Technische Universität Braunschweig

Mathematische Methoden der Algorithmik – Vorlesung #11

Arne Schmidt

Traveling Salesman Problem

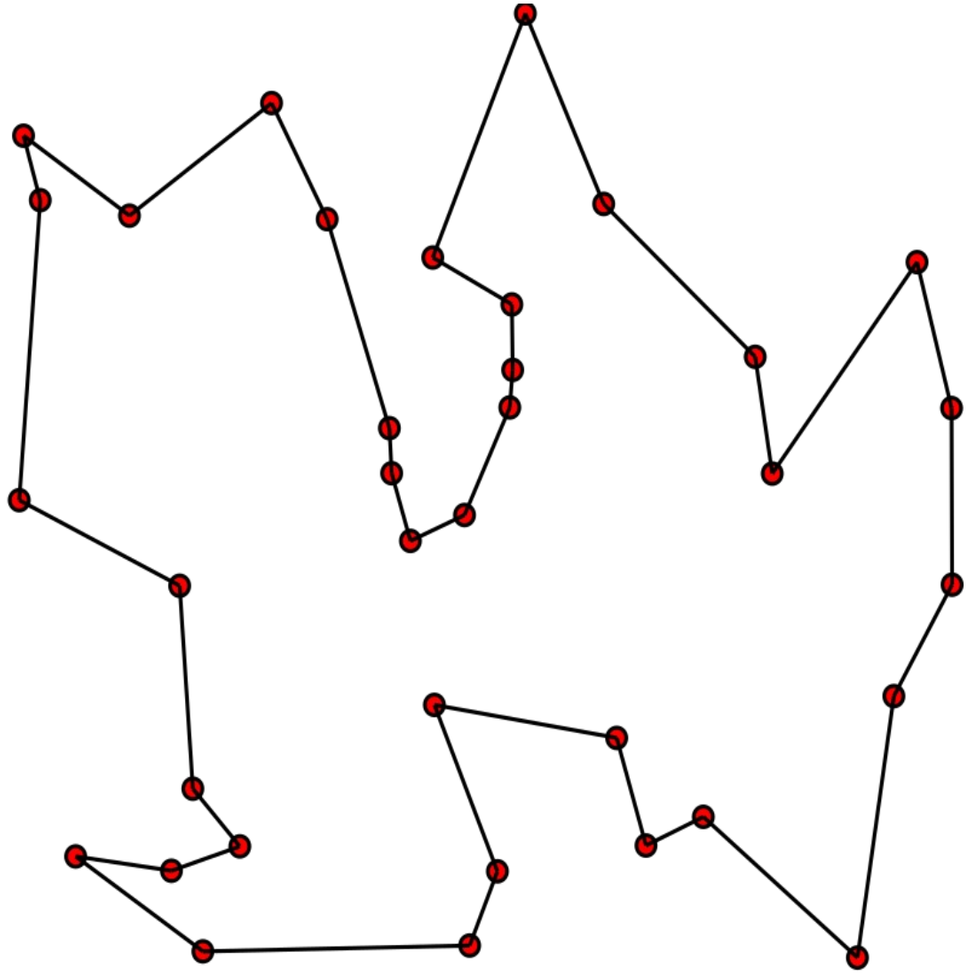
Ein Reisender

Ein Reisender möchte alle großen Städte eines Landes besuchen, dabei aber so wenig wie möglich Zeit mit dem tatsächlichen Reisen verbringen.

Klar ist:
Der Reisende kann sich nicht aufteilen.

Also:
Entweder gehen wir von Stadt x zu Stadt y, oder wir lassen es sein.

Wie kann der Reisende nun eine kürzeste Tour bestimmen?



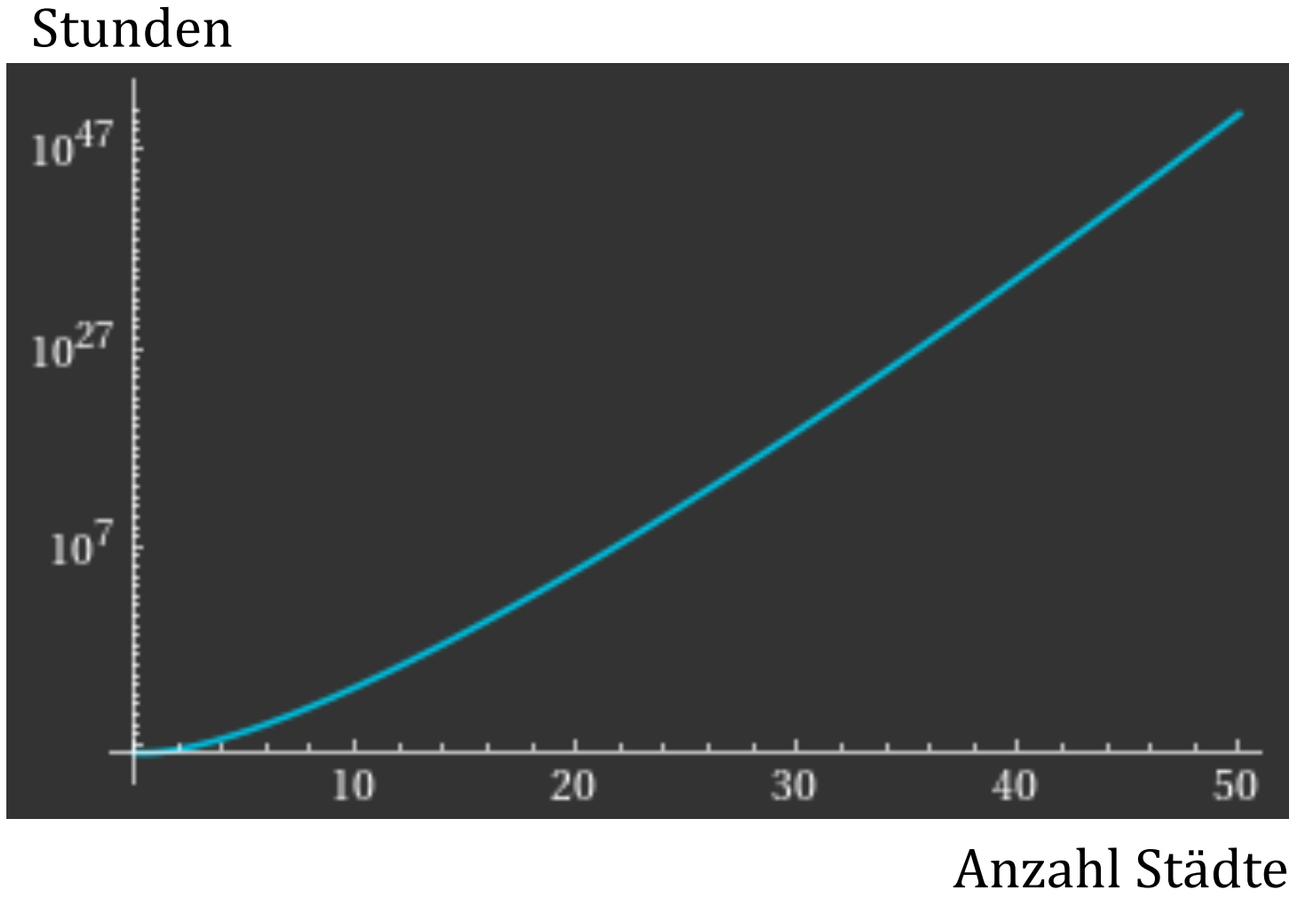
Brute Force

Es gibt $(n-1)!$ viele Touren durch n Städte.

Angenommen, wir können 18 Milliarden Touren in der Sekunde durchprüfen.

Der log-Plot zeigt dann die benötigte Zeit in Stunden an, um die beste Lösung zu finden.

Brute Force scheidet definitiv aus!



Graphen vs. Punkte

Auf einem gegebenen Graph mit beliebigen Kosten ist es schwer gute Lösungen zu finden.

Wie sieht das auf „realistischen“ Instanzen aus?

Es ist NP-schwer, TSP mit Faktor k zu approximieren.

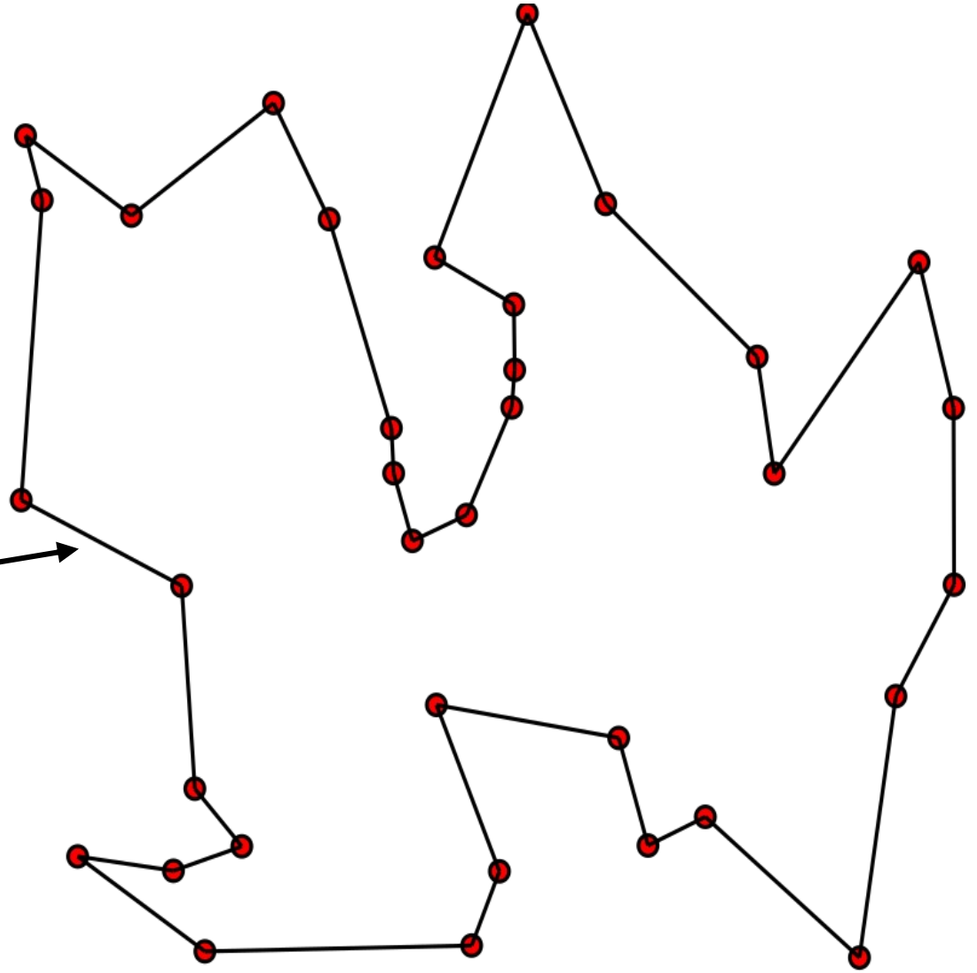
Euklidisches TSP - Definition

Gegeben: Punkte $p_1, \dots, p_n \in \mathbb{R}^2$
Gesucht: Eine Permutation π, sodass

$$\sum_{i=1}^{n} \left\| p_{\pi(i)} p_{\pi(i+1)} \right\|$$

mit $\pi(n+1) = \pi(1)$ minimal ist.

Länge der Kante ist die euklidische Distanz



Es existiert eine 1.5-Approximation für ETSP.

TSP-Formulierung – Idee 1

Wie formuliert man das als IP?

Erste Idee:

- Knotenvariablen $u \in \{1, \dots, n-1\}$
  Gibt die Permutation an.
- Kantenvariablen $x_{e^{\rightarrow}} \in \{0,1\}$
  Gibt die ausgewählte, gerichtete Kante an
  (Wir orientieren die Tour)

Nimmt man einen Knoten aus einer Tour, müssen die Knotenvariablen exakt aufsteigend sortiert sein.

TSP-Formulierung – Miller, Tucker, Zemlin

$$\min \sum_{e \in P \times P} c_e x_e$$

s.t.

$$\sum_{e \in \delta^+(p)} x_e = 1, \qquad \forall p \in P$$

$$\sum_{e \in \delta^-(p)} x_e = 1, \qquad \forall p \in P$$

$$u_i - u_j + n x_{ij} \leq n - 1, \qquad \forall i \neq j, i, j \in P \setminus \{p_1\}$$

$$u_i \in \{1, \dots, n-1\}$$

$$x_e \in \{0,1\}$$

c_e sind die Kosten der Kante. Sind entweder explizit (Graph) oder implizit (Punkte) gegeben.

Genau eine ausgehende Kante

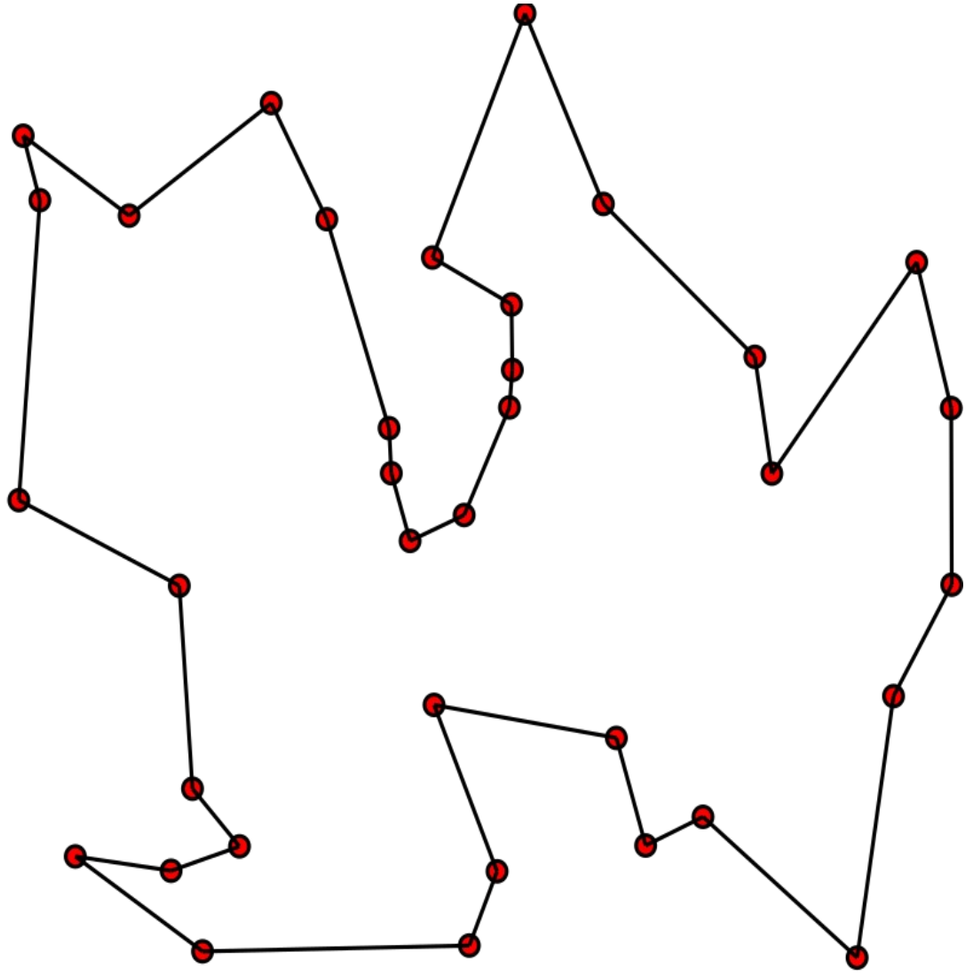
Genau eine eingehende Kante

Knoten müssen geordnet sein
($x_{ij} = 1 \Rightarrow u_j \geq u_i + 1$)

Vorteil: Polynomielle Größe.

Nachteil: Nutzt Big-M Methode (Wenn Kante nicht gewählt ist, kann die Differenz groß werden.)

TSP-Formulierung – Andere Idee

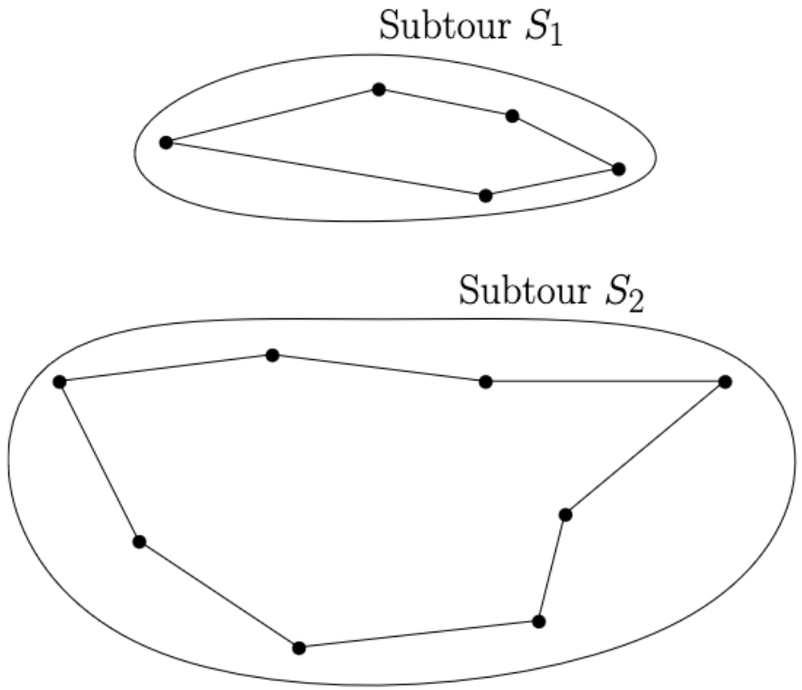


Betrachte Kanten ungerichtet.

1. An jedem Knoten liegen exakt zwei Kanten.
2. Die Tour ist zusammenhängend.

Constraint zu 1. ist einfach. Aber zu 2.?

TSP-Formulierung – Andere Idee



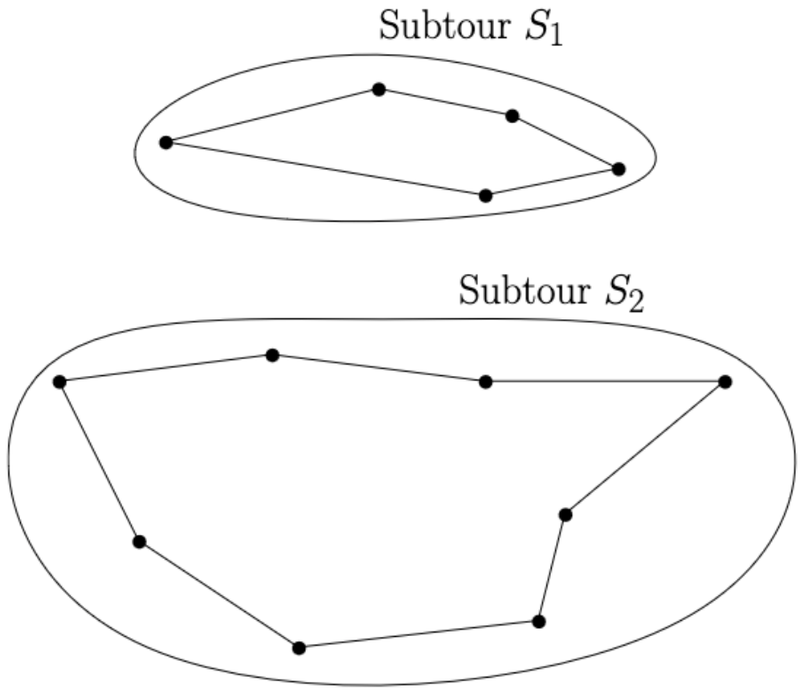
Betrachte Kanten ungerichtet.

1. An jedem Knoten liegen exakt zwei Kanten.
2. Die Tour ist zusammenhängend.

Constraint zu 1. ist einfach. Aber zu 2.?

Betrachte Menge S_1 hier herausführen?
→ Mindestens zwei!

TSP-Formulierung – Danzig, Fulkerson, Johnson



Degree-Constraint

Subtour-Constraint

$$\min \sum_{e \in P \times P} c_e x_e$$

s.t.

$$\sum_{e \in \delta(p)} x_e = 2, \qquad \forall p \in P$$

$$\sum_{e \in \delta(S)} x_e \geq 2, \qquad \forall \emptyset \neq S \setminus \subsetneq P$$

$$x_e \in \{0,1\}$$

Lazy Constraints

Problem:
Das IP hat exponentiell viele Constraints!

Wir brauchen aber u.U. nicht alle.

- Löse zunächst das IP ohne die Subtour-Constraints.
- Nach Erhalt der Lösung, prüfe, ob Constraints verletzt sind. Füge verletzte Subtour-Constraints hinzu!

$$\min \sum_{e \in P \times P} c_e x_e$$

s.t.

$$\sum_{e \in \delta(p)} x_e = 2, \qquad \forall p \in P$$

$$\sum_{e \in \delta(S)} x_e \geq 2, \qquad \forall \emptyset \neq S \backslash \subsetneq P$$

$$x_e \in \{0,1\}$$

Wir fügen Constraints also nur hinzu, wenn es unbedingt sein muss. Die Constraints werden auch „lazy constraints“ genannt.

Finden von verletzten Subtour-Constraints

Integral:

Nutze einen Graphscan-Algorithmus (DFS/BFS), um eine Komponente zu finden.

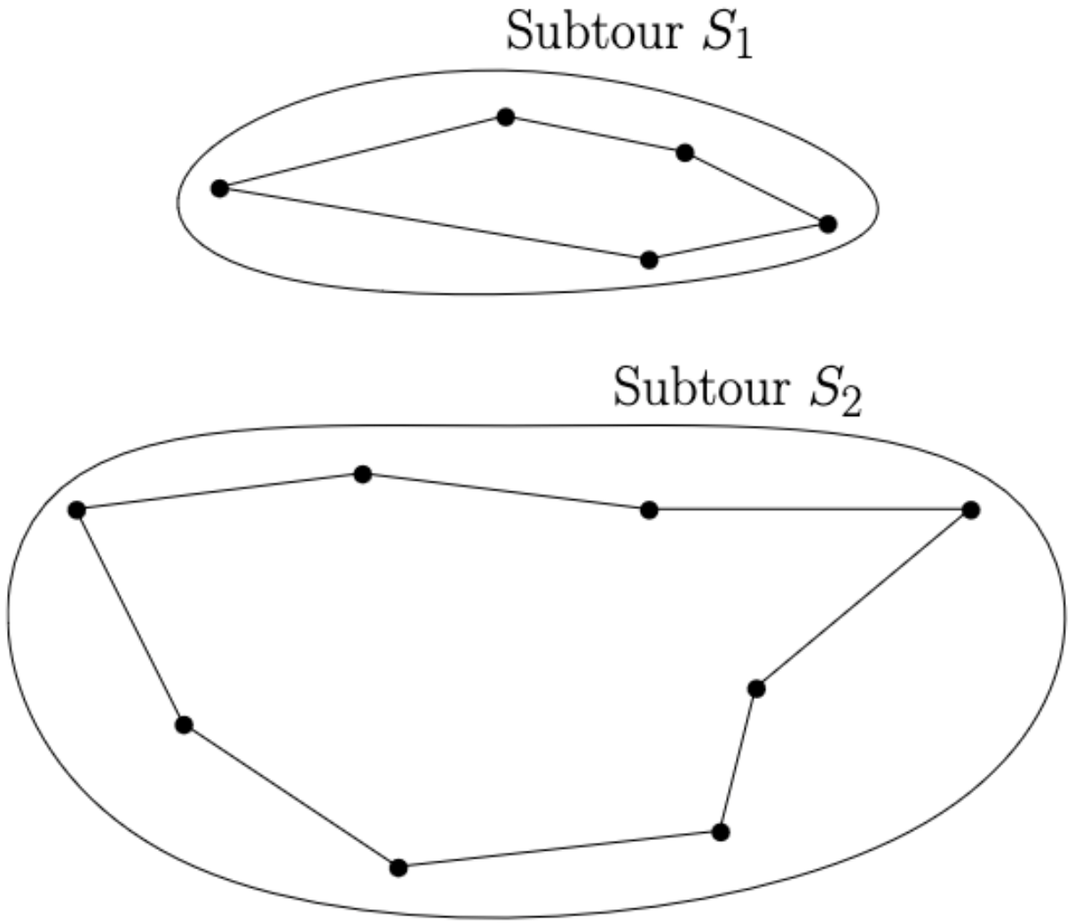
Enthält die Komponente nicht alle Knoten, dann wurde eine Subtour gefunden!

In diesem Fall:

$$\sum_{e \in \delta(S_1)} x_e = 0 < 2$$

⇒ Füge folgenden Constraint hinzu:

$$\sum_{e \in \delta(S_1)} x_e \geq 2$$



Finden von verletzten Subtour-Constraints

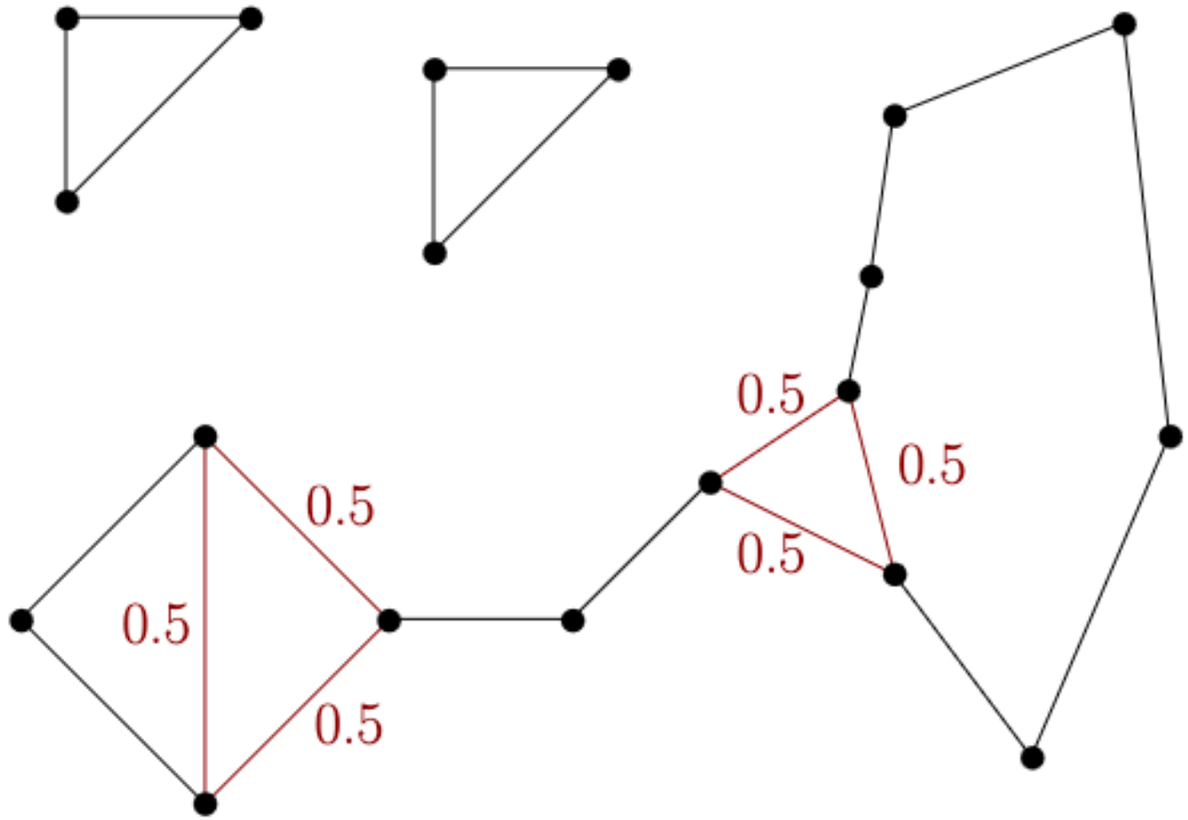
Fraktional:

Wieder BFS/DFS für einzelne Komponenten.

Ansonsten:
Für gültige Lösungen muss jeder *Cut* mindestens den Wert 2 besitzen.

Frage: Gibt es einen Cut, der kleiner als zwei ist?

Algorithmus:
Stoer-Wagner zum Finden eines globalen Minimum Cuts („minimum graph cut“).

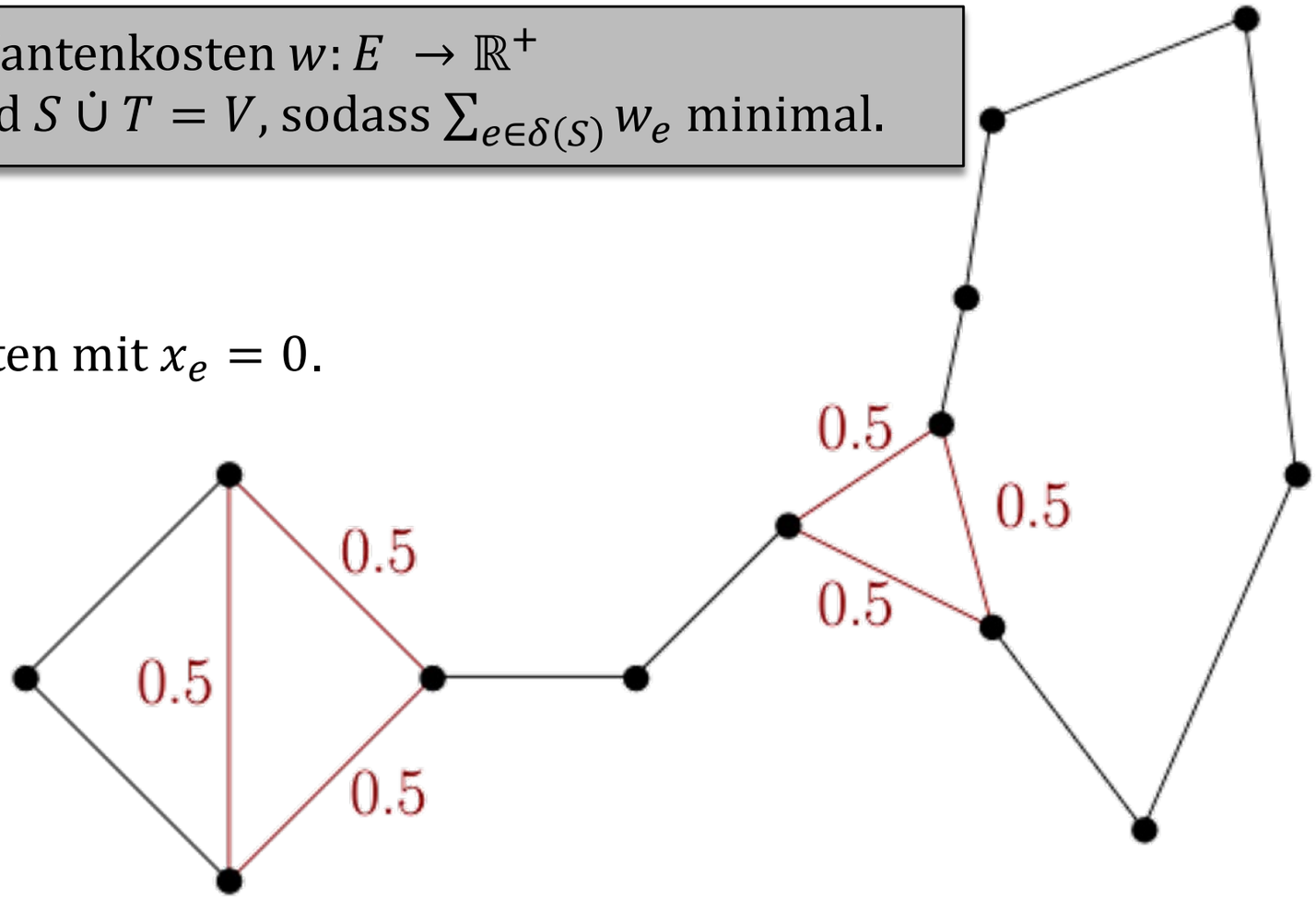


Minimum Graph Cut

Gegeben: Graph $G = (V, E)$ und Kantenkosten $w: E \to \mathbb{R}^+$
Gesucht: $S, T \subset V$ mit $S, T \neq \emptyset$ und $S \dot{\cup} T = V$, sodass $\sum_{e \in \delta(S)} w_e$ minimal.

In unserem Fall:
Setze $w_e := x_e$ und ignoriere Kanten mit $x_e = 0$.

Stoer-Wagner löst das Problem in Zeit $O(|V||E| + |V|^2 \log|V|)$.

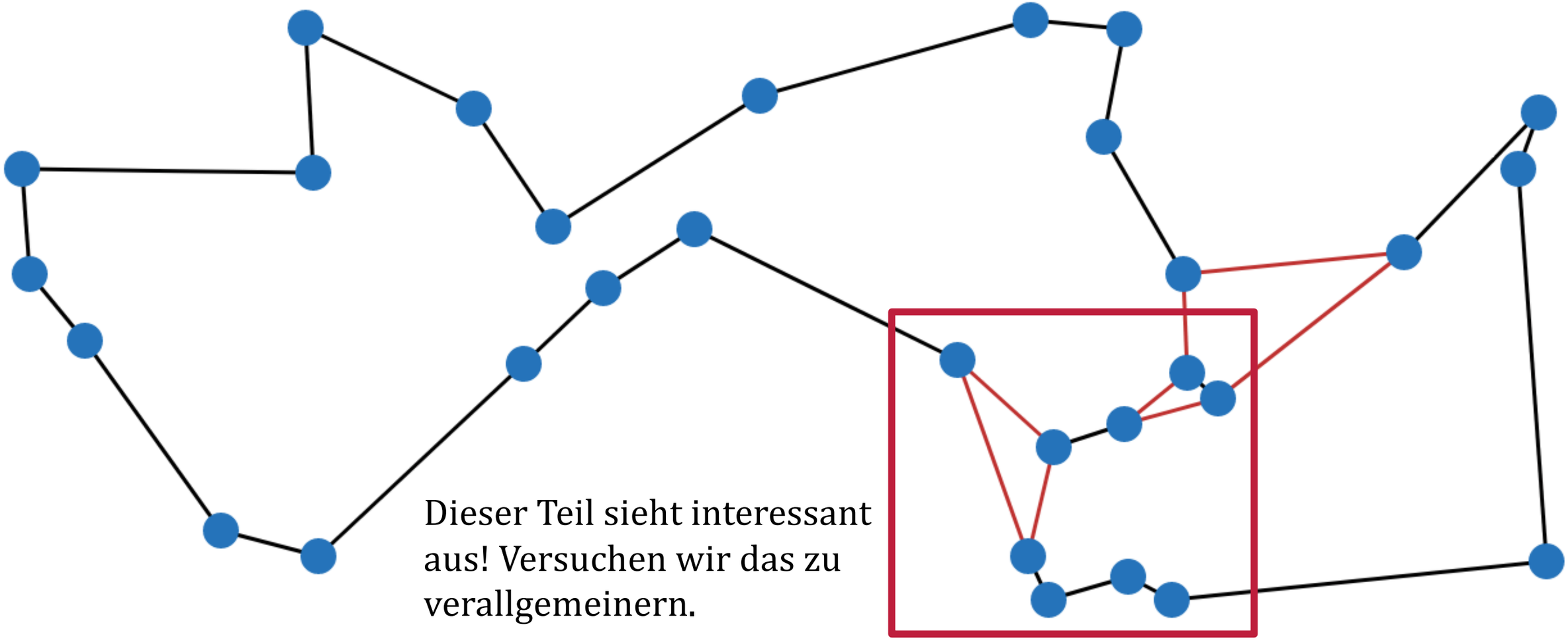


Live Demo

https://www.math.uwaterloo.ca/tsp/app/diy.html#

TSP – Cutting Planes

Cutting Planes

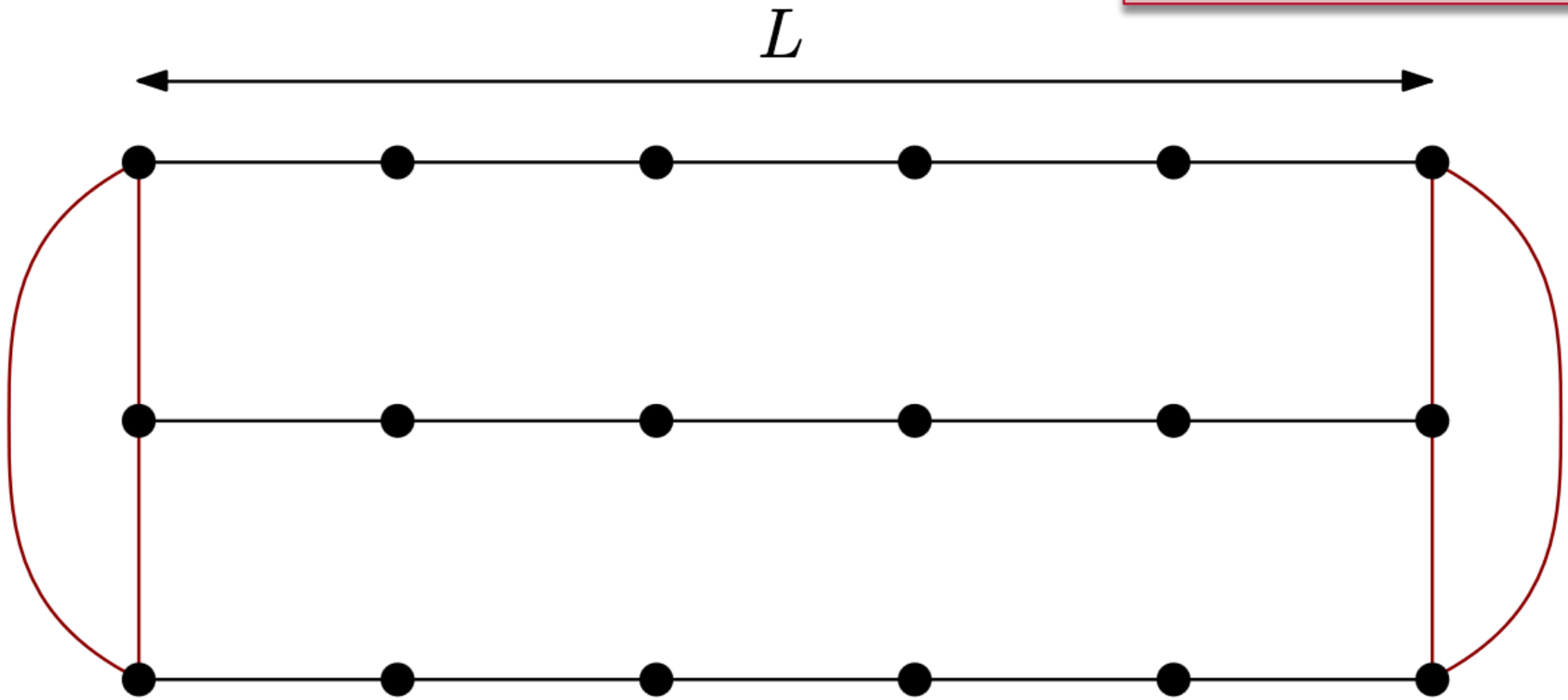


Dieser Teil sieht interessant aus! Versuchen wir das zu verallgemeinern.

Eine spezielle Situation

Alle Subtour-Constraints sind erfüllt.
Fraktionale Lösung geht für $L \to \infty$ gegen $3L + 4$.
Eine optimale, integrale Lösung hat Wert mindestens $4L$.
→ (Asymptotischer) Integrality Gap ist mindestens 4/3.

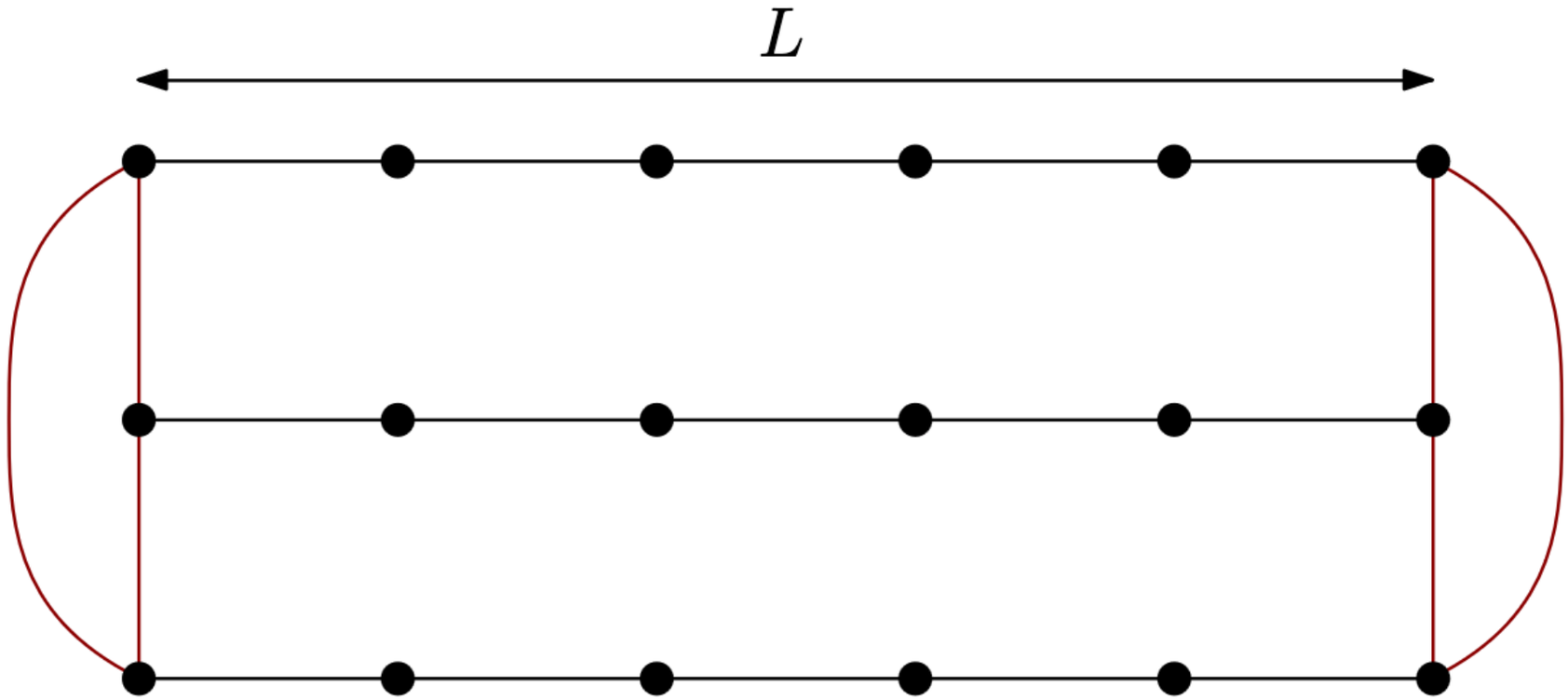
Vermutung:
Der Integrality Gap der Danzig-Formulierung ist 4/3.



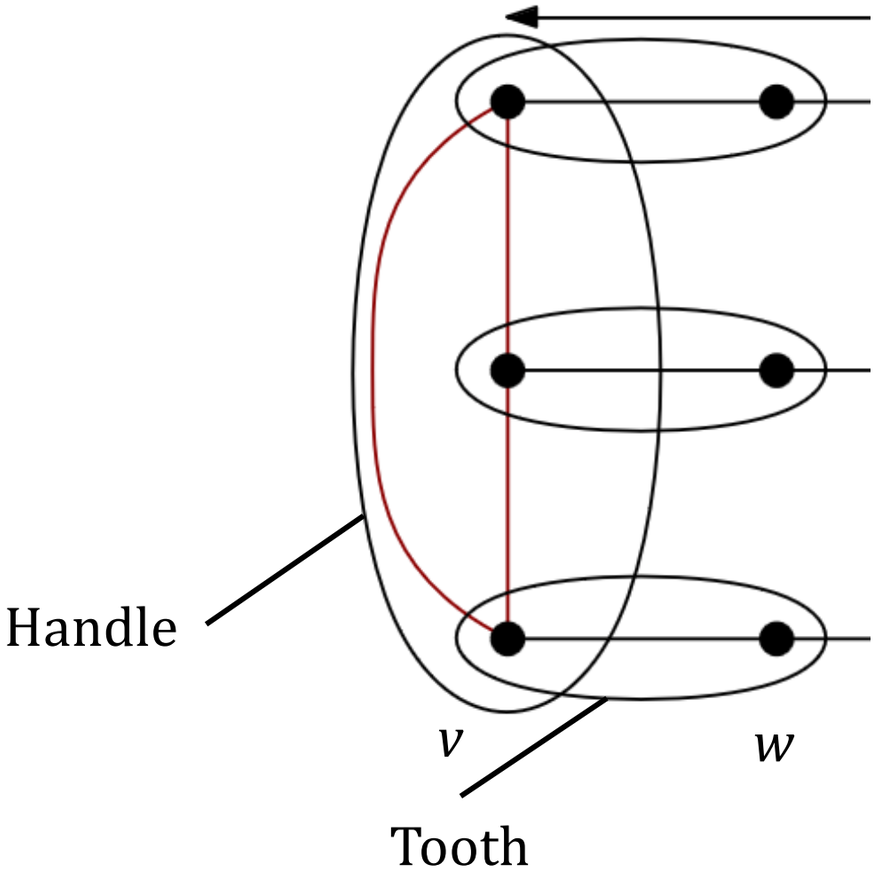
Eine spezielle Situation

Können wir hier noch etwas tun?
In diesem Beispiel: Für müssen die linke / rechte Seite entweder 2-mal oder 4-mal verlassen.

Hier passiert es aber an drei Stellen!



Comb-Ungleichungen



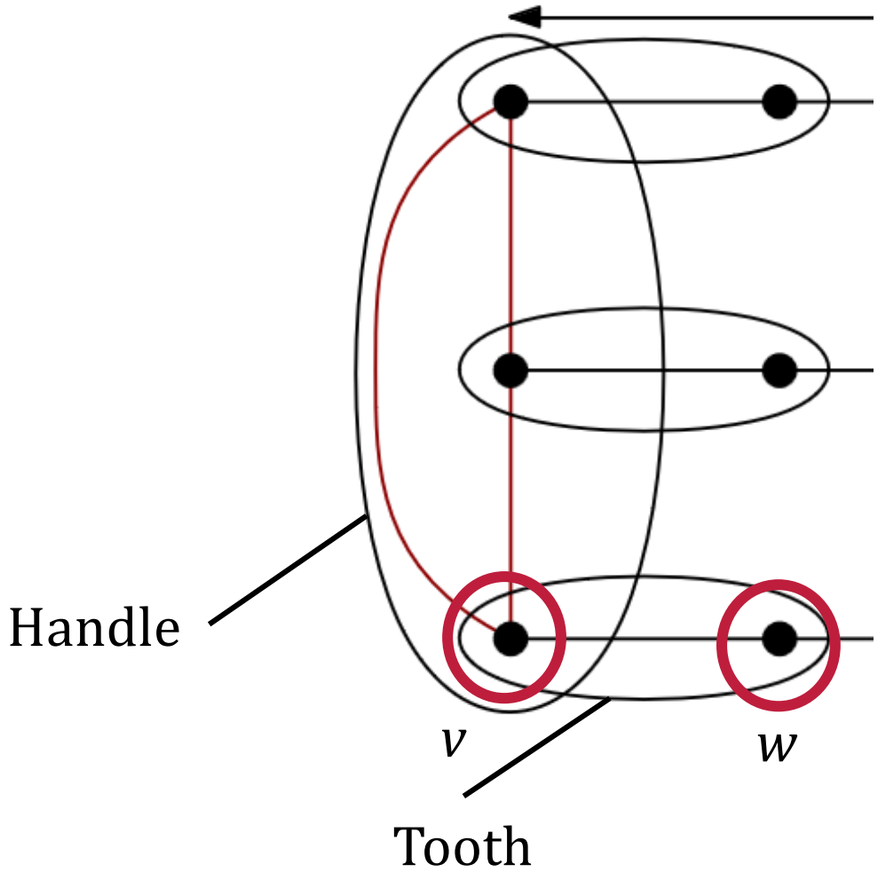
Eine Aufteilung von V in Knotenmengen $H, T_1, T_2, \dots, T_k$ mit

- $H \cap T_i \neq \emptyset$ für alle $i = 1, \dots, k$
- $T_i \setminus H \neq \emptyset$ für alle $i = 1, \dots, k$
- $T_i \cap T_j = \emptyset$ für alle $i \neq j \in \{1, \dots, k\}$
- k ungerade

wird **Comb** genannt. Dabei werden H der **Handle** und die T_i die **Teeth** des Combs genannt.

Betrachte was bei dem Handle bzgl. eines Tooth passiert!

Comb-Ungleichungen



Beide Knoten haben Grad 2. Fallunterscheidung für Handle H am Knoten v und Tooth T_i!
Sei dazu d_S die Anzahl an Kanten, die aus der Menge S herausführt und $d(v)$ der Grad eines Knoten.

1. Beide Kanten von v laufen innerhalb von H.
   Dann muss $d_{T_i}(v) + d_{T_i}(w) = d_{T_i} \geq 3$ sein.
2. Eine Kante von v läuft innerhalb von H.
   Dann muss $d_H(v) = 1$ und $d_{T_i} \geq 2$ sein.
3. Keine Kante von v läuft innerhalb von H.
   Dann ist $d_H(v) = 2$ und $d_{T_i}(w) \geq 1$.

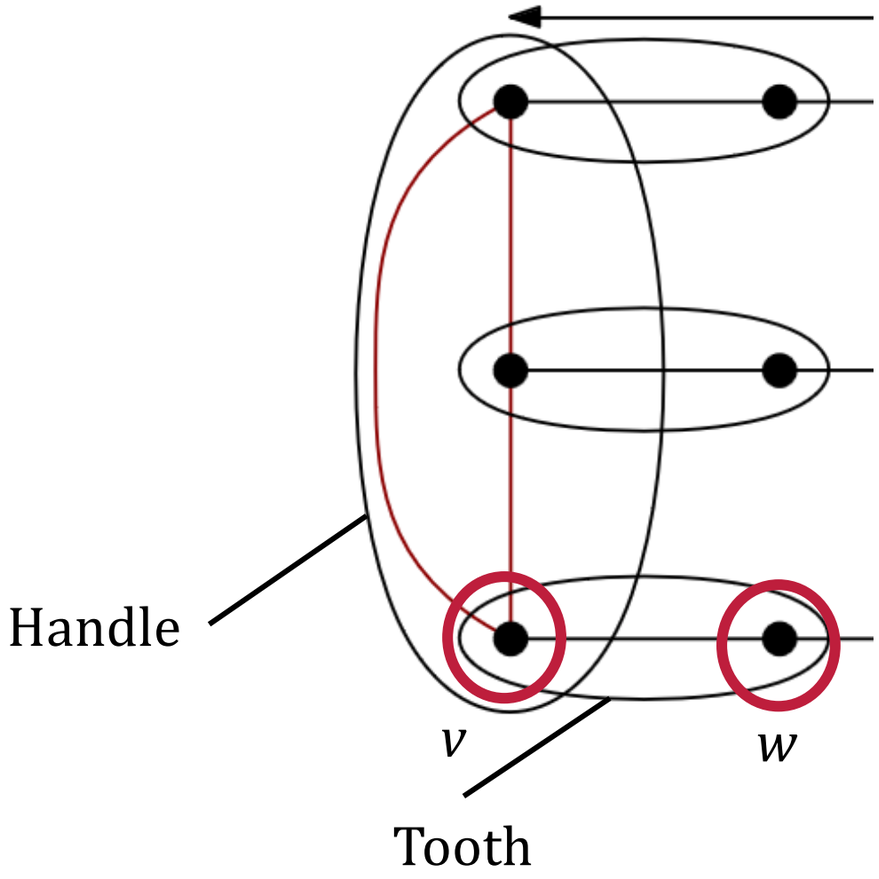
Damit ist also

$$d_H(v) + d_{T_i} \geq 3$$

Aufsummiert für alle Teeth:

$$d_H + \sum_{i=1}^{k} d_{T_i} \geq 3\mathrm{t}$$

Comb-Ungleichungen



Aufsummiert für alle Teeth:

$$d_H + \sum_{i=1}^{k} d_{T_i} \geq 3k$$

Nun müssen d_H und d_{T_i} gerade sein!

Da k ungerade ist also die linke Seite gerade und die rechte Seite der Ungleichung ungerade.

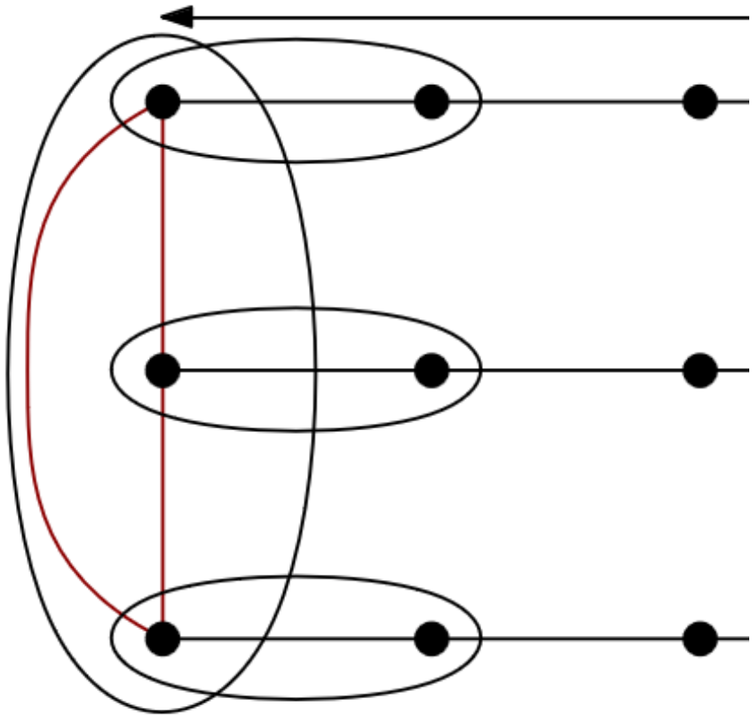
Also muss auch gelten:

$$d_H + \sum_{i=1}^{k} d_{T_i} \geq 3k + 1$$

Damit erhalten wir als Constraint:

$$\sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e \geq 3k + 1$$

Comb-Ungleichungen



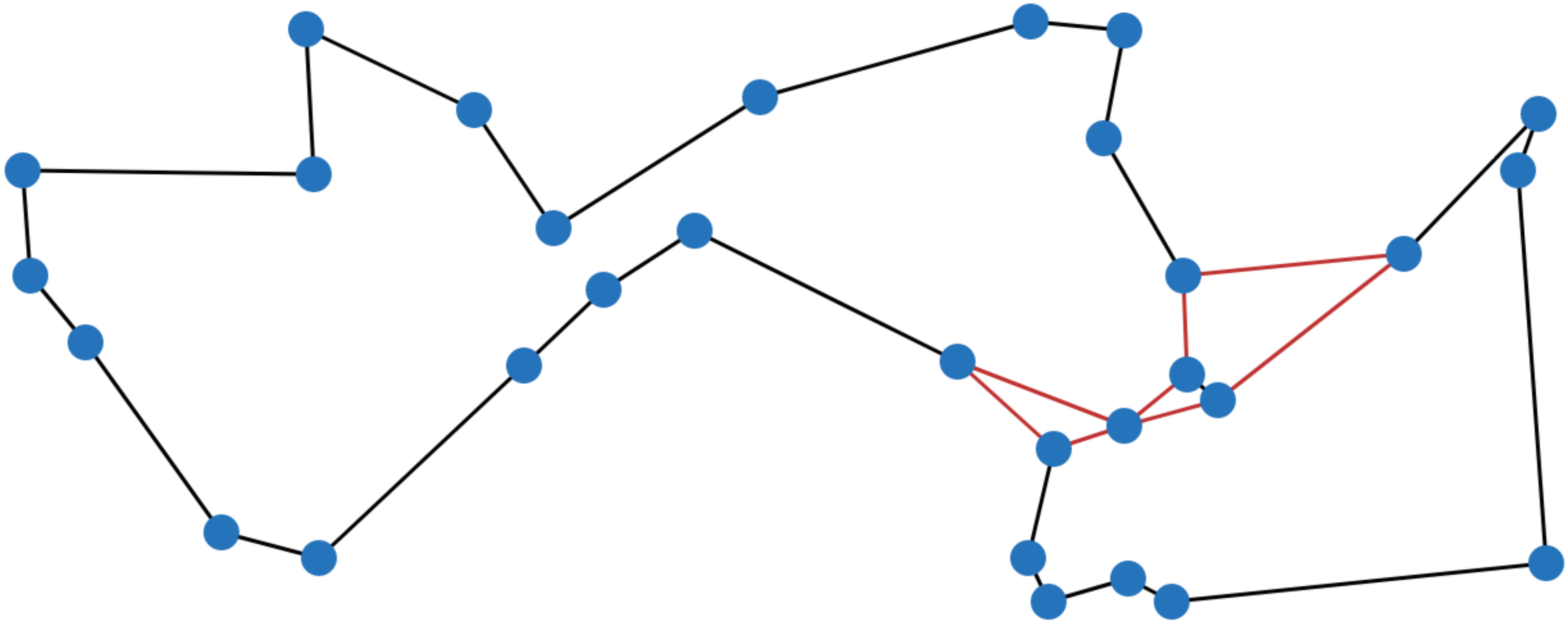
In dieser fraktionalen Lösung:

$$\sum_{e \in \delta(H)} x_e + \sum_{i=1}^{k} \sum_{e \in \delta(T_i)} x_e = 9$$

Gefordert sind aber $3k + 1 = 10$.

Wir können also einen neuen Constraint hinzufügen!

Lösung mit Comb-Ungleichungen



Comb-Ungleichungen

Es ist nicht bekannt, ob das Finden von Combs NP-schwer ist oder ob diese effizient gefunden werden können.

Es existieren aber gute Heuristiken, z.B.:
Suche den Handle über Komponenten, die durch fraktionale Kanten aufgespannt werden und Kanten als Teeth, die diese Komponente verlassen.

Branch-Cut-and-Price

Lösen von (E)TSP

TSP lässt sich über diese Constraints in der Regel schon sehr gut lösen.

Sehr große Instanzen (tausende Knoten), haben allerdings das Problem, dass der Speicher schnell zu wenig wird!

Gerade bei ETSP arbeiten wir implizit auf einem vollständigen Graphen. Alle Gewichte müssen irgendwo gespeichert werden!

Können wir nicht eine Teilmenge von Kanten betrachten?

Dann, nach und nach Kanten mit aufnehmen?

Sehr lange Kanten werden bspw. nicht betrachtet!

Branch-Cut-and-Price

Grundidee:

- Wähle Kandidatenmenge von Kanten
- Löse die Relaxierung inkl. Cutting Planes
- Prüfe, ob es Kanten gibt, die die Lösung noch verbessern könnten.
- Lösche ggf. Kanten und Cuts, die länger nicht mehr aktiv waren.

Dabei stellen sich die Fragen:

1. Wie wähle ich die Kanten aus?
2. Was passiert mit den Cutting Planes?

Kanten und Cutting Planes

Zunächst: Wir sind optimal, wenn $z^*_{\mathcal{N}} \geq 0$ ist, bzw. wenn wir dual zulässig sind.

Wenn es eine nicht-betrachtete Kante gibt, die unsere Lösung verbessern könnte, dann muss im dualen ein dazugehöriger Constraint existieren, der nicht erfüllt ist!

Suche also Variablen, die einen verletzten dualen Constraint hat, und füge sie hinzu.

Ähnliches gilt auch für Cutting Planes: Sie entsprechen Variablen im dualen!

Doch wie sieht das duale Problem überhaupt aus?

Das duale zu TSP

$$\min \sum_{e \in E} c_e x_e$$

s.t.

$$\sum_{e \in \delta(p)} x_e = 2, \qquad \forall p \in P$$

$$\sum_{e \in \delta(S)} x_e \geq 2, \qquad \forall \emptyset \neq S \setminus \subsetneq P$$

$$0 \leq x_e \leq 1$$

$$\max 2 \sum_{p \in P} y_p + 2 \sum_{S} y_S - \sum_{e \in E} y_e$$

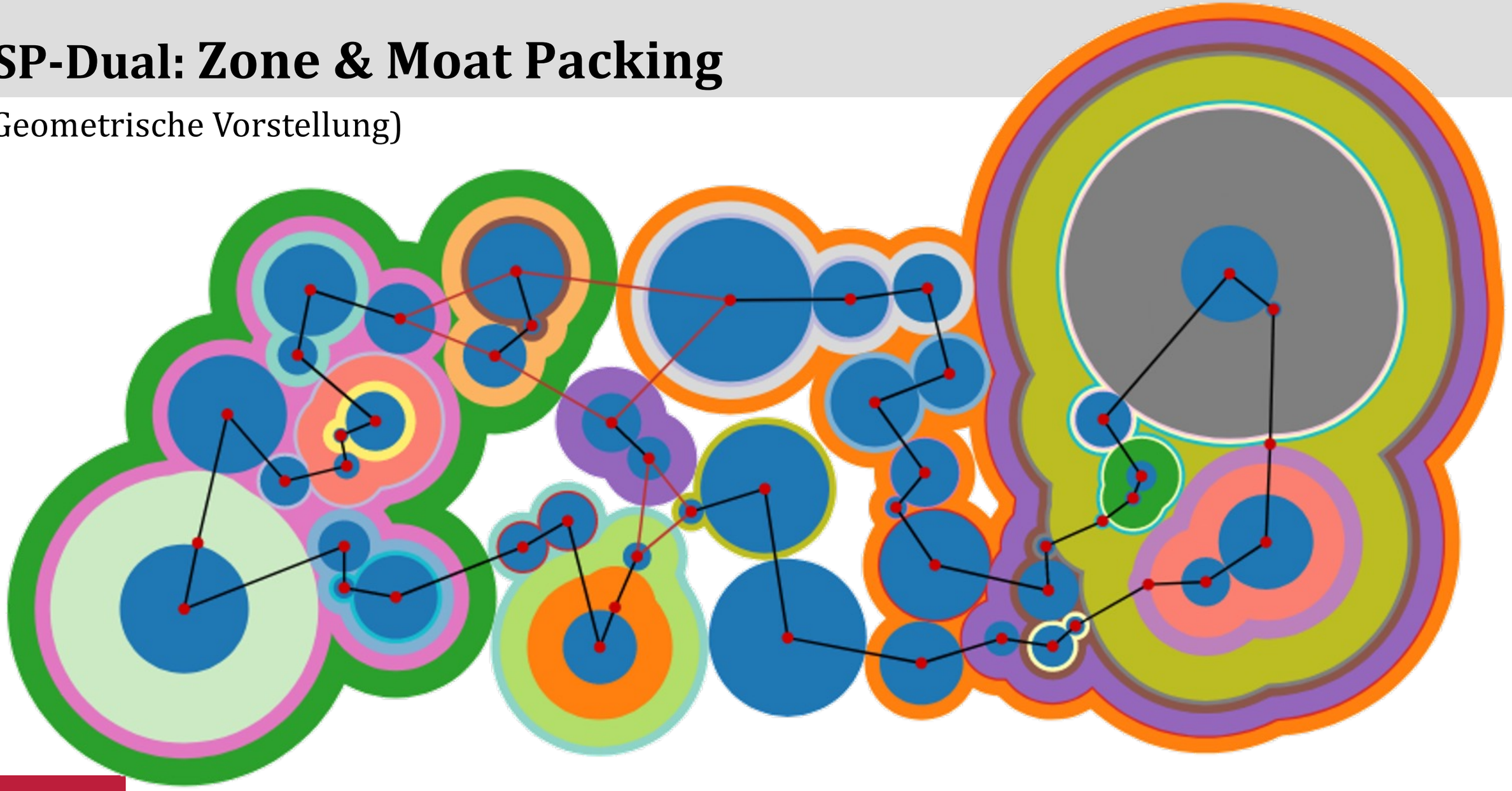
s.t.

$$y_v + y_w + \sum_{S : \{v,w\} \in \delta(S)} y_S - y_{\{v,w\}} \leq c_e, \qquad \forall \{v, w\} \in E$$

$$y_v \text{ free}, y_e, y_S \geq 0$$

TSP-Dual: Zone & Moat Packing

(Geometrische Vorstellung)



Allgemeine Cut-Repräsentation für TSP

$$\min \sum_{e \in E} c_e x_e$$

s.t.

$$\sum_{e \in \delta(p)} x_e = 2, \qquad \forall p \in P$$

$$\sum_{e \in \delta(S)} x_e \geq 2, \qquad \forall \emptyset \neq S \setminus \subsetneq P$$

$$0 \leq x_e \leq 1$$

Jeder Cut (z.B. Subtour, Comb) für TSP kann allgemein so festgehalten werden:

- Sei $\mathcal{F} = \{S_1, \dots, S_j\}$ eine Menge von Knotenmengen.
- $\mu \in \mathbb{N}$

Dann kann ein Cut folgendermaßen dargestellt werden:

$$\sum_{S \in \mathcal{F}} \sum_{e \in \delta(S)} x_e \geq \mu$$

Für

- Subtours: $\mathcal{F} = \{S\}$ und $\mu = 2$
- Combs: $\mathcal{F} = \{H, T_1, \dots, T_k\}$ und $\mu = 3k + 1$

Generalisiertes Duale LP

Sei nun

- $e(\mathcal{F}) = \bigcup_{S \in \mathcal{F}} \delta(S)$ die Menge von Kanten, die eine Knotenmenge in $\mathcal{F}$ verlässt.
- $\chi(e, \mathcal{F})$ die Anzahl an Knotenmengen, die e verlässt.

Dann ist das duale LP wie folgt:

$$\max 2 \sum_{p \in P} y_p + 2 \sum_{S} y_S - \sum_{e \in E} y_e$$

s.t.

$$y_v + y_w + \sum_{\mathcal{F}: \{v,w\} \in \delta(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} - y_{\{v,w\}} \leq c_e, \qquad \forall \{v, w\} \in E$$

$$y_v \text{ free}, y_e, y_S \geq 0$$

Suche nicht-betrachtete Kante e mit

$$\alpha_e = c_e - y_v - y_w - \sum_{\mathcal{F}: \{v,w\} \in \delta(\mathcal{F})} \chi(e, \mathcal{F}) z_{\mathcal{F}} + y_e < 0$$

Kostenabschätzung

Suche nicht-betrachtete Kante e mit $= 0$, da $x = 0 < 1$.

$$\alpha_e = c_e - y_v - y_w - \sum_{\mathcal{F}:\{v,w\}\in\delta(\mathcal{F})} \chi(e,\mathcal{F}) z_{\mathcal{F}} + y_e < 0$$

Das Berechnen von

$$\sum_{\mathcal{F}:\{v,w\}\in\delta(\mathcal{F})} \chi(e,\mathcal{F}) z_{\mathcal{F}}$$

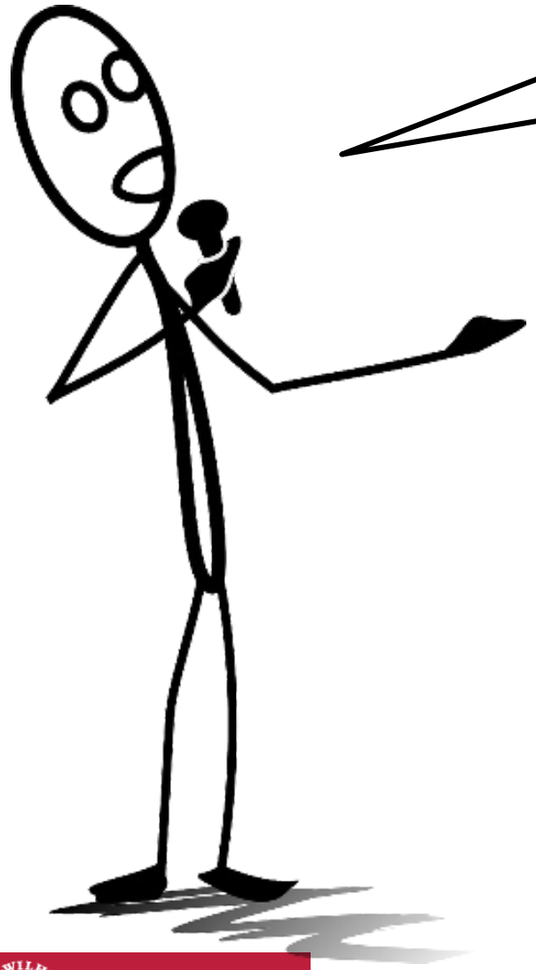
ist allerdings zeitaufwändig.

Bestimme ein $\bar{\alpha}_e \leq \alpha_e$.
Betrachte für eine Kante $e = \{v, w\} \in E$

$$\bar{y}_v := y_v + \sum_{\mathcal{F}} \sum_{S\in\mathcal{F}:\, v\in S} z_{\mathcal{F}} \qquad \text{und} \qquad \bar{\alpha}_e = c_e - \bar{y}_v - \bar{y}_w \leq \alpha_e$$

Zähle also nur wie oft die Endknoten in den Mengen vorkommen. Dadurch wird ignoriert, dass die Kante innerhalb einer Knotenmenge S verlaufen kann.

Schlusswort



TSP lässt sich für große Instanzen lösen. Nutze dazu LP/IP Techniken!

Viele Solver können das nur beschränkt umsetzen.

Lazy Constraints und Cutting Planes können in der Regel einfach hinzugefügt werden.

Pricing kann durch wiederholtes lösen umgesetzt werden.

Branch-Cut-and-Price ist nicht in allen Solvern möglich. SCIP erlaubt es.