of the Problem

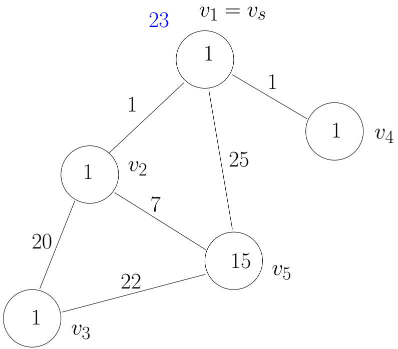
- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s
- Rules for the movement:
  1. Edge e, weight w_e: *Lower bound* on the number of agents required for traversal of e.
  2. Vertex v, weight w_v: Number of agents that have to be placed at the vertex.
  3. First visit of v: Full amount w_v have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly
- Interesting computational questions:
  How many agents are required in total?
  How long does this take?
  k agents given: How many settlements can we get?

Model of the Problem

- Edge- and vertex-weigthed graph G, Graph is fully known
- Some start vertex v_s
- Rules for the movement:
  1. Edge e, weight w_e: *Lower bound* on the number of agents required for traversal of e.
  2. Vertex v, weight w_v: Number of agents that have to be placed at the vertex.
  3. First visit of v: Full amount w_v have to be placed, these agents cannot be removed any more.
- Visit and occupy the vertices accordingly
- Interesting computational questions:
  How many agents are required in total?
  How long does this take?
  k agents given: How many settlements can we get?

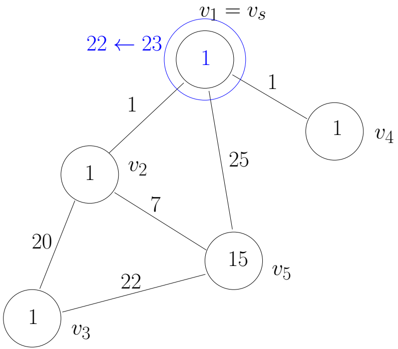
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



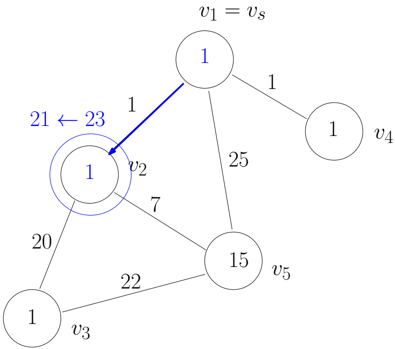
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



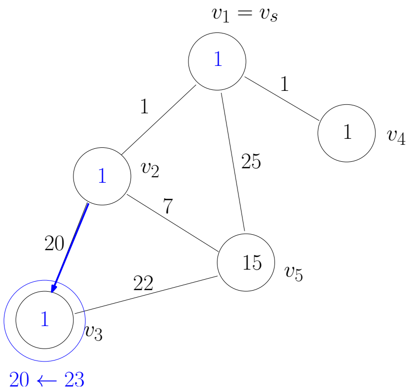
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



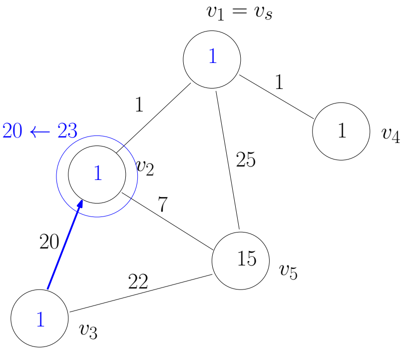
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



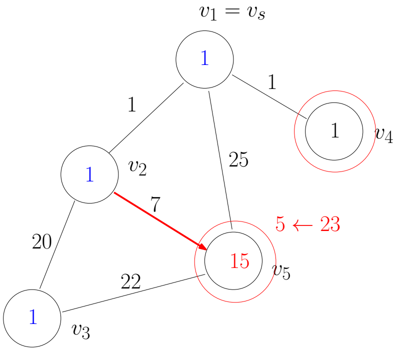
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



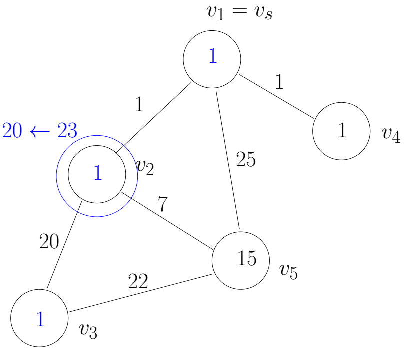
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



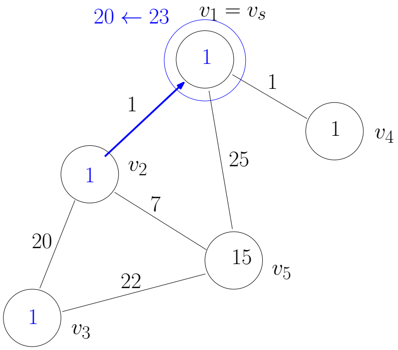
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



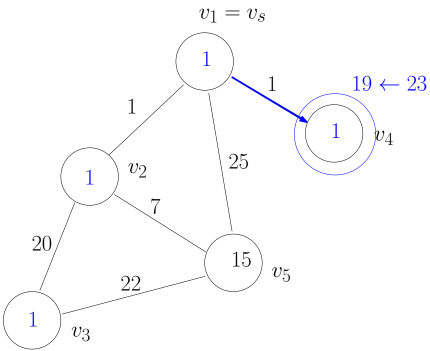
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



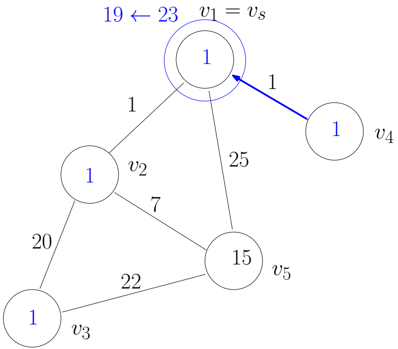
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



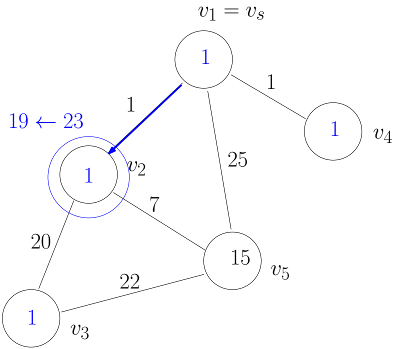
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



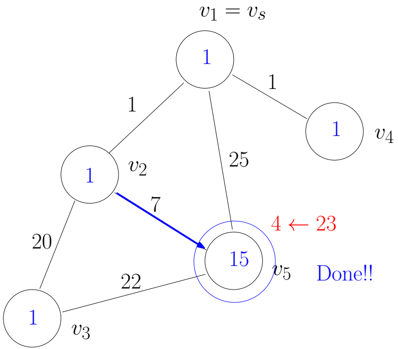
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



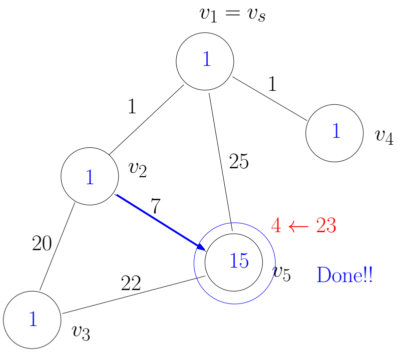
Example: Minimal number of agents required is 23!

1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



Example: Minimal number of agents required is 23!

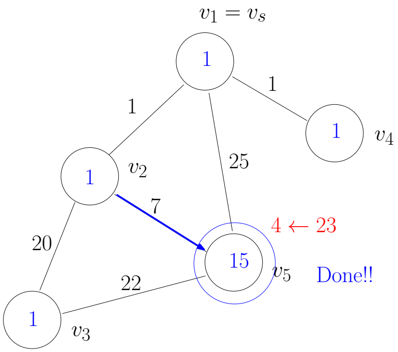
1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



4 agents remain unsettled! No other strategy is better!!

Example: Minimal number of agents required is 23!

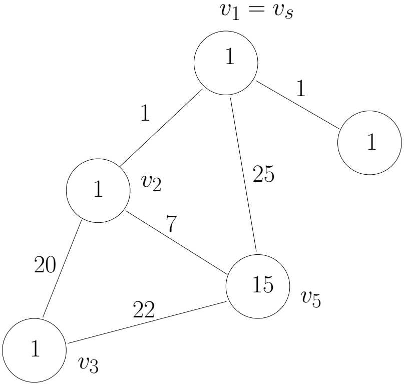
1. At least w_e for traversing edge e are required
2. At the first visit of v exactly w_v have to be placed and will never by removed!



4 agents remain unsettled! No other strategy is better!!
Is the problem clear?

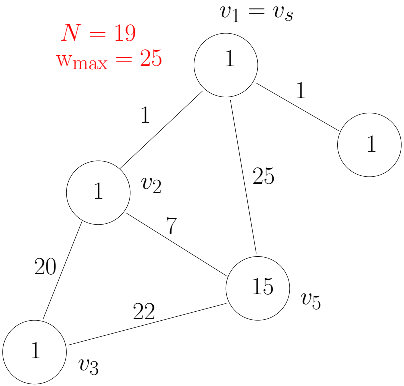
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$



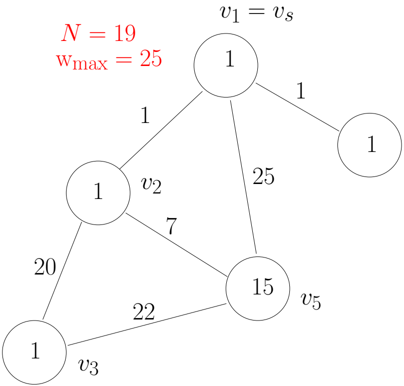
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$



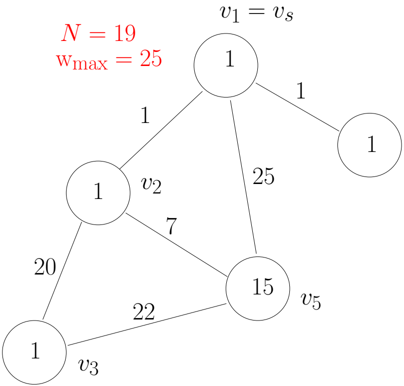
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$
- $N + w_{\max}$ on G is sufficient!



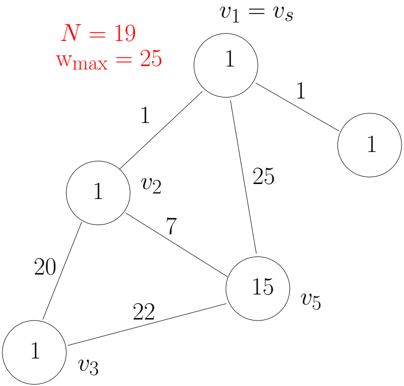
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$
- $N + w_{\max}$ on G is sufficient!
- Strategy S:



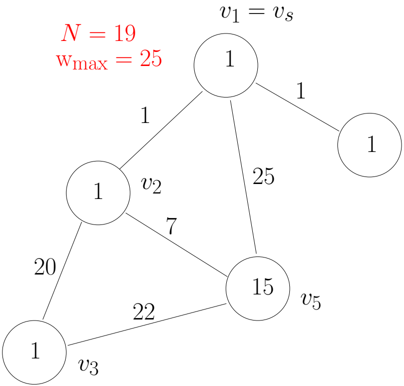
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$
- $N + w_{\max}$ on G is sufficient!
- Strategy S:
  $w_S := \max\{w_e | e \text{ was visited by } S\}$



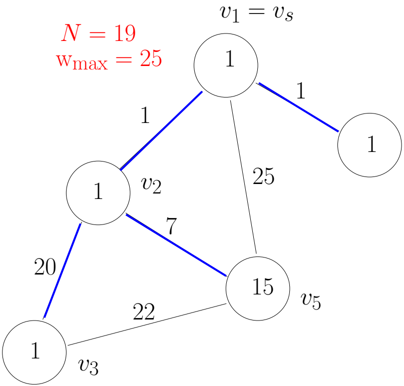
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$
- $N + w_{\max}$ on G is sufficient!
- Strategy S:
  $w_S := \max\{w_e | e$ was visited by $S\}$
  S requires at most $N + w_S$
  S requires at least $\max\{N, w_S\}$



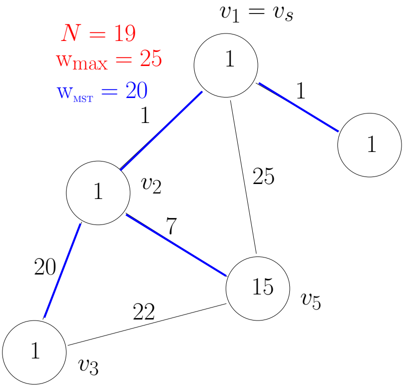
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$
- $N + w_{\max}$ on G is sufficient!
- Strategy S:
  $w_S := \max\{w_e | e \text{ was visited by } S\}$
  S requires at most $N + w_S$
  S requires at least $\max\{N, w_S\}$
- Minimum Spanning Tree (MST),
  $w_{\text{MST}} := \max\{w_e | e \in \text{ MST}\}$



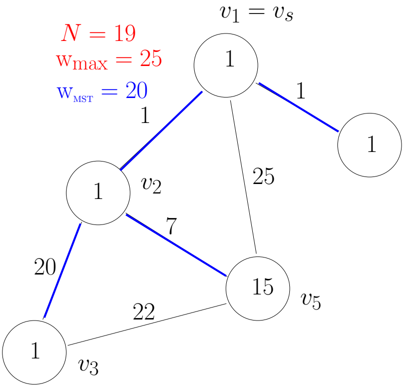
Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$
- $N + w_{\max}$ on G is sufficient!
- Strategy S:
  $w_S := \max\{w_e | e \text{ was visited by } S\}$
  S requires at most $N + w_S$
  S requires at least $\max\{N, w_S\}$
- Minimum Spanning Tree (MST),
  $w_{\text{MST}} := \max\{w_e | e \in \text{ MST}\}$

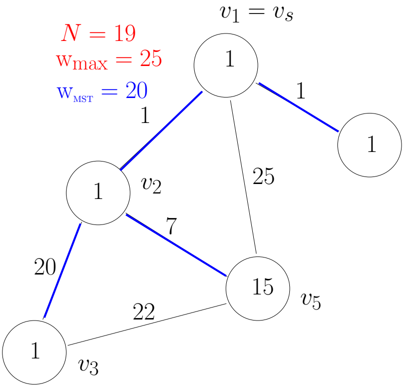


Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$
- $N + w_{\max}$ on G is sufficient!
- Strategy S:
  $w_S := \max\{w_e | e \text{ was visited by } S\}$
  S requires at most $N + w_S$
  S requires at least $\max\{N, w_S\}$
- Minimum Spanning Tree (MST),
  $w_{MST} := \max\{w_e | e \in \text{ MST}\}$
- $N + w_{MST}$ on MST is sufficient

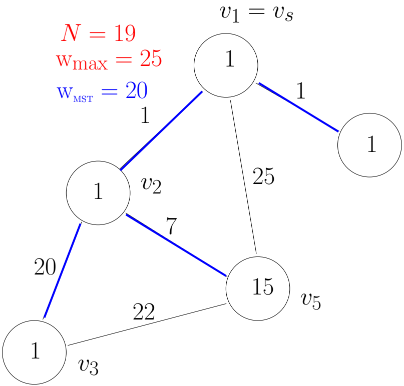


- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$
- $N + w_{\max}$ on G is sufficient!
- Strategy S:
  $w_S := \max\{w_e | e \text{ was visited by } S\}$
  S requires at most $N + w_S$
  S requires at least $\max\{N, w_S\}$
- Minimum Spanning Tree (MST),
  $w_{\text{MST}} := \max\{w_e | e \in \text{ MST}\}$
- $N + w_{\text{MST}}$ on MST is sufficient
  Any Strategy S on G requires
  at least $\max\{N, w_{\text{MST}}\}$



Number of agents required: Simple bounds!

- $G = (V, E)$, $N := \sum_{v \in V} w_v$
  $w_{\max} := \max\{w_e | e \in E\}$
- $N + w_{\max}$ on G is sufficient!
- Strategy S:
  $w_S := \max\{w_e | e \text{ was visited by } S\}$
  S requires at most $N + w_S$
  S requires at least $\max\{N, w_S\}$
- Minimum Spanning Tree (MST),
  $w_{MST} := \max\{w_e | e \in \text{ MST}\}$
- $N + w_{MST}$ on MST is sufficient
  Any Strategy S on G requires
  at least $\max\{N, w_{MST}\}$
- **Lemma:** Optimal Strategy for MST gives 2-Approximation for G



Variants: Return or No-Return!

(No-return) It suffices to fill the vertices as required, no agents have to return to the start vertex.

Variants: Return or No-Return!

(No-return) It suffices to fill the vertices as required, no agents have to return to the start vertex.

(Return) Finally some agents have to return to the start vertex and report the success of the whole operation.

Variants: Return or No-Return!

(No-return) It suffices to fill the vertices as required, no agents have to return to the start vertex.

(Return) Finally some agents have to return to the start vertex and report the success of the whole operation.

Comparable to *routes* (round-trips) and *tours* (open paths) in TSP

Variants: Return or No-Return!

(No-return) It suffices to fill the vertices as required, no agents have to return to the start vertex.

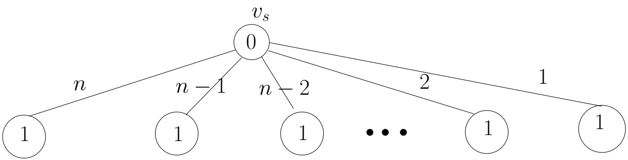
(Return) Finally some agents have to return to the start vertex and report the success of the whole operation.

Comparable to *routes* (round-trips) and *tours* (open paths) in TSP

Reporting the success formally means:
Set, M, of agents return to v_s, the union of *all* vertices visited by the members of M equals V.

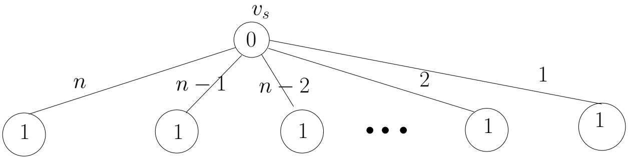
Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!



Optimal Algorithm for Trees: Return Variant

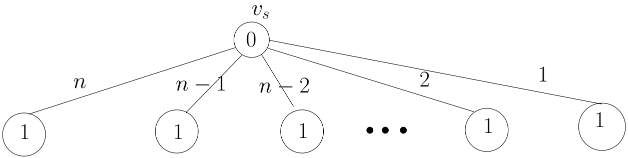
Computational lower bound and algorithmic idea! Example!



Optimal strategy: $n+1$ agents, visit vertices in order of decreasing edge weights: n,$n-1$,$n-2$,...,2,1

Optimal Algorithm for Trees: Return Variant

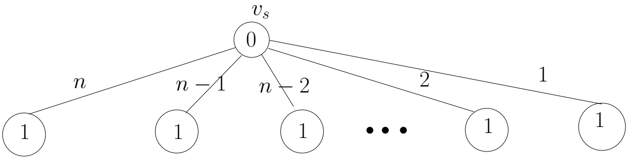
Computational lower bound and algorithmic idea! Example!



Optimal strategy: $n+1$ agents, visit vertices in order of decreasing edge weights: n,$n-1$,$n-2$,...,2,1
Any other order will increase the number!

Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!

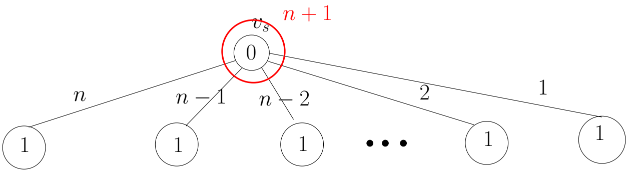


Optimal strategy: $n+1$ agents, visit vertices in order of decreasing edge weights: n,$n-1$,$n-2$,...,2,1

Any other order will increase the number!

Example: Visit $n-2$ before n

Computational lower bound and algorithmic idea! Example!



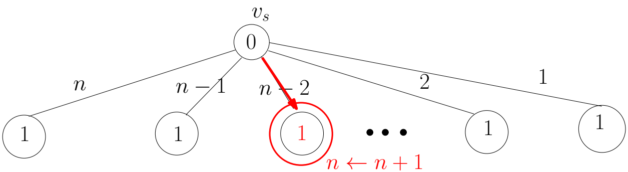
Optimal strategy: $n+1$ agents, visit vertices in order of decreasing edge weights: $n, n-1, n-2, \ldots, 2, 1$

Any other order will increase the number!

Example: Visit $n-2$ before n

Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!



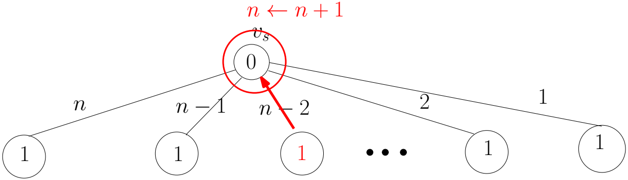
Optimal strategy: $n+1$ agents, visit vertices in order of decreasing edge weights: n,$n-1$,$n-2$,...,2,1

Any other order will increase the number!

Example: Visit $n-2$ before n

Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!



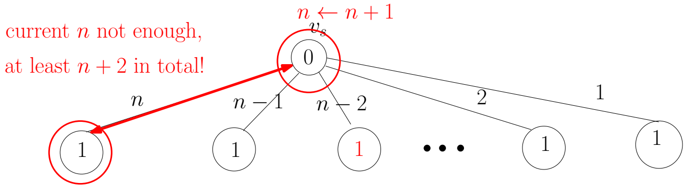
Optimal strategy: $n+1$ agents, visit vertices in order of decreasing edge weights: n,$n-1$,$n-2$,...,2,1

Any other order will increase the number!

Example: Visit $n-2$ before n

Optimal Algorithm for Trees: Return Variant

Computational lower bound and algorithmic idea! Example!

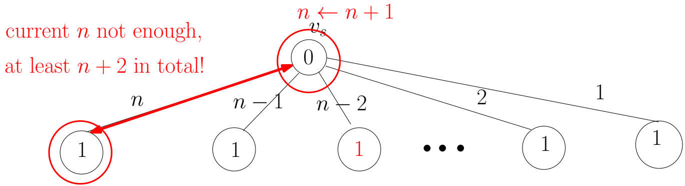


Optimal strategy: $n+1$ agents, visit vertices in order of decreasing edge weights: n,$n-1$,$n-2$,...,2,1

Any other order will increase the number!

Example: Visit $n-2$ before n

Computational lower bound and algorithmic idea! Example!



Optimal strategy: $n+1$ agents, visit vertices in order of decreasing edge weights: n,$n-1$,$n-2$,...,2,1

Any other order will increase the number!

Example: Visit $n-2$ before n

Lemma: Computational lower bound $\Omega(n \log n)$ by sorting (both variants, but real weights)!

Mögliche Aufgabenstellungen

- Im allgemeinen NP-hard
- Verschiedene Modelle betrachten
- Vollständiger Graph gegeben, gleiche Gewichte, Kantenauswahl?
- Planarer Graph, Euklidische Distanzen als Gewichte, Knotengewicht 1
- Festgelegte Gewichte, Z.B.: 1 oder 2?
- Knoten gesichert, dann Kanten gesichert, freies bewegen, Unterschied?
- Graphen, Bäume