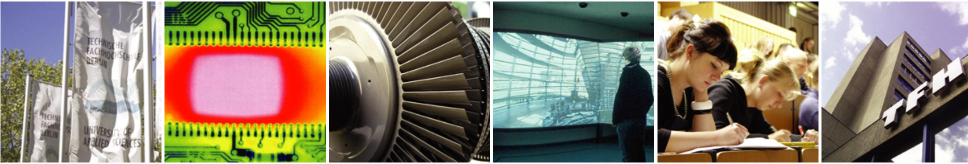
Das Rundreiseproblem

[Traveling Salesman Problem (TSP)]

Prof. Dr. Thomas Winter

Beuth Hochschule für Technik Berlin



Das (symmetrische) Traveling Salesman Problem (TSP)

Ein Handlungsreisender soll n verschiedenen Städten jeweils genau einen Kunden besuchen. Zu je zwei Städten ist die Entfernung zwischen diesen beiden Städten angegeben.
Gesucht ist eine **Rundreise**, bei der der Handlungsreisende jede Stadt genau einmal besucht und zu seinem Ausgangspunkt zurückkehrt, wobei er eine **Wegstrecke mit minimaler Länge** zurücklegen soll.

Graphentheoretische Modellierung:

- Wir definieren einen Graphen, dessen n Knoten den Städten entsprechen.
- Zwischen je zwei Knoten gibt es eine Kante, d.h. wir betrachten den vollständigen Graphen K_n.
- Jeder Kante ist $\{i, j\}$ ein Gewicht $w(i, j) = w(j, i)$ zugeordnet, wobei $w(i, j) \geq 0$ der Entfernung zwischen den Städten entspricht, die den Knoten i und j entsprechen.

Gesucht: Kreise, die jeden Knoten durchlaufen und minimales Gesamtgewicht haben

Ein Handlungsreisender soll n verschiedenen Städten jeweils genau einen Kunden besuchen. Zu je zwei Städten ist die Entfernung zwischen diesen beiden Städten angegeben.
Gesucht ist eine **Rundreise**, bei der der Handlungsreisende jede Stadt genau einmal besucht und zu seinem Ausgangspunkt zurückkehrt, wobei er eine **Wegstrecke mit minimaler Länge** zurücklegen soll.

Graphentheoretische Modellierung:

- Wir definieren einen Graphen, dessen n Knoten den Städten entsprechen.
- Zwischen je zwei Knoten gibt es eine Kante, d.h. wir betrachten den vollständigen Graphen K_n.
- Jeder Kante ist $\{i, j\}$ ein Gewicht $w(i, j) = w(j, i)$ zugeordnet, wobei $w(i, j) \geq 0$ der Entfernung zwischen den Städten entspricht, die den Knoten i und j entsprechen.

Gesucht: Kreise, die jeden Knoten durchlaufen und minimales Gesamtgewicht haben

Definition (Traveling Salesman Problem (TSP))

Gegeben sei der vollständige, gewichtete Graph K_n mit der Knotenmenge $V = \{1, 2, \ldots, n\}$ und der Kantenmenge $E = V \times V$.

Gesucht ist eine teilzyklenfreie Permutation $\pi : V \to V$ der Knoten in V, also $1, 2, \ldots, n$, mit $1 \mapsto \pi(1), 2 \mapsto \pi(2), \ldots, n \mapsto \pi(n)$ mit minimalem Gesamtgewicht $w(\pi)$, das gegeben ist durch

$$w(\pi) = \sum_{i=1}^{n} w(i, \pi(i))$$

Die Permutation gibt die Reihenfolge an, in der die Städte besucht werden: $\pi(i)$ wird nach i besucht. Dabei darf man erst, nachdem alle Knoten besucht wurden, zum Ausgangsknoten zurückkehren.
Teilzyklen sind daher nicht erlaubt. Die Kanten (Schlingen) (i, i) können daher auch für alle $i \in V$ ausgeschlossen werden.
Zur Vereinfachung setzen wir $w(i, i) = M$ mit $M > \sum_{i \in V} \sum_{j \in V, j \neq i} w(i, j)$.

Nearest-Neighbour-Heuristik

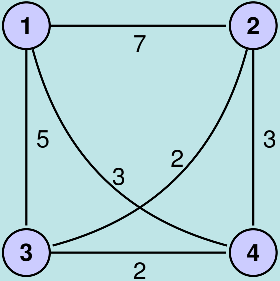
1. Starte mit einem beliebigen Knoten v, z. B. mit Knoten $v = 1$.
   Markiere v als besucht. Setze $w = 0$ und $i = v$.
2. Bestimme für i den nächstgelegenen noch nicht besuchten Knoten j, d.h. j sei so gewählt, dass

$$w(i,j) = \min\{w(i,k) \mid k \in V\}$$

   Markiere j und setze $i = j$. Setze $w(\pi) = w(\pi) + w(i,j)$.
3. Sind alle Knoten markiert, füge zur Rundreise die Kante von i zum Startknoten v hinzu. Setze $w(\pi) = w(\pi) + w(i,v)$.
   Anderenfalls gehe zu 2.)

Die Nearest-Neighbour-Heuristik liefert in $O(n^2)$ Schritten eine heuristische Lösung, die in der Regel nicht optimal ist.

Beispiel

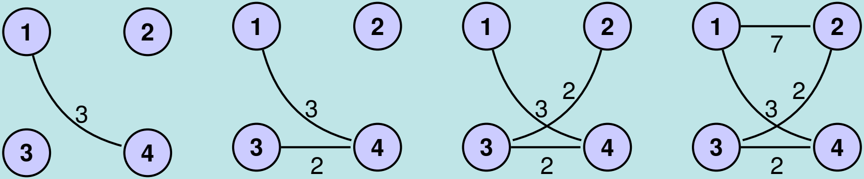


$$W = \begin{pmatrix} M & 7 & 5 & 3 \\ 7 & M & 2 & 3 \\ 5 & 2 & M & 2 \\ 3 & 3 & 2 & M \end{pmatrix}$$

Optimale Tour
$1 - 4 - 3 - 2 - 1$

i	1	2	3	4
$\pi(i)$	4	1	2	3

$$w(\pi) = 3+2+2+7 = 14$$



Farthest-Insert-Heuristik

1. Starte mit einem beliebigen Knoten v, z. B. mit Knoten $v = 1$. Wähle den Knoten j mit $w(v,j) = \max\{w(v,i) \mid i \in V \setminus \{v\}\}$. Bilde die Teiltour $(v,j),(j,v)$. Setze $\pi(v) = j$ und $\pi(j) = v$ sowie $w(\pi) = w(v,j) + w(j,w)$ und $V' = \{v,j\}$.
2. Bestimme für alle Knoten $i \notin V'$ die minimale Distanz von i zur aktuellen Teiltour
$$w(i,V') = \min\{w(i,j) \mid j \in V'\}$$
3. Wähle $k \notin V'$ so, dass
$$w(k,V') = \max\{w(i,V') \mid i \notin V'\}$$

Farthest-Insert-Heuristik

4. Berechne für alle Kanten (i, j) der aktuellen Teiltour

$$\delta_{ij} = w(i,k) + w(k,j) - w(i,j)$$

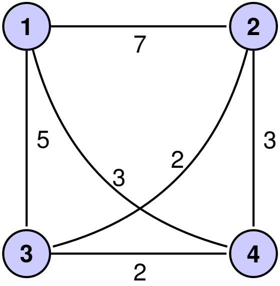
und füge k zwischen die Knoten i und j in die aktuelle Teiltour ein, für die δ_{ij} minimal wird, d.h.

$$(v, \pi(v)), \ldots (\pi^{-1}(i), i), (i,k), (k,j), (j, \pi(j)), \ldots, (\pi^{-1}(v), v)$$

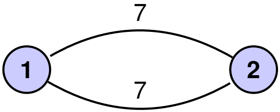
Setze $\pi(i) = k, \pi(k) = j$. Füge k zu V' hinzu und setze $w(\pi) = w(\pi) + \delta_{ij}$.

5. Ist $V' = V$ stop. Sonst gehe zu 2.)

In jedem Schritt nimmt man somit den entferntesten, noch nicht besuchten Knoten in die Teiltour auf.
Die Farthest-Insert-Heuristik liefert in $O(n^2)$ Schritten eine heuristische Lösung, die in der Regel nicht optimal ist.



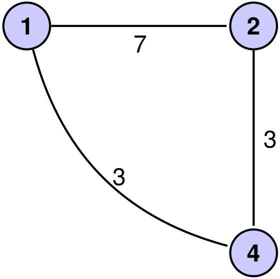
$k = 2$ mit weitester Entfernung von 1
(Sub-)Tourlänge:
$7 + 7 = 14$



$j \backslash i$	3	**4**
1	5	3
2	2	3
$w(i, V')$	2	**3**

größte Mindestentfernung zu $k = 4$

$k = 4$
zwischen 1 und 2:
$\delta_{12} = 3 + 3 - 7 = -1$
(Sub-)Tourlänge:
$14 + (-1) = 13$

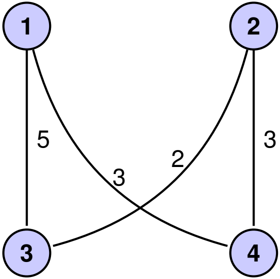


verbleibender Knoten
$k = 3$

$k = 3$ zwischen

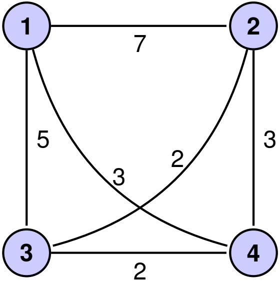
- 1 und 4:
  $\delta_{14} = 5 + 2 - 3 = 4$
- 4 und 2:
  $\delta_{42} = 2 + 2 - 3 = 1$
- 2 und 1:
  $\delta_{21} = 2 + 5 - 7 = 0$

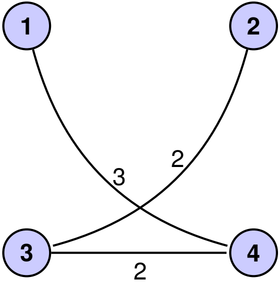
also zwischen 2 und 1
Tourlänge: 13 (zufällig optimal)

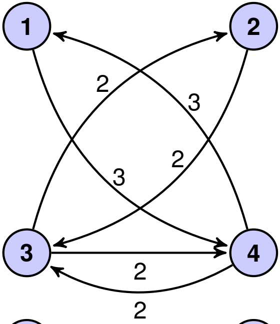


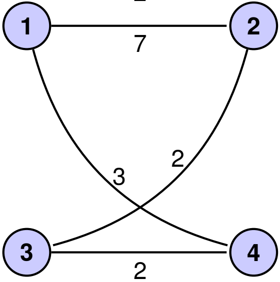
Spanning-Tree-/ Minimum-Tree-Approximation:

1. Man bestimme einen minimal spannenden Baum.
2. Man verdopple die Kanten des Baums.
3. Man bestimme einen **geschlossenen Euler-Zug** C
   (= Kantenzug, der jede Kante genau einmal durchläuft)
   (z.B. mit dem Algorithmus von Fleury)
   und gebe ihm eine Orientierung.
4. Man wähle einen Startknoten i setze $p = i$ und $T = \emptyset$.
5. Sind alle Knoten markiert, dann setze $T := T \cup \{(p, i)\}$ und STOP (T ist eine Tour).
6. Man laufe von p entlang der Orientierung von C, bis ein unmarkierter Knoten q erreicht ist.
   Setze $T := T \cup \{(p, q)\}$, markiere q, setze $p := q$ und gehe zu 5.
   (Man sucht also eine „Abkürzung“, wenn man auf einen schon besuchten Knoten trifft.)









4. $p=1,\ T=\emptyset$
5. nicht alle markiert
6. $q=4,\ T=\{(1,4)\},\ p=4$
5. nicht alle markiert
6. $q=3,\ T=\{(1,4),(4,3)\},\ p=3$
5. nicht alle markiert
6. $q=2,\ T=\{(1,4),(4,3),(3,2)\},\ p=2$
5. alle markiert,
   $T=\{(1,4),(4,3),(3,2),(2,1)\}$.

Definition

Ein **(geschlossener) Euler-Zug** ist ein geschlossener Kantenzug, der jede Kante des Graphen genau einmal enthält.

Algorithmus von Fleury (zur Konstruktion eines Euler-Zugs (sofern vorhanden))

1. Starte in einem beliebigen Knoten.
2. Wähle einen benachbarten Knoten und entferne die dabei durchlaufene Kante, wobei der übrigbleibende Restgraph zusammenhängend bleiben muss.
3. Entferne den Anfangsknoten der Kante, falls dieser nun isoliert, aber nicht der Startknoten ist.
   Ist das nicht möglich: STOP: Kein Euler-Zug vorhanden.
4. Besteht der Restgraph nur noch aus dem Anfangsknoten: STOP: Euler-Zug gefunden.
   Ansonsten: gehe zu 2.

Die entfernten Kanten ergeben in der Reihenfolge ihrer Entfernung einen Euler-Zug.

Ein symmetrisches TSP heißt **euklidisch**, wenn für die Kantengewichte die Dreiecksungleichung gilt:

$$w(i,k) \leq w(i,j) + w(j,k)$$

Heuristik	Güte ($\varepsilon \cdot 100$ % Fehler)	Laufzeit
Nearest Neighbour	$\frac{1}{2}(\lceil \log n \rceil - 1)$	n^2
Nearest Insert	1	n^2
Farthest Insert		n^2
Cheapest Insert	1	n^2
Spanning Tree	1	$n^2 \log n$
Christofides	$\frac{1}{2}$	n^3

Gütegarantie ε bedeutet: $w(T_{opt}) \leq w(T_H) \leq (1+\varepsilon)w(T_{opt})$

wobei T_{opt} eine optimale Tour ist und $w(T_{opt})$ ihr Gesamtgewicht sowie T_H eine mit Heuristik ermittelte Tour und $w(T_H)$ deren Gesamtgewicht.

Wichtige Algorithmenklasse in der Kategorie der Verbesserungsverfahren:

Austauschverfahren

Beispiel: **2-OPT: Zweier-Austausch**

Gegeben:

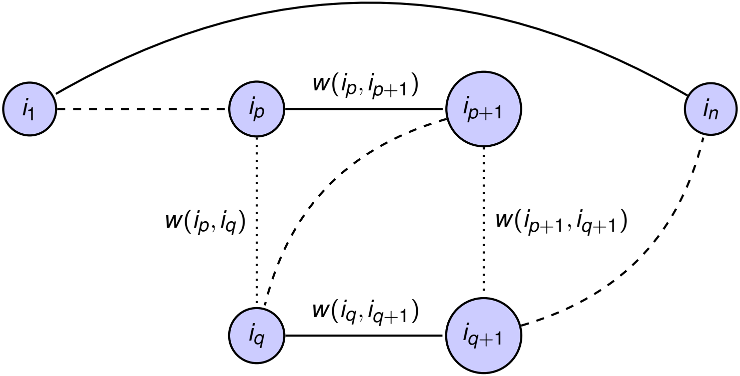
- eine Anfangstour mit der Kantenmenge
  $T = \{(i_1, i_2), (i_2, i_3), \cdots, (i_{n-1}, i_n)\}$

1. Setze
   $K := \{\{(i_p, i_{p+1}), (i_q, i_{q+1})\} \mid p+1 \neq q, p \neq q, q+1 \neq p, 1 \leq p, q \leq n\}$
2. Für alle Kantenpaare $\{(i_p, i_{p+1}), (i_q, i_{q+1})\} \in K$ führe aus:

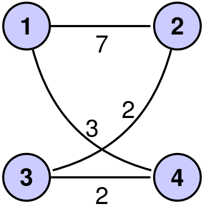
   Ist $w(i_p, i_{p+1}) + w(i_q, i_{q+1}) > w(i_p, i_q) + w(i_{p+1}, i_{q+1})$, dann setze
   $T := (T \setminus \{(i_p, i_{p+1}), (i_q, i_{q+1})\}) \cup \{(i_p, i_q), (i_{p+1}, i_{q+1})\}$

   und gehe zu Schritt 1.

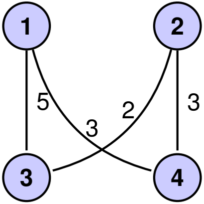
3. Gib T aus.



Falls $w(i_p, i_q) + w(i_{p+1}, i_{q+1}) < w(i_p, i_{p+1}) + w(i_q, i_{q+1})$ ist,
dann plane die Tour um: Aus
$i_1 - \ldots - i_p - i_{p+1} - \ldots - i_q - i_{q+1} - \ldots i_n - i_1$ wird dann
$i_1 - \ldots - i_p - i_q - \ldots - i_{p+1} - i_{q+1} - \ldots i_n - i_1$.



Alte Tour $1 - 4 - 3 - 2 - 1$ mit Länge 14
$i_p = 4$, $i_{p+1} = 3$, $i_q = 2$, $i_{q+1} = 1$
Die alte Verbindung hat die Länge: $w(i_p, i_{p+1}) + w(i_q, i_{q+1}) = 2 + 7 = 9$
Die neue Verbindung ist kürzer: $w(i_p, i_q) + w(i_{p+1}, i_{q+1}) = 3 + 5 = 8$
Neue Tour: $1 - 4 - 2 - 3 - 1$ mit Länge 13



Im Zuordnungsproblem müssen je n Objekte aus verschiedenen Klassen S und T einander zugeordnet werden.

Anwendungen sind z.B.:

- im Betrieb: Zuordnung von Aufgaben zu Personal
- im Computer: Zuordnung von Ressourcen zu Jobs (Tasks)
- im Leben: Zuordnung von Partnern
- ...

Graphentheoretische Modellierung:

Gegeben: vollständiger, bipartiter Graph $G = (S \dot{\cup} T, S \times T)$,
Kantengewichte $w : S \times T \to \mathbb{R}_0^+$.
Gesucht: eine Zuordnung von S zu T, d.h. Auswahl entsprechender Kanten zwischen S und T, so dass die Summe der Kantengewichte minimal ist

Das Zuordnungsproblem lässt sich auch

- als Spezialfall des Transportproblems mit $a_i = b_j = 1$ für alle $i \in S$ und $j \in T$
- als Spezialfall eines Netzwerkflussproblems
- mittels Permutationen der Menge $\{1, 2, \ldots, n\}$, d.h. Zuordnung der n Elemente von S zu den
  n Elementen von T

beschreiben.

$\Rightarrow$ Eine Lösung des Zuordnungsproblems ergibt eine untere Schranke für das zugehörige Rundreiseproblem. – Warum?

Weil eine Lösung des Zuordnungsproblems eine beliebige Permutation der Zahlen $\{1, 2, \ldots, n\}$ ist, während als Lösung des Rundreiseproblems nur zyklische Permutationen in Frage kommen.

Das zugehörige (ganzzahlige) lineare Programm lautet:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_{ij}$$

$$\begin{array}{llcll} \text{s.t.} & \sum_{i=1}^{n} x_{ij} & = & 1 & \text{für alle } j \in T \\ & \sum_{j=1}^{n} x_{ij} & = & 1 & \text{für alle } i \in S \\ & x_{ij} \in \{0,1\} & & & \end{array}$$

Das zugehörige duale lineare Programm ist:

$$\begin{array}{llcll} \max & \sum_{j=1}^{n} v_j - \sum_{i=1}^{n} u_i & & & \\ \text{unter} & v_j - u_i & \leq & w_{ij} & \text{für alle } i \in S, j \in T \end{array}$$

(Achtung: Das LP ist ein Minimierungsproblem. Zur Dualisierung wurde der erste Satz an Restriktionen mit -1 multipliziert)

Aus der Dualität ergibt sich: $x_{ij} > 0 \quad \Leftrightarrow \quad v_j - u_i = w_{ij}$.

0. Initialisierung: $M = \emptyset$, $u_i = v_j = 0$ für alle $i,j = 1,2,\ldots,n$. Setze $l = 1$. $\bar{J} = \emptyset$.
1. Konstruiere Hilfsgraph $G_l = (S \cup T, E_B \cup E_R)$ mit

$$\begin{aligned} E_B &= \{(i,j) \in E \setminus M \mid i \in S, j \in T\} \\ E_R &= \{(j,i) \in M \mid i \in S, j \in T\} \end{aligned}$$

mit Kantengewichten

$$\tilde{w}(i,j) = \begin{cases} \bar{w}(i,j) = w(i,j) + u_i - v_j & (i,j) \in E_B \\ 0 & (i,j) \in E_R \end{cases}$$

Berechne

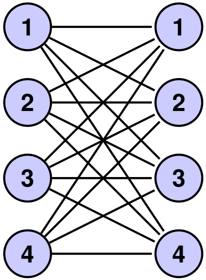
$$\bar{w}(M) = \sum_{(i,j)\in M} \bar{w}(i,j) = \sum_{(i,j)\in M} w(i,j) + \sum_{i=1}^{n} u_i - \sum_{j=1}^{n} v_j$$

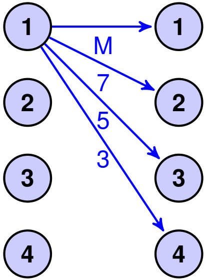
2. Suche einen kürzesten Weg W in G_l von l nach $J = T \setminus \bar{J}$, z. B. mit dem Dijkstra-Algorithmus. Stoppe den Dijkstra-Algorithmus, sobald ein Knoten $k \in J$ erreicht wird. Die resultiernden kürzesten Entfernungen seien $\tilde{u}_i$ für $i \in S$ und $\tilde{v}_j$ für $j \in T$.
3. Passe M, u und v an:

$$\begin{aligned} M &= M \setminus \{(i,j) | (j,i) \in W \cap E_R\} \cup \{(i,j) | (i,j) \in W \cap E_B\} \\ u_i &= u_i + \min\{\tilde{u}_i, \tilde{v}_k\} \\ v_j &= v_j + \min\{\tilde{v}_j, \tilde{v}_k\} \end{aligned}$$

4. Füge k zu $\bar{J}$ hinzu.
5. Falls $l = n$: STOP: optimale Zuordnung ist M.
   Anderenfalls setze $l = l + 1$. Gehe zu 1.)

Das Zuordnungsproblem: primal-dualer Algorithmus nach Lawler





$$
\begin{aligned}
W &= \begin{pmatrix} M & 7 & 5 & 3 \\ 7 & M & 2 & 3 \\ 5 & 2 & M & 2 \\ 3 & 3 & 2 & M \end{pmatrix} \\
M &= \emptyset \\
l &= 1 \\
J &= \emptyset
\end{aligned}
$$

Kürzester Weg von $1 \in S$ nach $4 \in T$

$k = 4$, $\tilde{u} = (0, \infty, \infty, \infty)^T$, $\tilde{v} = (M, 7, 5, 3)^T$

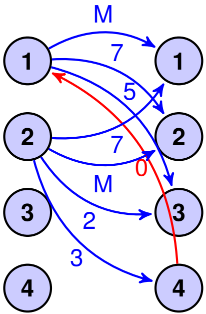
$v_k = 3$.

$M = \{(1,4)\}$.

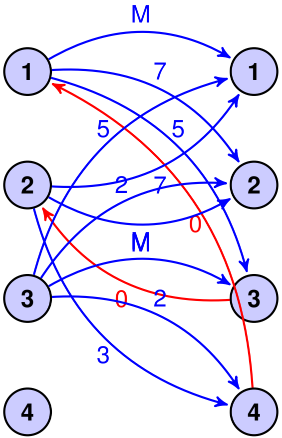
$u = (0,3,3,3)$, $v = (3,3,3,3)$, $(\bar{w}(M) = 0)$

$l = 2$

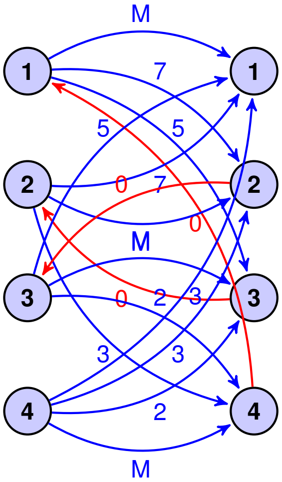
$E_B = E \setminus \{(1,4)\}$, $E_R = \{(4,1)\}$.



Kürzester Weg von $l = 2 \in S$ nach $3 \in J = \{1,2,3\}$
$k = 3$, $\tilde{u} = (3,0,\infty,\infty)^T, \tilde{v} = (7,10,2,3)^T$
$v_k = 2$
$M = \{(1,4),(2,3)\}$.
$u = (2,3,5,5)$, $v = (5,5,5,5)$, $(\bar{w}(M) = 0)$
$l = 3$
$E_B = E \setminus \{(1,4),(2,3)\}$, $E_R = \{(4,1),(3,2)\}$.



Kürzester Weg von $l = 3 \in S$ nach $2 \in J = \{1,2\}$
$k = 2$, $\tilde{u} = (2,7,0,\infty)^T, \tilde{v} = (5,2,5,2)^T$
$v_k = 2$
$M = \{(1,4),(2,3),(3,2)\}$.
$u = (4,5,5,7)$, $v = (7,7,7,7)$, $(\bar{w}(M) = 0)$
$l = 4$
$E_B = E \setminus \{(1,4),(2,3),(3,2)\}$, $E_R = \{(4,1),(3,2),(2,3)\}$.



Kürzester Weg von $l = 4 \in S$ nach $1 \in J = \{1\}$
$k = 1$, $\tilde{u} = (M, 2, 3, 0)^T, \tilde{v} = (3, 3, 2, M)^T$
$v_k = 3$
$M = \{(1,4),(2,3),(3,2),(1,4)\}$.
$u = (7,7,8,7)$, $v = (10,10,9,10)$
$\bar{w}(M) = w(M) + \sum u_i - \sum v_j = 10 + 29 - 39 = 0$
$w(M) = 10$. Optimale Zuordnung: $\pi(1) = 4, \pi(2) = 3, \pi(3) = 2, \pi(4) = 1$

1. Initialisierung: $x_{ij} = 0$, $v_j = 0$, $u_i = 0$ für alle $i, j \in \{1, 2, \dots, n\}$
2. Start: Wähle $x_{1t} = 1$ mit $w_{1t} = \min\{w_{1j} | 1 \leq j \leq n\}$.
   Setze $v_t = w_{1t}$ und $u_i = w_{1t}$ für alle $i \geq 2$. (die Lösung ist dual zulässig, aber noch nicht primal zulässig) Setze $l = 2$
3. Sei $G_l = (S \times T, E_B \cup E_R)$ ein gerichteter Hilfsgraph mit
   $E_B = \{(i,j) | x_{ij} = 0\}$ und $E_R = \{(j,i) | x_{ij} = 1\}$ und
   Kantengewichten $c_{ij} = w_{ij} + u_i - v_j$ für $(i,j) \in E_B$ und $c_{ij} = 0$ für $(i,j) \in E_R$
   Bestimme in G_l mit dem Dijkstra-Verfahren einen kürzesten Weg von l zu einem noch nicht zugeordneten Knoten in
   $J = \{j | x_{ij} = 0 \text{ für alle } i\}$.
   Breche das Dijkstra-Verfahren ab, sobald ein Knoten $k \in J$ erreicht wurde mit $\bar{v}_k \leq \bar{v}_j$ für alle $j \in T$.
   Sei P der kürzeste Weg von l nach k in G_l.
   Seien $\bar{u}_i$ und $\bar{v}_j$ die berechneten Wegentfernungen für die Knoten $i \in S$ bzw. $j \in T$.

4. Setze
   - $x_{ij} = 0$ für alle $(j,i) \in P \cap E_R$ (Rückwärtskanten auf dem Weg)
   - $x_{ij} = 1$ für alle $(i,j) \in P \cap E_B$ (Vorwärtskanten auf dem Weg)

   Update der dualen Variablen mit den Dijkstra-Ergebnissen:
   - $u_i = u_i + \min(\bar{u}_i, \bar{v}_k)$ für $1 \le i \le l$
   - $u_i = u_i + \bar{v}_k$ für $l < i \le n$
   - $v_j = v_j + \min(\bar{v}_j, \bar{v}_k)$ für $1 \le j \le n$.

   Setze $l = l + 1$ und gehe zu 3.)

Der Aufwand des Algorithmus ist $O(n^3)$.

4. Setze
   - $x_{ij} = 0$ für alle $(j, i) \in P \cap E_R$ (Rückwärtskanten auf dem Weg)
   - $x_{ij} = 1$ für alle $(i, j) \in P \cap E_B$ (Vorwärtskanten auf dem Weg)

   Update der dualen Variablen mit den Dijkstra-Ergebnissen:
   - $u_i = u_i + \min(\bar{u}_i, \bar{v}_k)$ für $1 \leq i \leq l$
   - $u_i = u_i + \bar{v}_k$ für $l < i \leq n$
   - $v_j = v_j + \min(\bar{v}_j, \bar{v}_k)$ für $1 \leq j \leq n$.

   Setze $l = l + 1$ und gehe zu 3.)

Der Aufwand des Algorithmus ist $O(n^3)$.

- Zuordnung schrittweise aufbauen und dabei die Komplementaritätsbedingung erfüllen (= dual zulässig bleiben)
- jeweils den nächsten noch nicht zugeordneten Knoten aus S hinzunehmen und über den Hilfsgraphen den besten Partnerknoten in T finden, der noch keinem Knoten aus S zugeordnet ist:
  - entweder direkte (=neue) Zuordnung oder
  - indirekt mit Weg über anderen Knoten - dann muss anhand des gefundenen Weges im Hilfsgraphen die bestehende Zuordnung „umarrangiert" werden
- $\bar{u}$ und $\bar{v}$ geben die Entfernungen im Hilfsgraphen an und werden zur Berechnung von u und v benötigt, d.h. man braucht sie, um die Komplementaritätsbedingung anzupassen.
- In jedem Iterationsschritt hat man eine Lösung, die für die Mengen der zugeordneten Knoten optimal ist.

Sei $n = 3$, $S = T = \{1,2,3\}$ und $W = \begin{pmatrix} 2 & 3 & 1 \\ 7 & 2 & 3 \\ 7 & 1 & 3 \end{pmatrix}$.

- $l = 1$. $t = 3$, da $w_{1t} = w_{13} = 1$. $x_{13} = 1$. $v_3 = 1$, $u_2 = u_3 = 1$.
- $l = 2$. $C = \begin{pmatrix} 2 & 3 & 0 \\ 8 & 3 & 3 \\ 8 & 2 & 3 \end{pmatrix}$

  c_{ij} =Entfernung (kürzester Weg) vom Knoten $i \in S$ zum Knoten $j \in T$; Kosten für Rückwärtskanten von T nach S sind 0

  $P = ((2,2))$, $k = 2$, mit $\bar{u} = (3,0,\infty)$ und $\bar{v} = (8,3,3)$

  $x_{13} = 1, x_{22} = 1$,$v = (3,3,4)$, $u = (3,1,4)$
- $l = 3$. $C = \begin{pmatrix} 2 & 3 & 0 \\ 5 & 0 & 0 \\ 8 & 2 & 3 \end{pmatrix}$

  $P = ((3,2),(2,2),(2,3),(3,1),(1,1))$, $k = 1$, mit $\bar{u} = (2,2,0)$ und $\bar{v} = (4,2,2)$

  $x_{13} = 0, x_{22} = 0$, $x_{11} = 1, x_{23} = 1, x_{32} = 1$,$v = (7,5,6)$, $u = (5,3,4)$

Beispiel

Sei $n = 3$, $S = T = \{1,2,3\}$ und $W = \begin{pmatrix} 2 & 3 & 1 \\ 7 & 2 & 3 \\ 7 & 1 & 3 \end{pmatrix}$.

- $l = 1$. $t = 3$, da $w_{1t} = w_{13} = 1$. $x_{13} = 1$. $v_3 = 1$, $u_2 = u_3 = 1$.
- $l = 2$. $C = \begin{pmatrix} 2 & 3 & 0 \\ 8 & 3 & 3 \\ 8 & 2 & 3 \end{pmatrix}$

  c_{ij} =Entfernung (kürzester Weg) vom Knoten $i \in S$ zum Knoten $j \in T$; Kosten für Rückwärtskanten von T nach S sind 0

  $P = ((2,2))$, $k = 2$, mit $\bar{u} = (3,0,\infty)$ und $\bar{v} = (8,3,3)$

  $x_{13} = 1, x_{22} = 1$,$v = (3,3,4)$, $u = (3,1,4)$
- $l = 3$. $C = \begin{pmatrix} 2 & 3 & 0 \\ 5 & 0 & 0 \\ 8 & 2 & 3 \end{pmatrix}$

  $P = ((3,2),(2,2),(2,3),(3,1),(1,1))$, $k = 1$, mit $\bar{u} = (2,2,0)$ und $\bar{v} = (4,2,2)$

  $x_{13} = 0, x_{22} = 0$, $x_{11} = 1, x_{23} = 1, x_{32} = 1$,$v = (7,5,6)$, $u = (5,3,4)$

Beispiel

Sei $n = 3$, $S = T = \{1, 2, 3\}$ und $W = \begin{pmatrix} 2 & 3 & 1 \\ 7 & 2 & 3 \\ 7 & 1 & 3 \end{pmatrix}$.

- $l = 1$. $t = 3$, da $w_{1t} = w_{13} = 1$. $x_{13} = 1$. $v_3 = 1$, $u_2 = u_3 = 1$.
- $l = 2$. $C = \begin{pmatrix} 2 & 3 & 0 \\ 8 & 3 & 3 \\ 8 & 2 & 3 \end{pmatrix}$

  c_{ij} =Entfernung (kürzester Weg) vom Knoten $i \in S$ zum Knoten $j \in T$; Kosten für Rückwärtskanten von T nach S sind 0

  $P = ((2,2))$, $k = 2$, mit $\bar{u} = (3, 0, \infty)$ und $\bar{v} = (8, 3, 3)$

  $x_{13} = 1, x_{22} = 1$,$v = (3, 3, 4)$, $u = (3, 1, 4)$
- $l = 3$. $C = \begin{pmatrix} 2 & 3 & 0 \\ 5 & 0 & 0 \\ 8 & 2 & 3 \end{pmatrix}$

  $P = ((3,2),(2,2),(2,3),(3,1),(1,1))$, $k = 1$, mit $\bar{u} = (2, 2, 0)$ und $\bar{v} = (4, 2, 2)$

  $x_{13} = 0, x_{22} = 0$, $x_{11} = 1, x_{23} = 1, x_{32} = 1$,$v = (7, 5, 6)$, $u = (5, 3, 4)$

Sei $n=3$, $S=T=\{1,2,3\}$ und $W=\begin{pmatrix} 2 & 3 & 1 \\ 7 & 2 & 3 \\ 7 & 1 & 3 \end{pmatrix}$.

- $l=1$. $t=3$, da $w_{1t}=w_{13}=1$. $x_{13}=1$. $v_3=1$, $u_2=u_3=1$.
- $l=2$. $C=\begin{pmatrix} 2 & 3 & 0 \\ 8 & 3 & 3 \\ 8 & 2 & 3 \end{pmatrix}$
  c_{ij} =Entfernung (kürzester Weg) vom Knoten $i\in S$ zum Knoten $j\in T$; Kosten für Rückwärtskanten von T nach S sind 0
  $P=((2,2))$, $k=2$, mit $\bar{u}=(3,0,\infty)$ und $\bar{v}=(8,3,3)$
  $x_{13}=1, x_{22}=1$,$v=(3,3,4)$, $u=(3,1,4)$
- $l=3$. $C=\begin{pmatrix} 2 & 3 & 0 \\ 5 & 0 & 0 \\ 8 & 2 & 3 \end{pmatrix}$
  $P=((3,2),(2,2),(2,3),(3,1),(1,1))$, $k=1$, mit $\bar{u}=(2,2,0)$ und $\bar{v}=(4,2,2)$
  $x_{13}=0, x_{22}=0$, $x_{11}=1, x_{23}=1, x_{32}=1$,$v=(7,5,6)$, $u=(5,3,4)$

Aus dem Zuordnungsproblem und dem Problem des minimal aufspannenden Baumes ergeben sich für das TSP die folgenden Schranken.
Seien

- $z_{Zuordnung}$ der Wert einer optimalen Zuordnung
- z_{TSP} der Wert einer optimalen Rundreise
- z_{MST} der Wert eines minimal aufspannenden Baumes

Dann gilt

$$z_{Zuordnung} \leq z_{TSP} \leq 2 \cdot z_{MST}$$

Das heißt, dass man ein Branch-and-Bound-Verfahren für das TSP auch mit Hilfe dieser beiden verwandten Teilprobleme aufstellen kann, um schneller bessere obere und untere Schranken zu finden.

Literatur: Burkard, Dell'Amico, S. Martello: Assignment problems, SIAM, 2009

Ausgehend vom IP-Modell für das Zuordnungproblems

$$\begin{array}{llll}
\min & \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_{ij} & & \\
\text{s.t.} & \sum_{i=1}^{n} x_{ij} = 1 & & \text{für alle } j \in T \\
& \sum_{j=1}^{n} x_{ij} = 1 & & \text{für alle } i \in S \\
& x_{ij} \in \{0,1\} & &
\end{array}$$

ergibt sich das IP-Modell für das TSP durch Hinzuführen der Subtour-Elimations-Constraints

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \quad \text{für alle } S \subset V, |S| \geq 1$$

Dies bedeutet, dass jede echte, nicht-leere Teilmenge S der Knotenmenge V höchstens eine Kante weniger als die Anzahl der Knoten in S enthalten darf. Ansonsten enthält S eine Subtour/einen Teilzyklus.

Ein (einfaches) Schnittebenenverfahren für das TSP erhält man ausgehend vom IP-Modell für das Zuordnungsproblem.

Enthält die zur optimalen Zuordnung gehörende Tour Subtouren, so kann man diese explizit ausschließen durch Hinzufügen der (alternativen) Subtour-Elimations-Constraint

$$\begin{aligned} \sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} &\geq 1 \\ \sum_{i \in V \setminus S} \sum_{j \in S} x_{ij} &\geq 1 \end{aligned}$$

S ist hier die Teilmenge von V, die ein Subtour enthält. Beide Ungleichungen kann man auch addieren.

Wir betrachten wiederum dasselbe Beispiel mit

$$W = \begin{pmatrix} M & 7 & 5 & 3 \\ 7 & M & 2 & 3 \\ 5 & 2 & M & 2 \\ 3 & 3 & 2 & M \end{pmatrix}$$

Die optimale Lösung der Zuordnungsproblems lautet:
$x_{14} = x_{41} = x_{23} = x_{32} = 1$ (alle anderen $x_{ij} = 0$). Der Zielfunktionswert ist $z_{Zuordnung} = 10$.

Die Lösung beinhaltet 2 Teilzyklen: $1 - 4 - 1$, $2 - 3 - 2$

Die Subtour-Eliminations-Bedingung für die 1. Teiltour ist: $x_{14} + x_{41} \leq 1$
Das Hinzufügen zum LP des Zuordnungsproblems ergibt die optimale Lösung: $x_{13} = x_{32} = x_{24} = x_{41} = 1$ mit Zielfunktionswert 13.