Technische Universität Braunschweig

Algorithmen und Datenstrukturen 2 – Übung #3

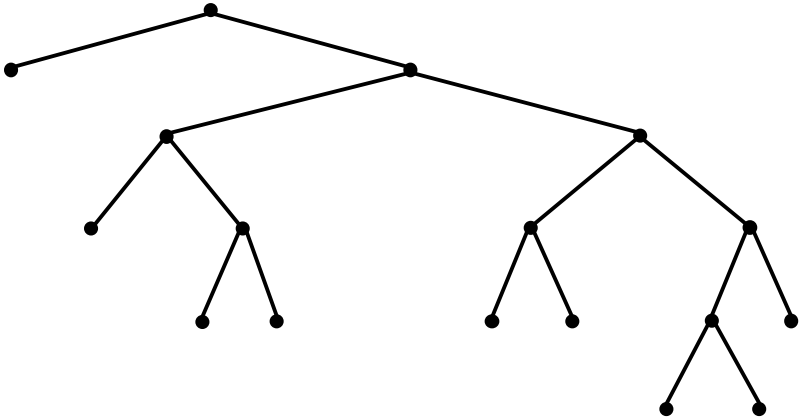
Branch-and-Bound, Euklidisches TSP

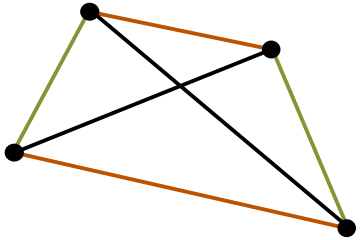
Arne Schmidt

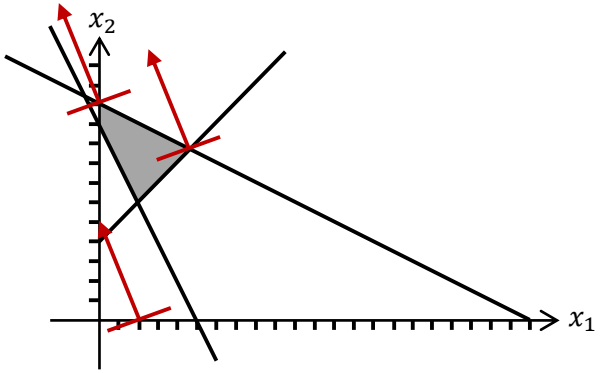
23.05.2019

Heute

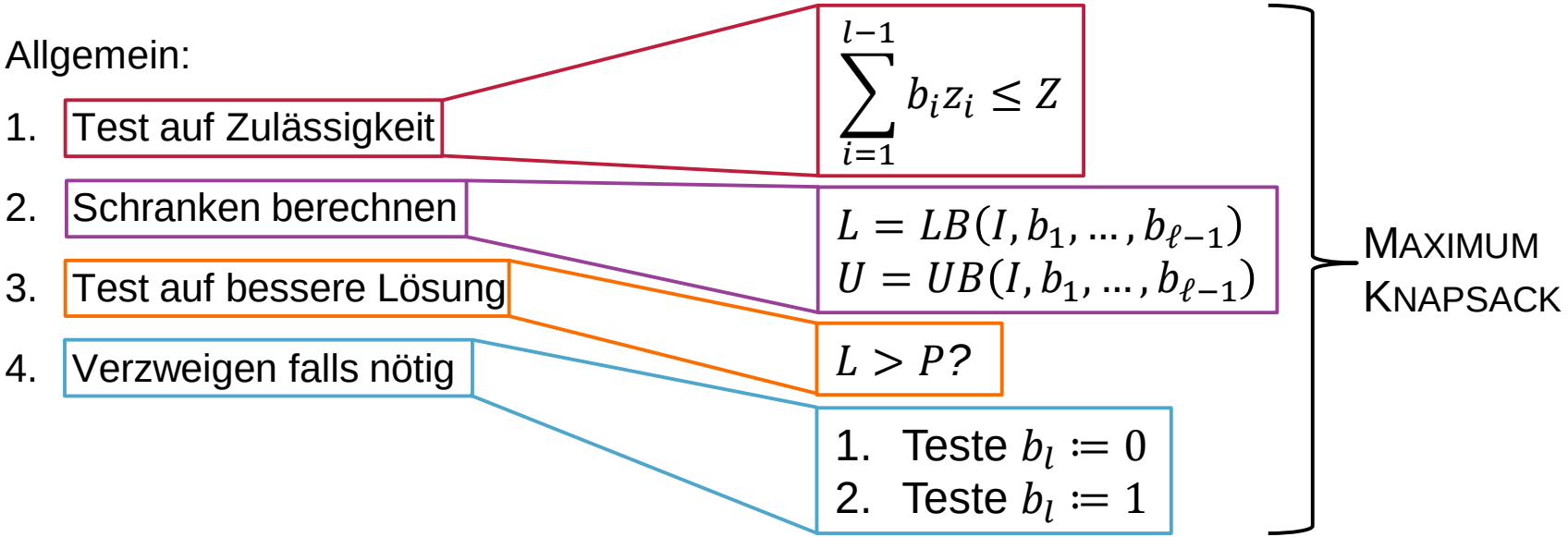
- Branch-and-Bound für Knapsack
- Euklidisches TSP
  - Definition
  - Schranken
  - Branch-and-Bound
  - Ausblick: Linear Programming



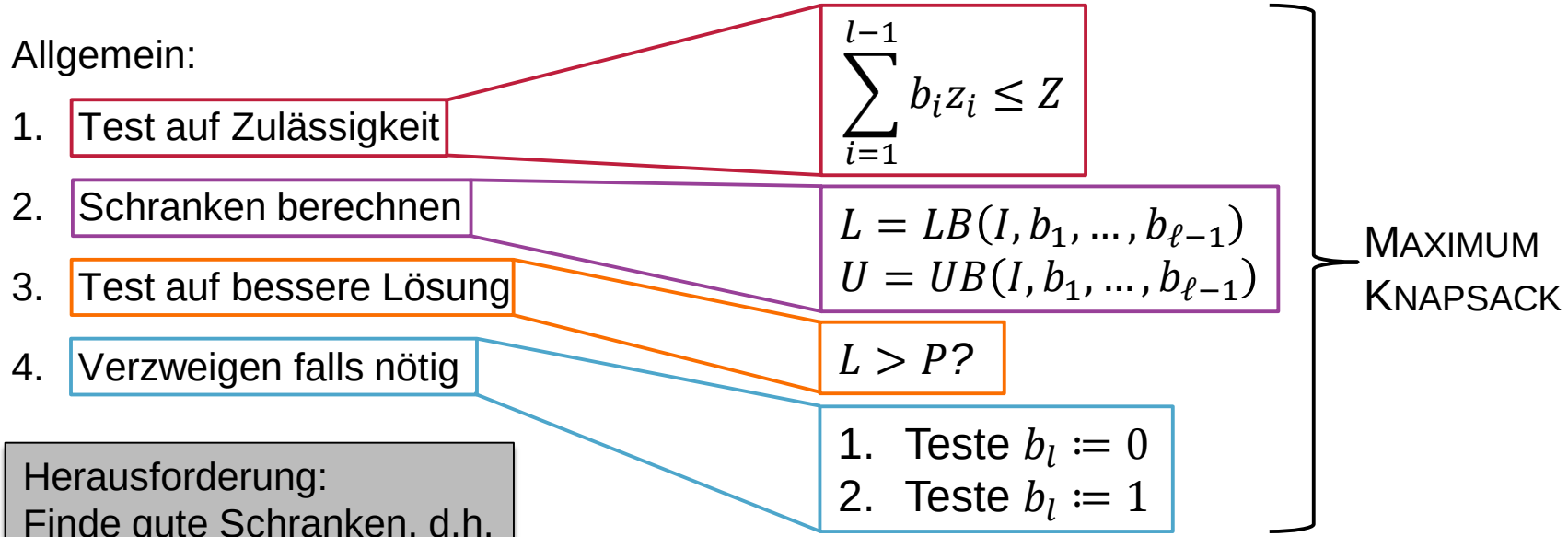




Branch-and-Bound



Branch-and-Bound



Herausforderung:
Finde gute Schranken, d.h. finde **kleine obere** und **große untere** Schranken.

Branch-and-Bound – Beispiel

i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert

Branch-and-Bound – Beispiel

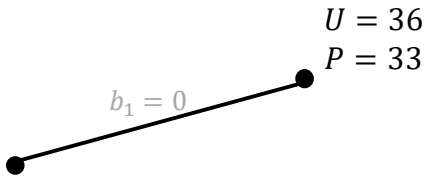
$U = 36$
$P = 33$

i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
$\{1,2,3\}$	33

Branch-and-Bound – Beispiel

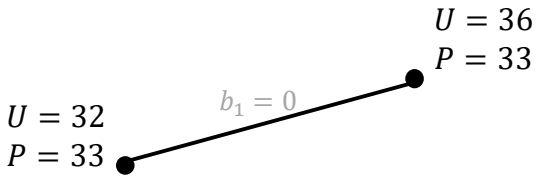


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
$\{1,2,3\}$	33

Branch-and-Bound – Beispiel

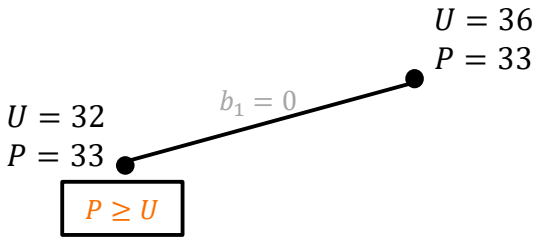


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33

Branch-and-Bound – Beispiel

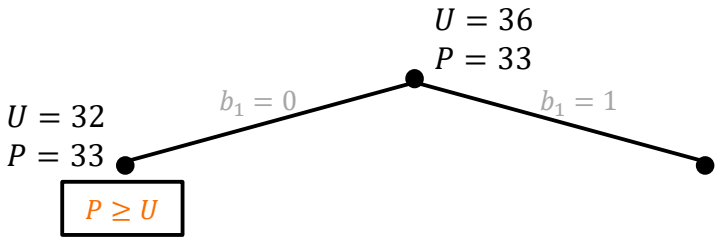


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33

Branch-and-Bound – Beispiel

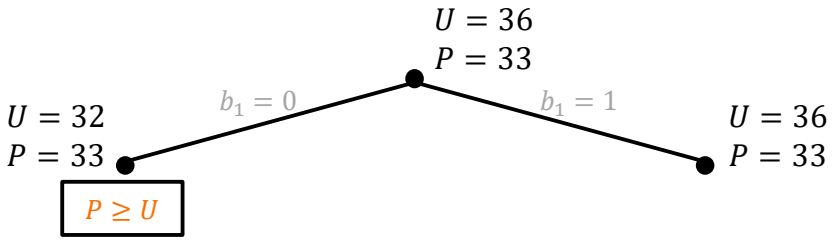


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33

Branch-and-Bound – Beispiel

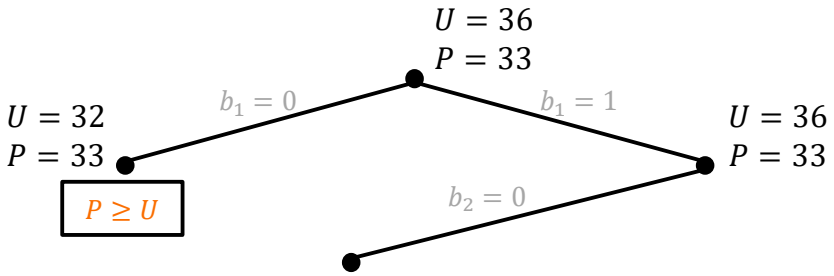


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33

Branch-and-Bound – Beispiel

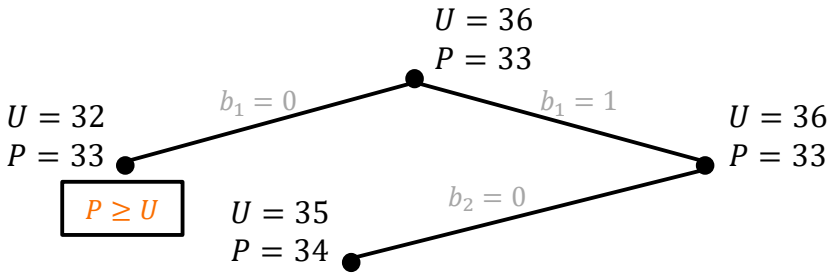


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33

Branch-and-Bound – Beispiel

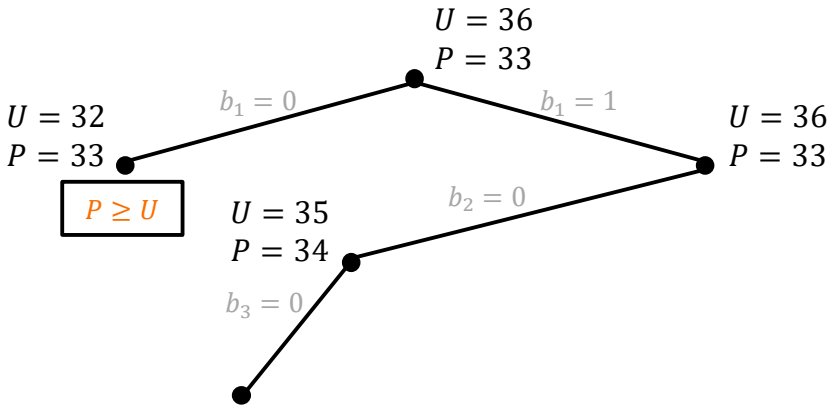


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
$\{1,2,3\}$	33
$\{1,3,5\}$	34

Branch-and-Bound – Beispiel

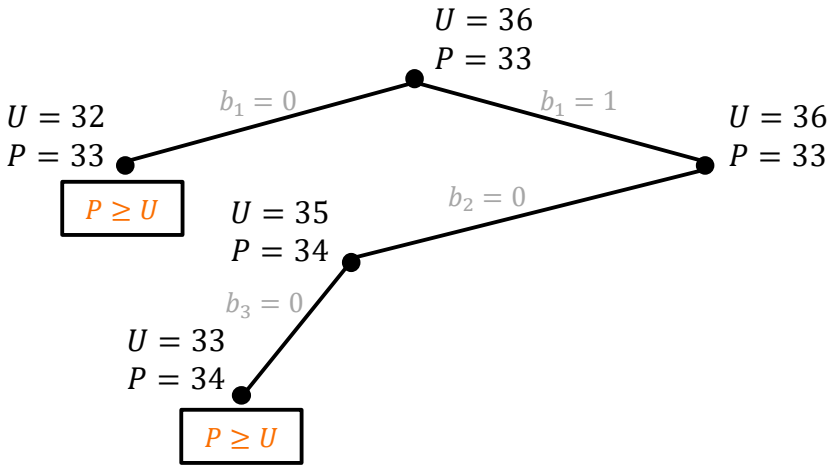


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
$\{1,2,3\}$	33
$\{1,3,5\}$	34

Branch-and-Bound – Beispiel

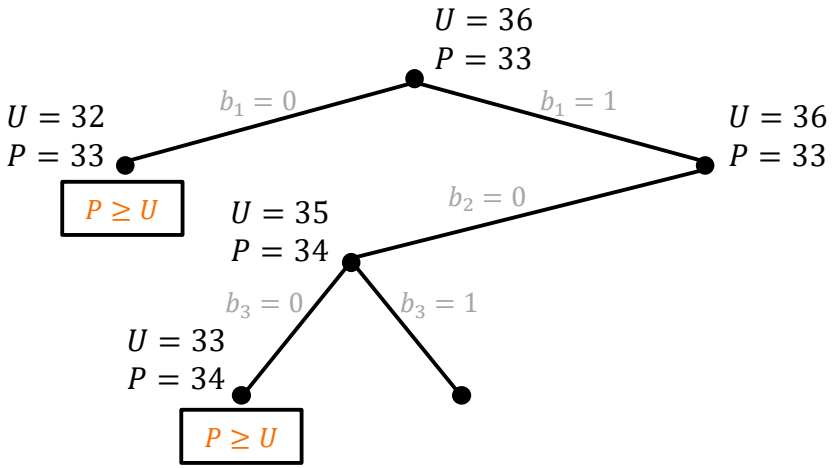


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

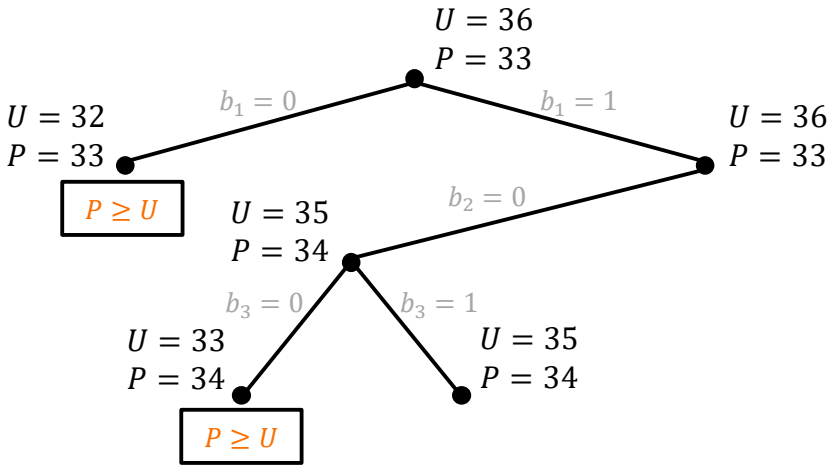


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
$\{1,2,3\}$	33
$\{1,3,5\}$	34

Branch-and-Bound – Beispiel

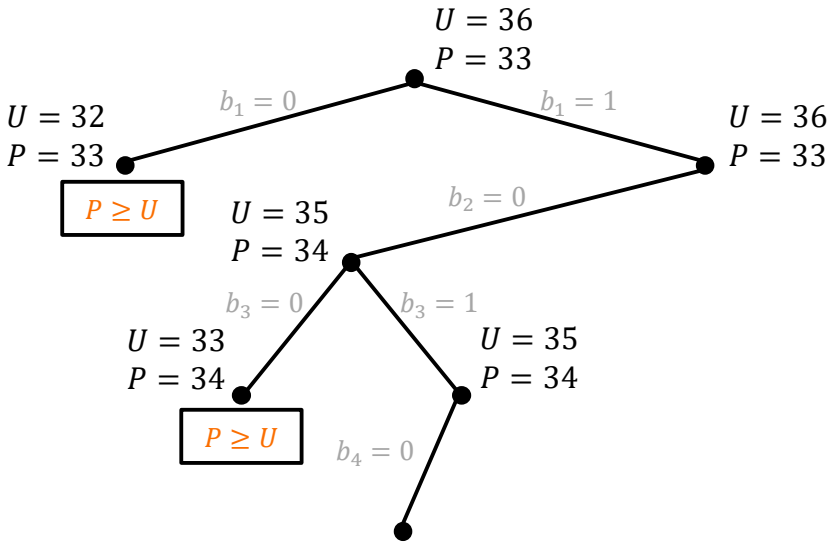


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
$\{1,2,3\}$	33
$\{1,3,5\}$	34

Branch-and-Bound – Beispiel

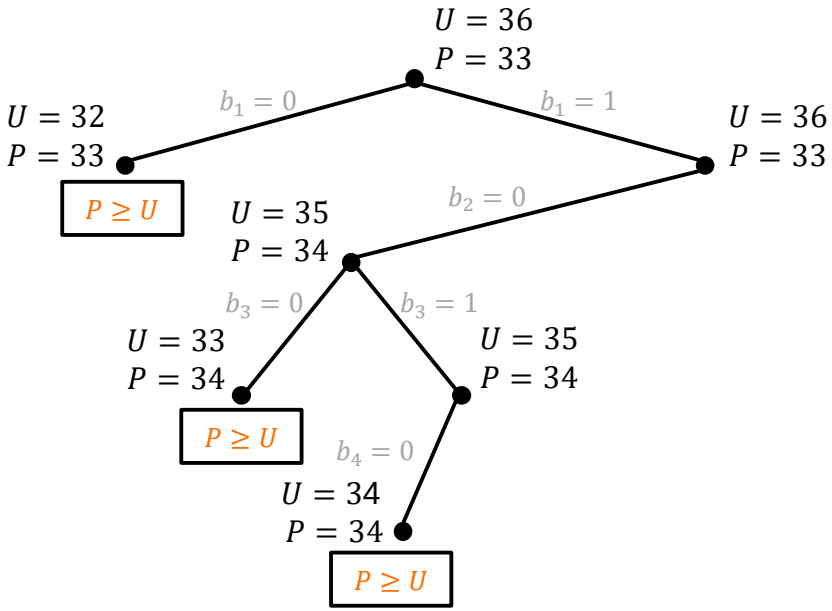


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

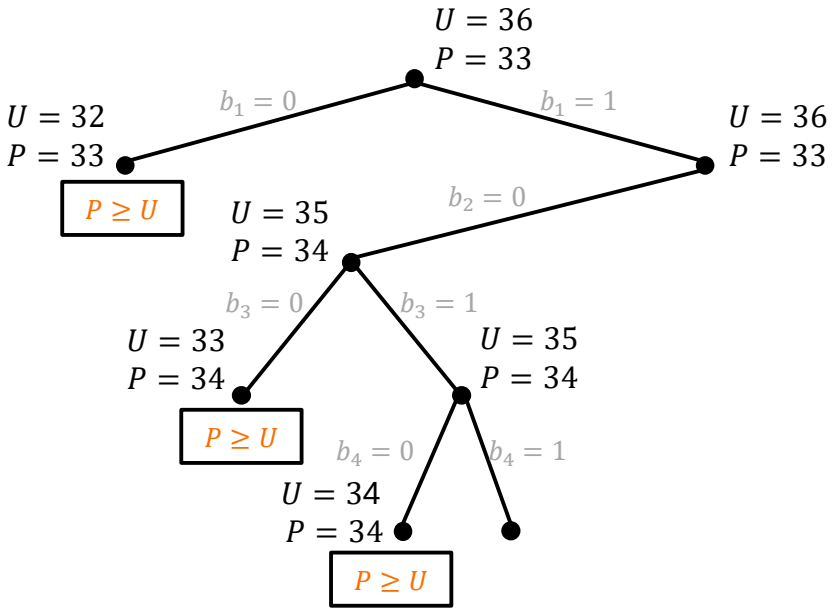


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

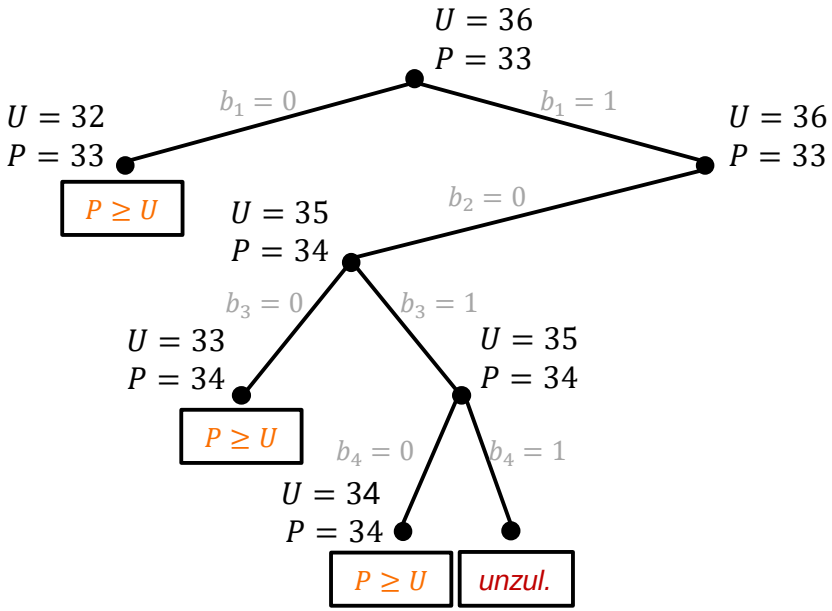


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
$\{1,2,3\}$	33
$\{1,3,5\}$	34

Branch-and-Bound – Beispiel

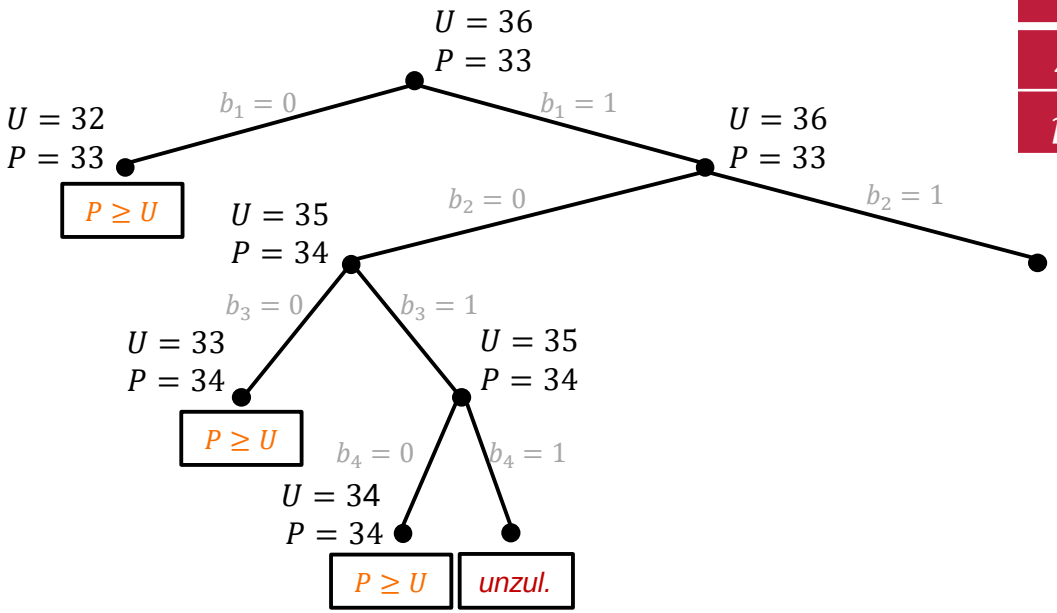


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
$\{1,2,3\}$	33
$\{1,3,5\}$	34

Branch-and-Bound – Beispiel

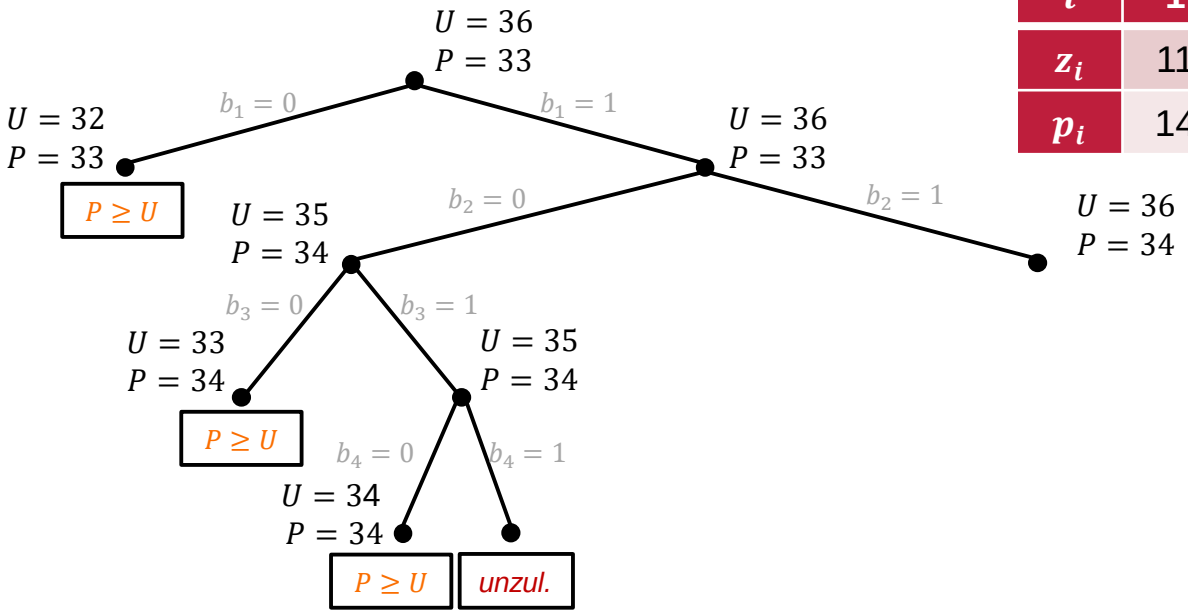


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

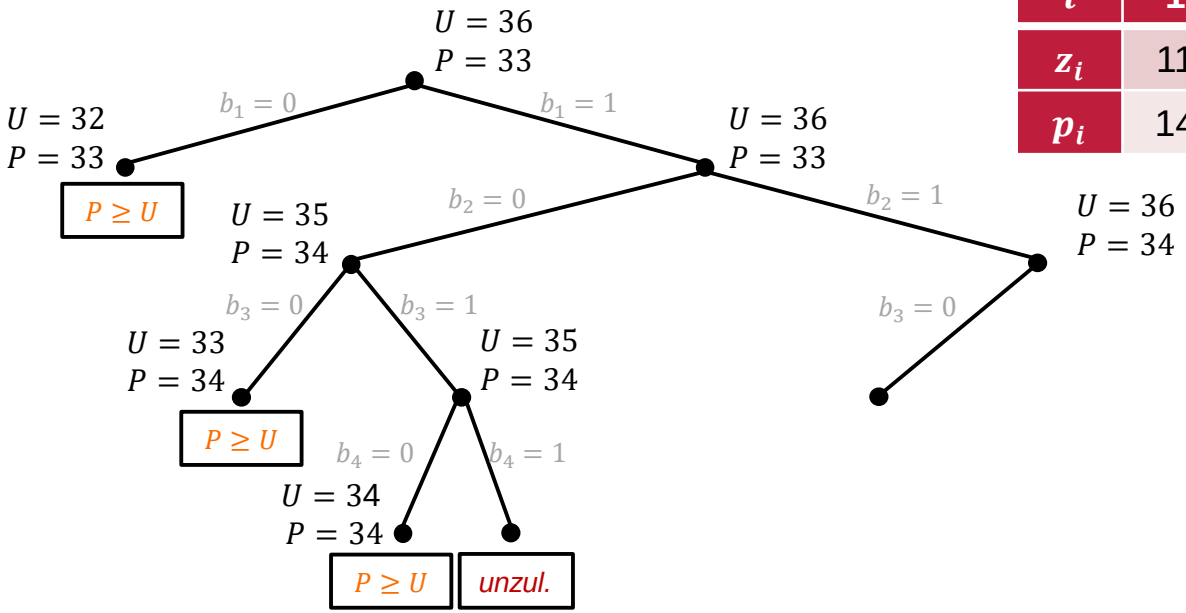


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

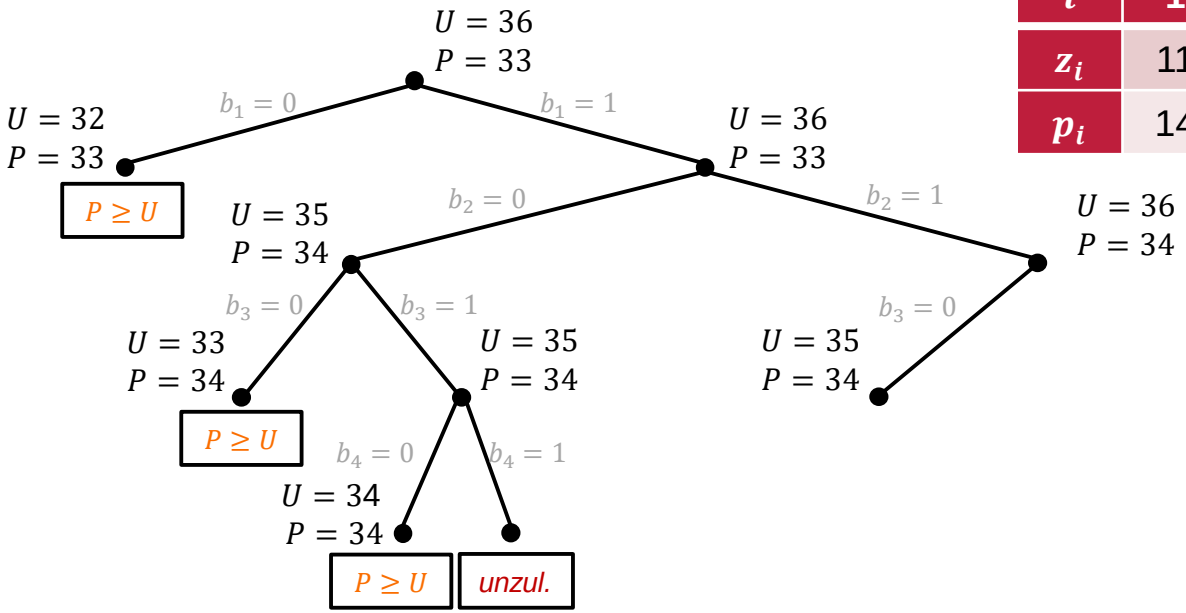


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

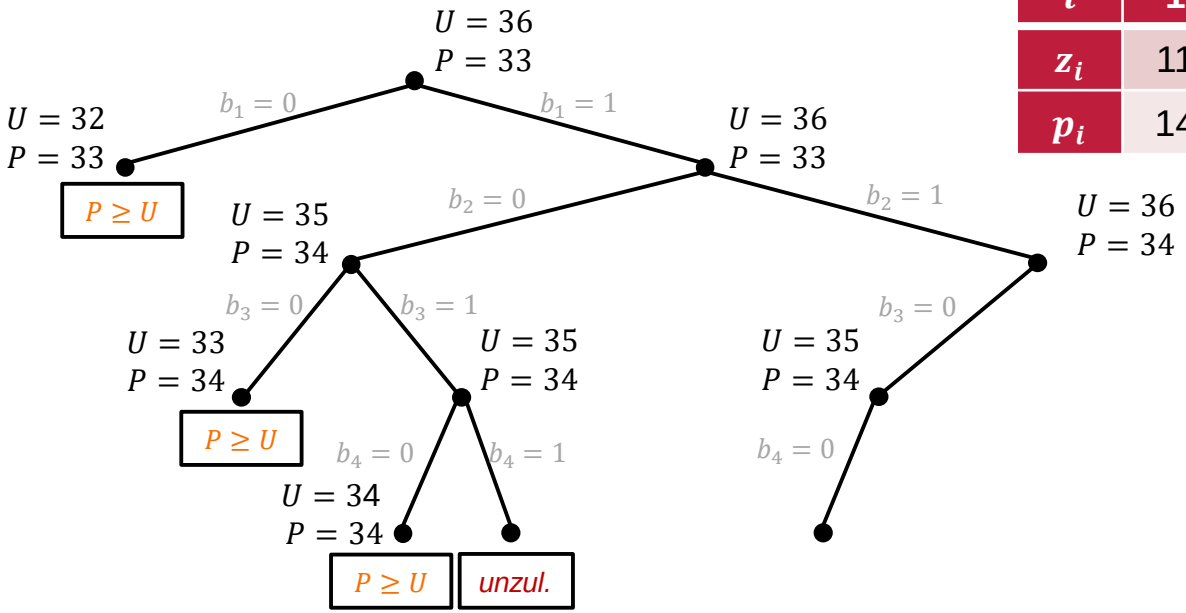


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

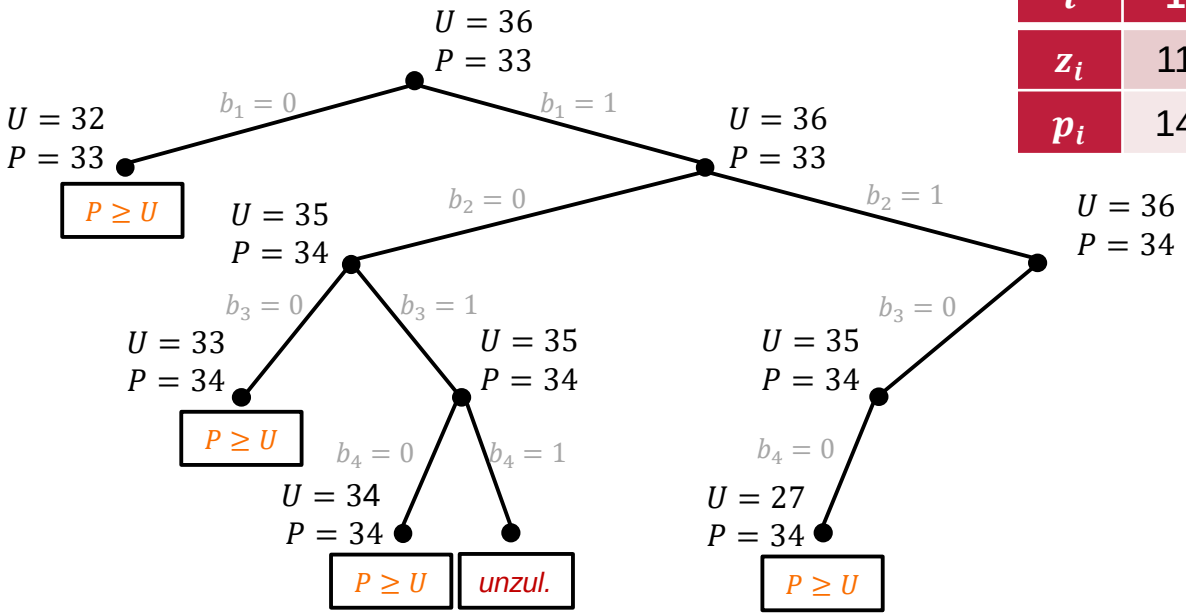


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

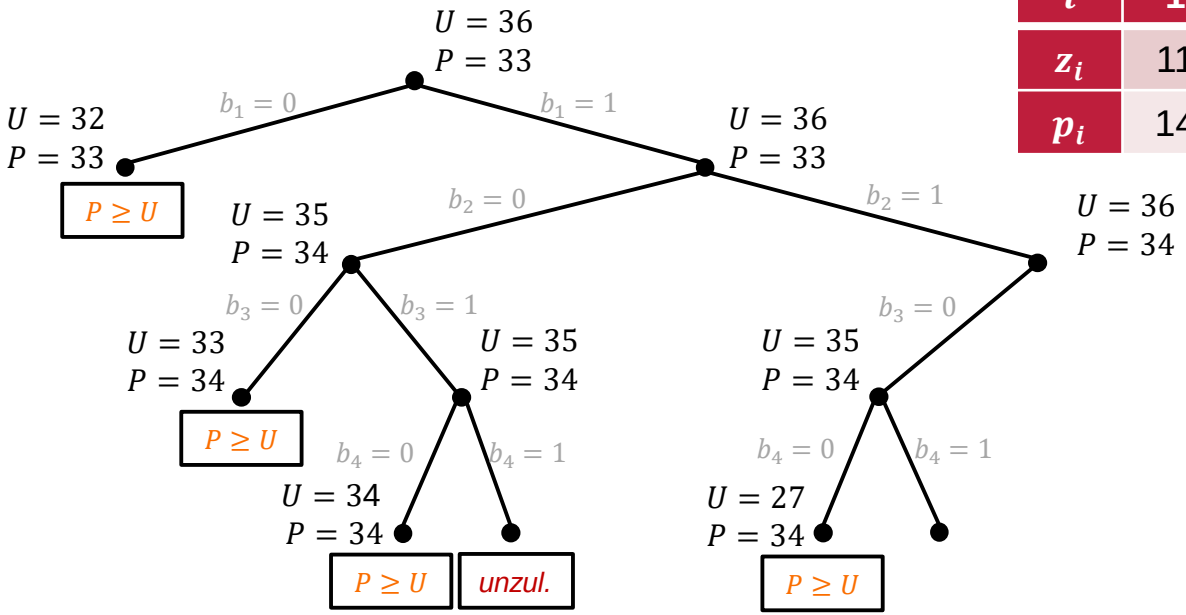


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

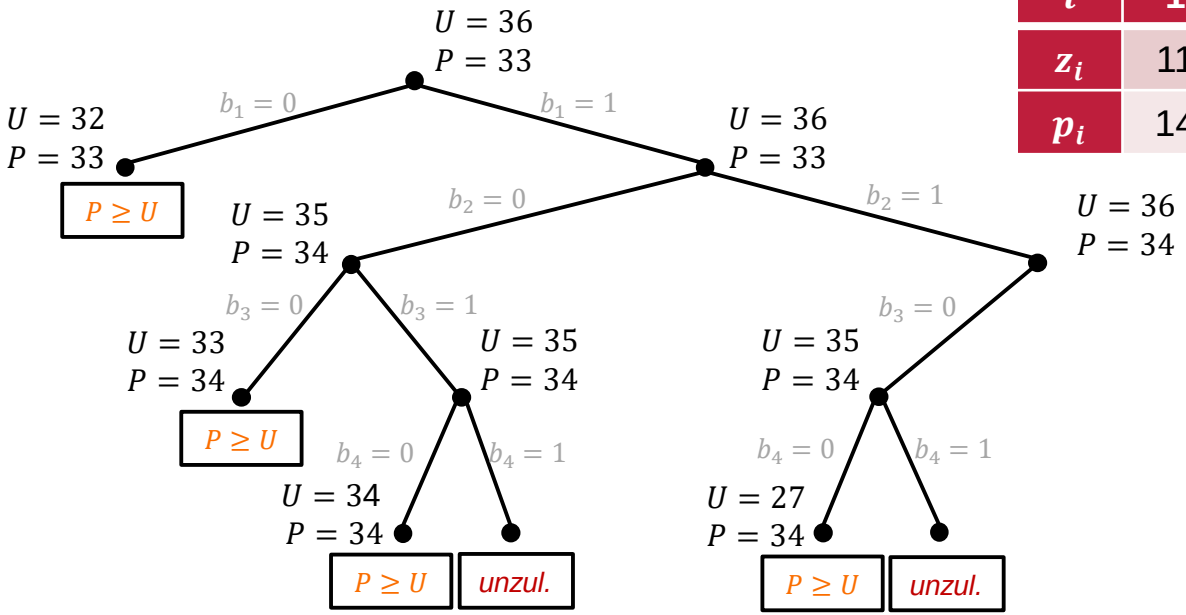


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

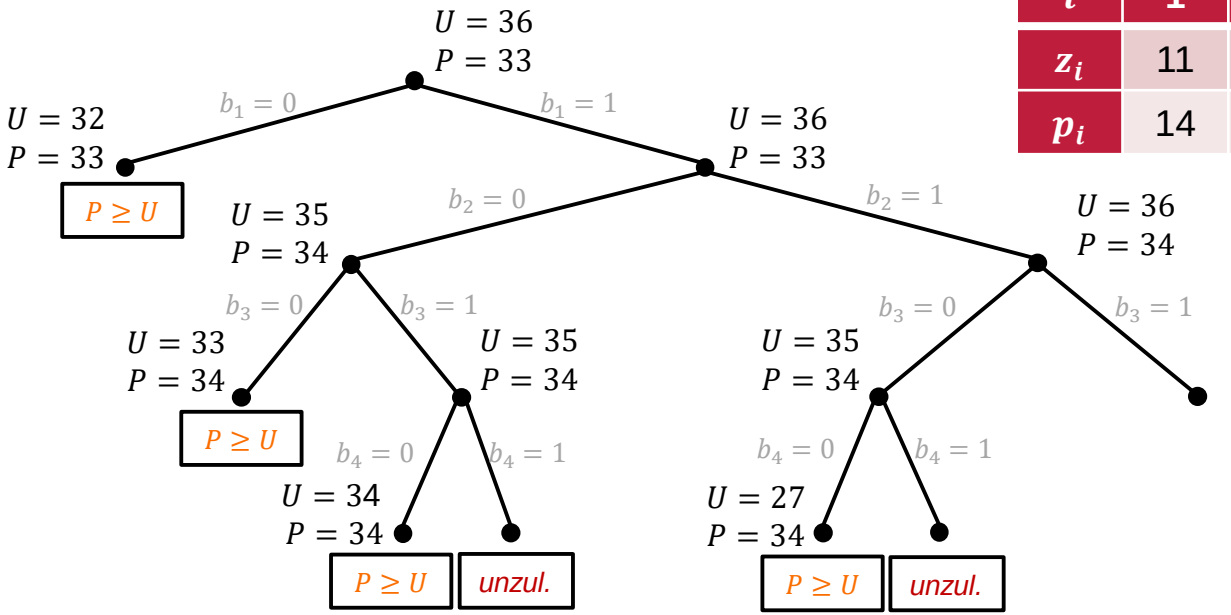


i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel



i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

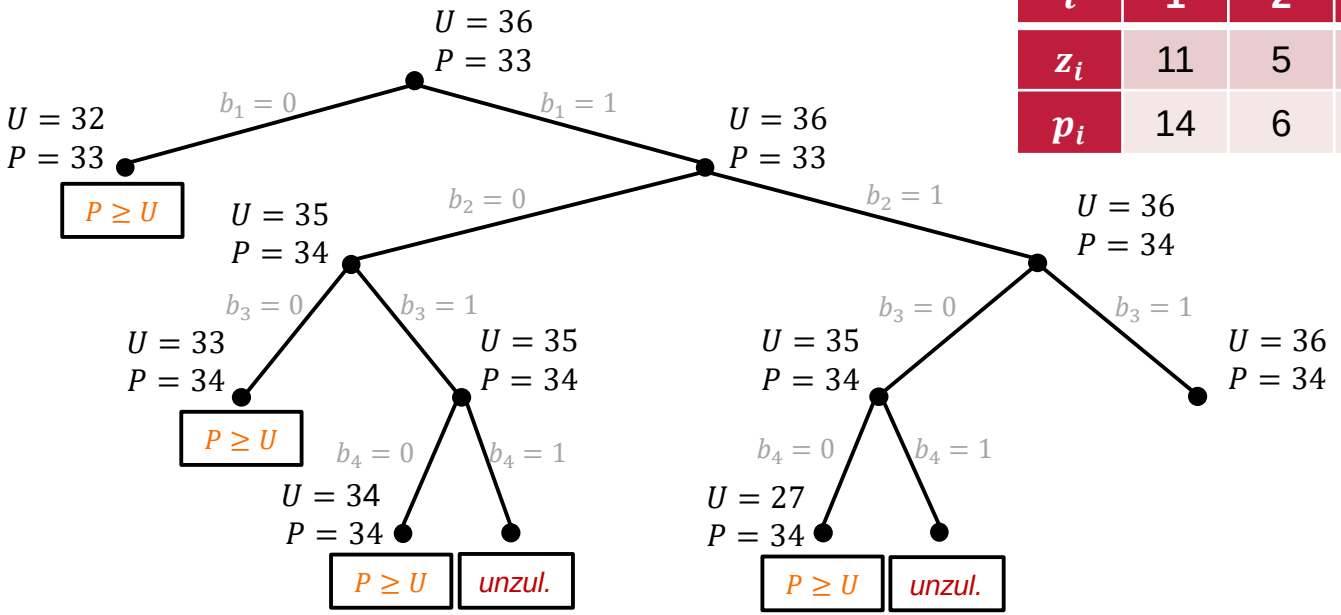
$Z = 33$

Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

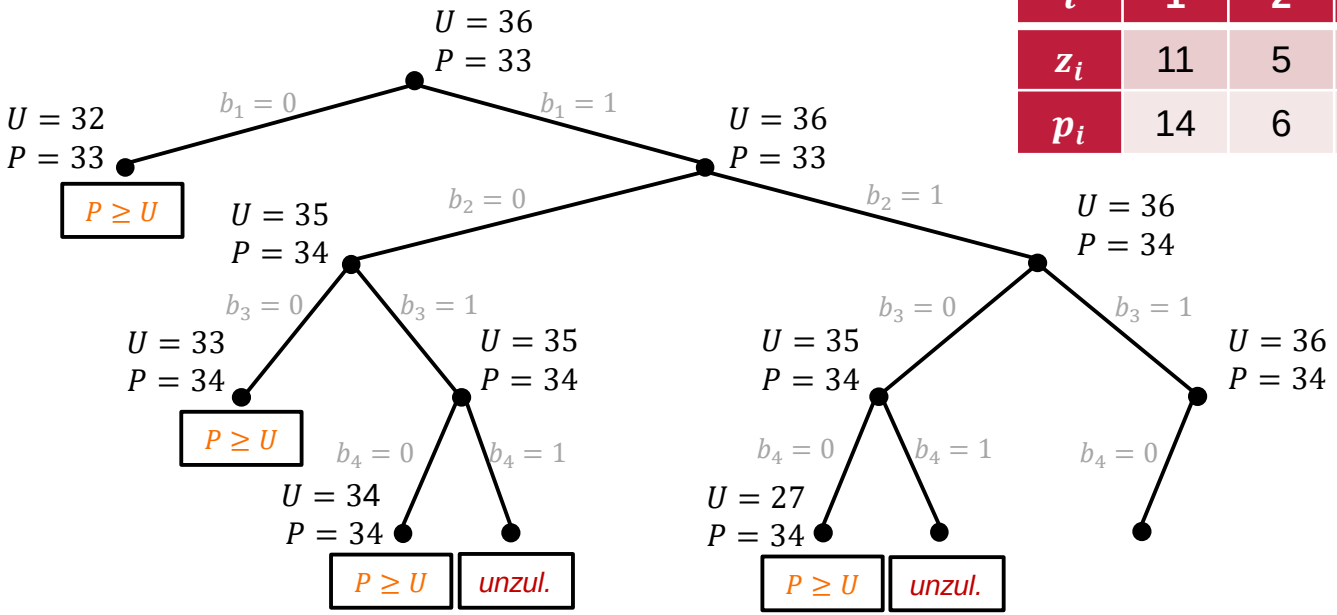


Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

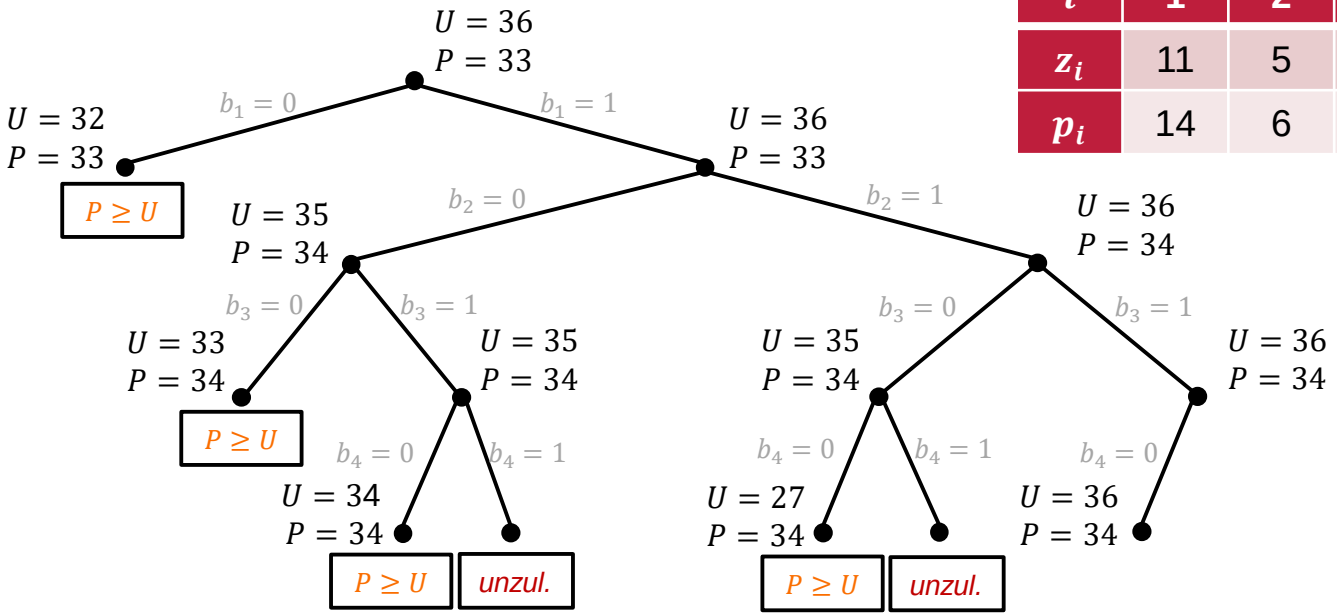


Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

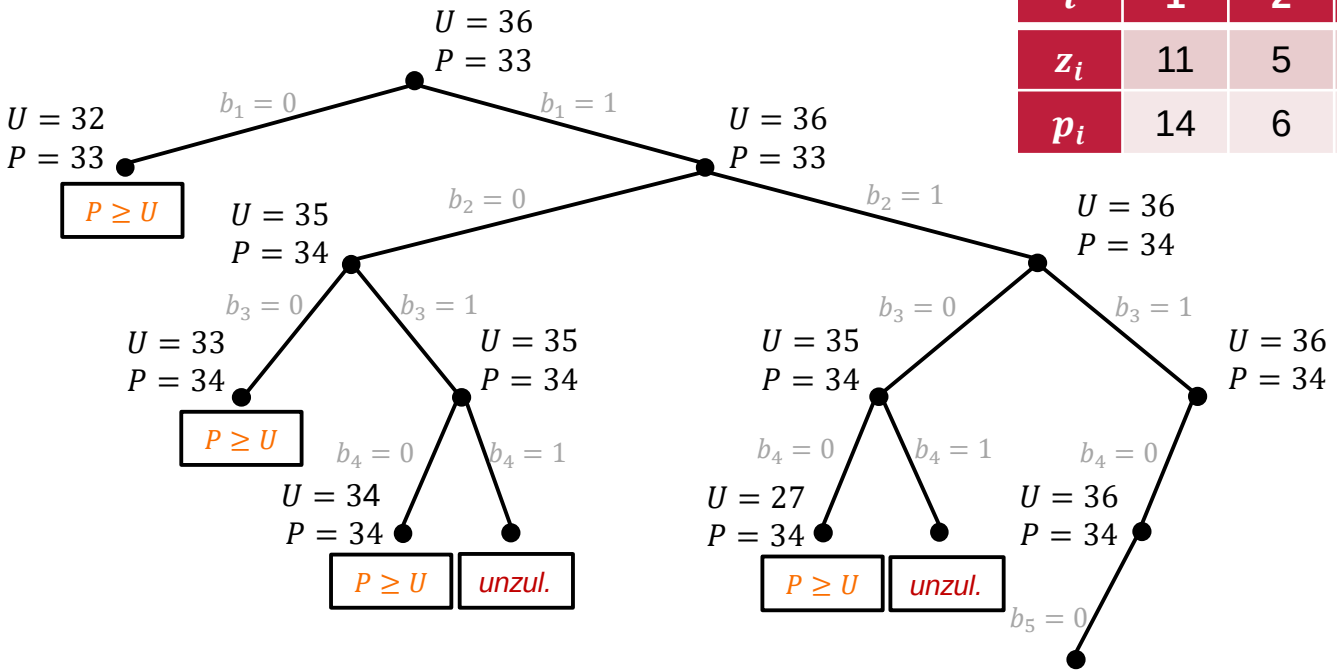


Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

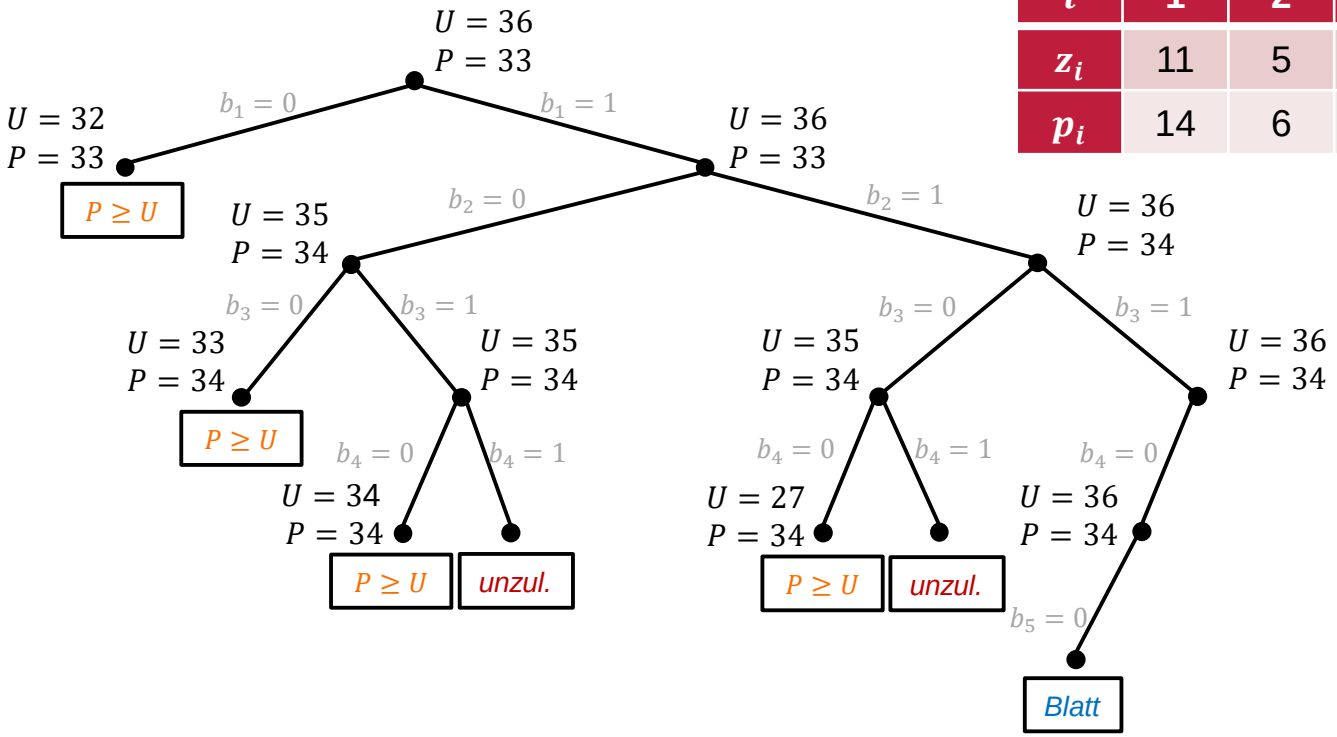


Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

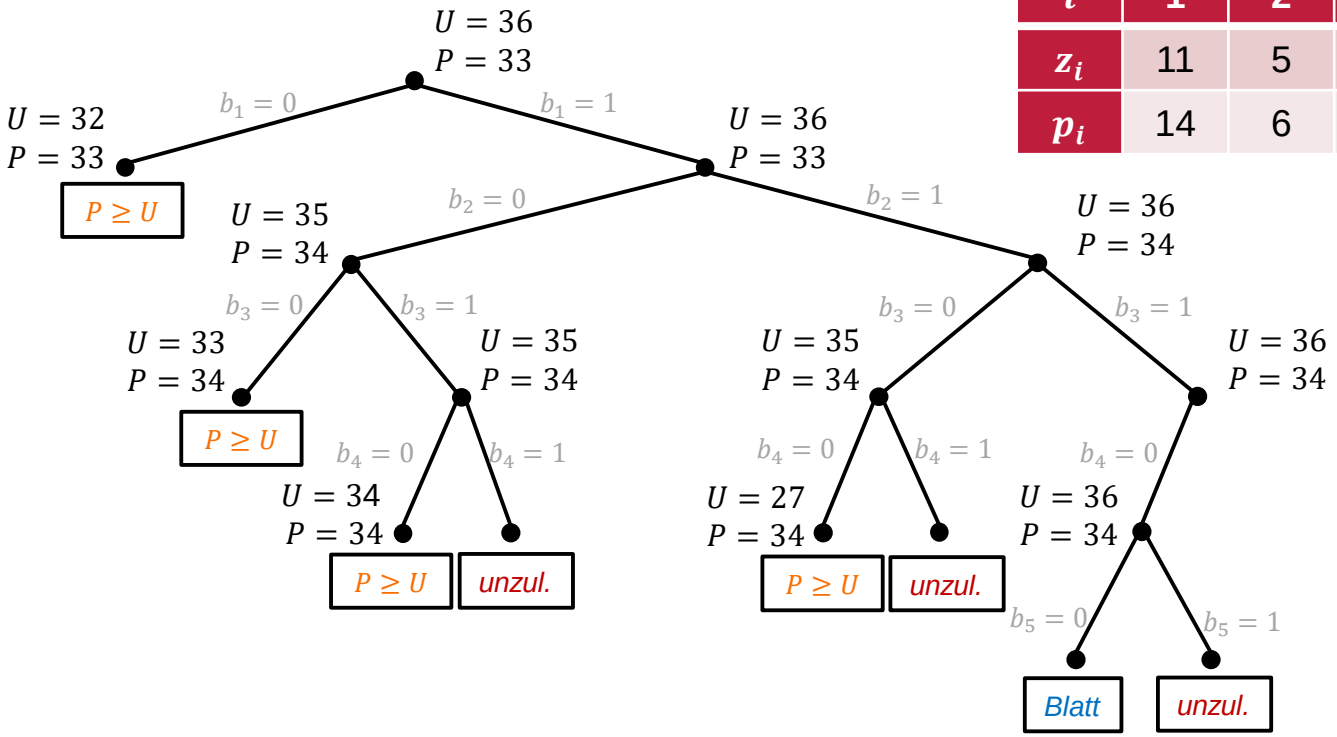


Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$

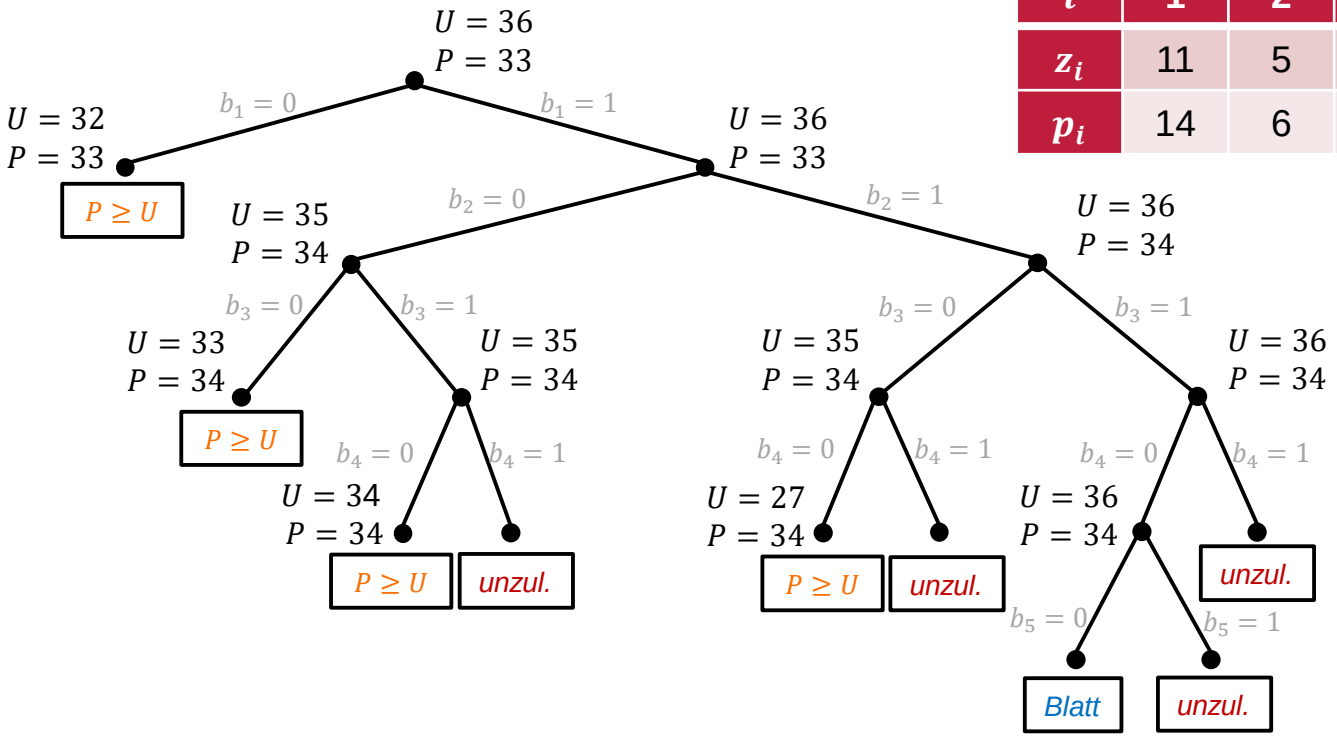


Menge	Wert
{1,2,3}	33
{1,3,5}	34

Branch-and-Bound – Beispiel

i	1	2	3	4	5
z_i	11	5	13	18	9
p_i	14	6	13	16	7

$Z = 33$



Menge	Wert
{1,2,3}	33
{1,3,5}	34

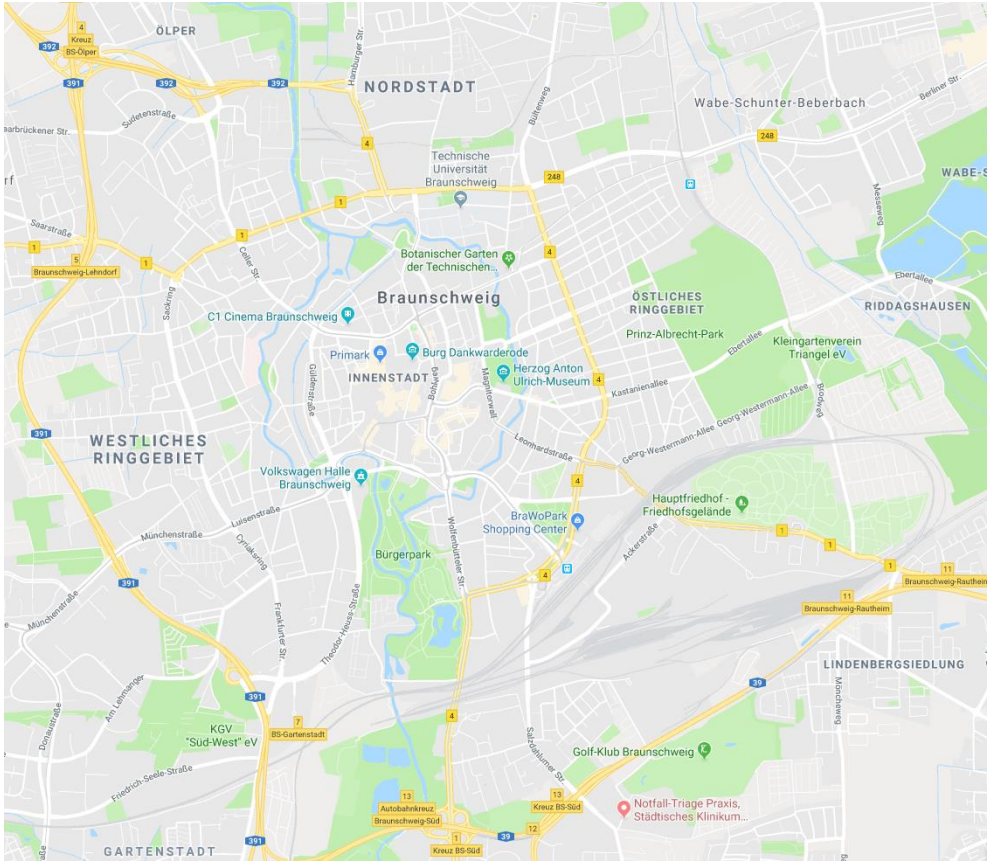
Euclidean Traveling Salesman Problem

Gegeben:
Punkte $p_1, \ldots, p_n \in \mathbb{R}^2$

Gesucht:
Permutation π mit

$$\sum_{i=1}^{n} \left\| p_{\pi(i)} - p_{\pi(i+1)} \right\|_2$$

minimal. Dabei ist $p_{\pi(n+1)} = p_{\pi(1)}$.
Finde also eine **kürzeste** Rundreise, die alle Punkte besucht.



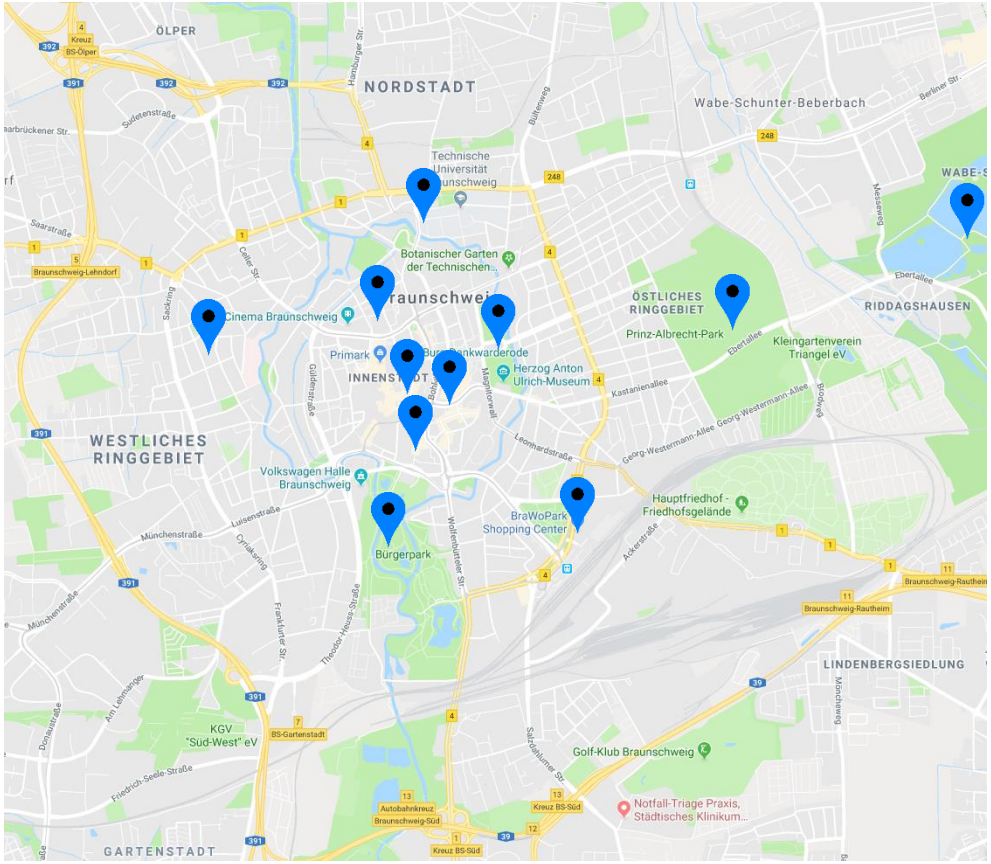
Euclidean Traveling Salesman Problem

Gegeben:
Punkte $p_1, \dots, p_n \in \mathbb{R}^2$

Gesucht:
Permutation π mit

$$\sum_{i=1}^{n} \left\| p_{\pi(i)} - p_{\pi(i+1)} \right\|_2$$

minimal. Dabei ist $p_{\pi(n+1)} = p_{\pi(1)}$.
Finde also eine **kürzeste** Rundreise, die alle Punkte besucht.



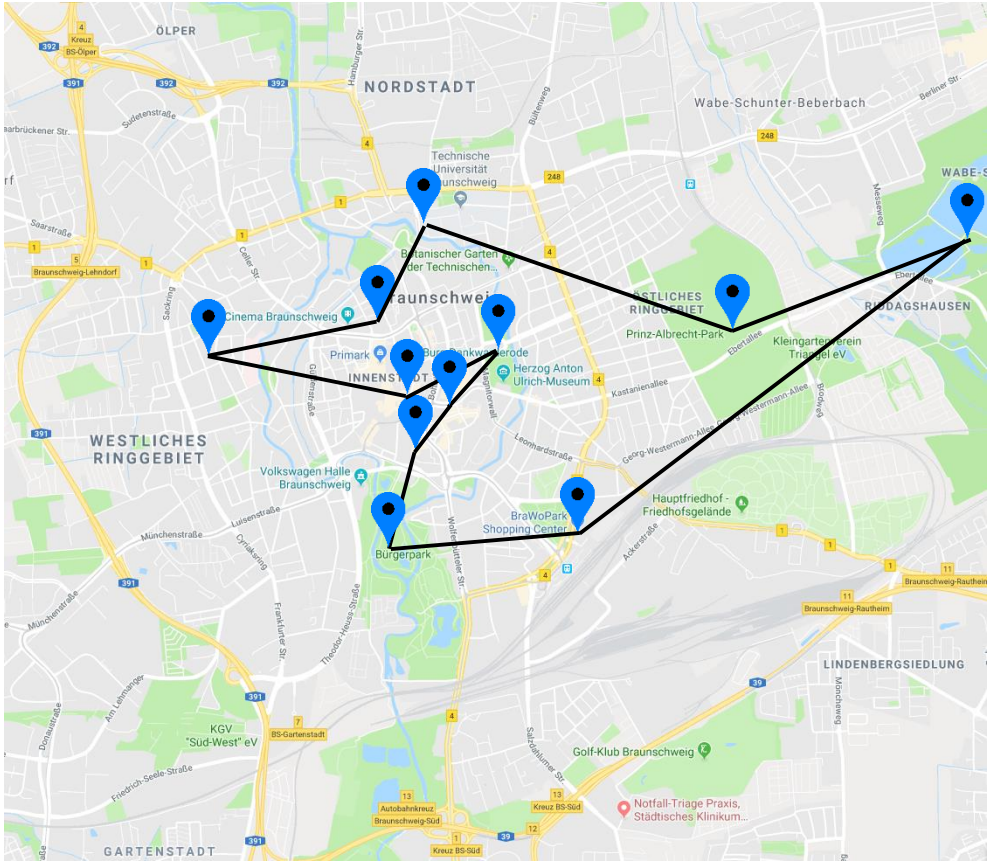
Euclidean Traveling Salesman Problem

Gegeben:
Punkte $p_1, \dots, p_n \in \mathbb{R}^2$

Gesucht:
Permutation π mit

$$\sum_{i=1}^{n} \left\| p_{\pi(i)} - p_{\pi(i+1)} \right\|_2$$

minimal. Dabei ist $p_{\pi(n+1)} = p_{\pi(1)}$.
Finde also eine **kürzeste** Rundreise, die alle Punkte besucht.



Euclidean Traveling Salesman Problem

Gegeben:
Punkte $p_1, \ldots, p_n \in \mathbb{R}^2$

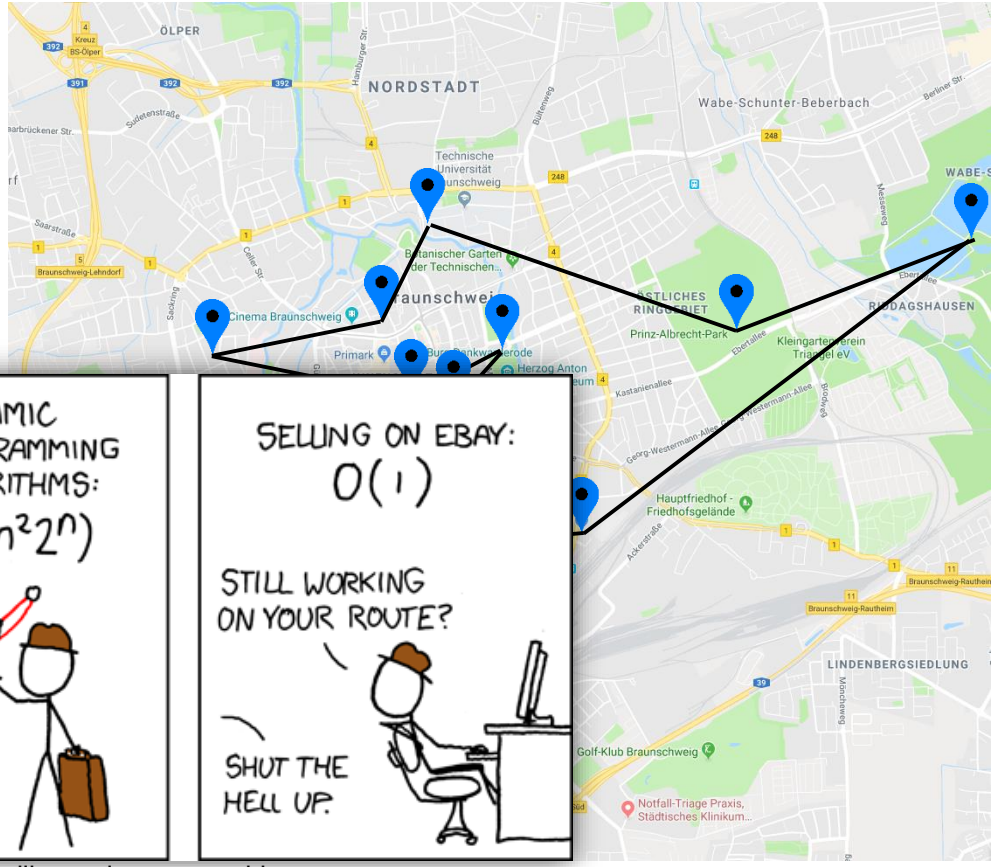
Gesucht:
Permutation π

$$\sum_{i=1}^{n} \|p_{\pi}$$

minimal. Dabe

Finde also eine

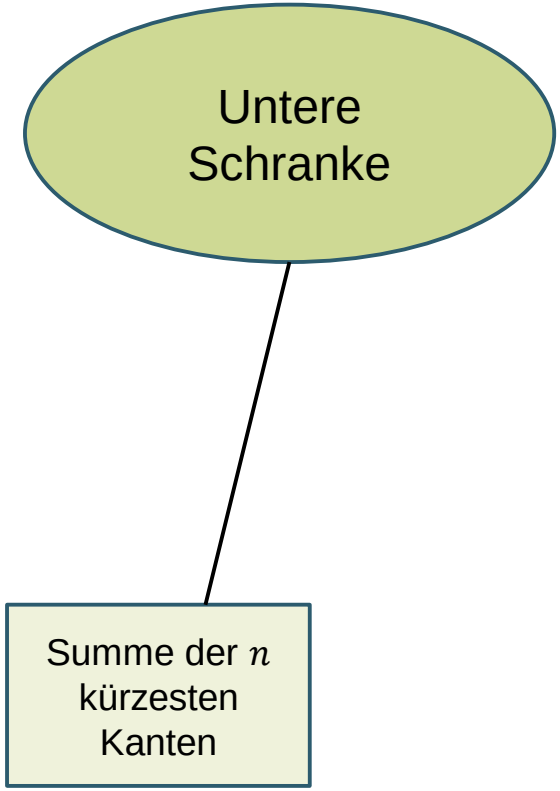
die alle Punkte



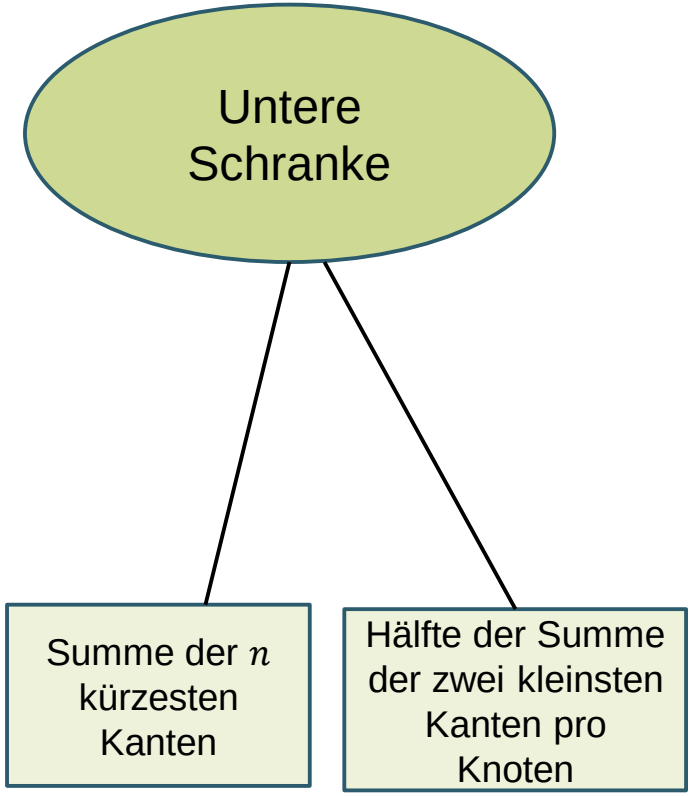
https://imgs.xkcd.com/comics/travelling_salesman_problem.png

ETSP – Schranken

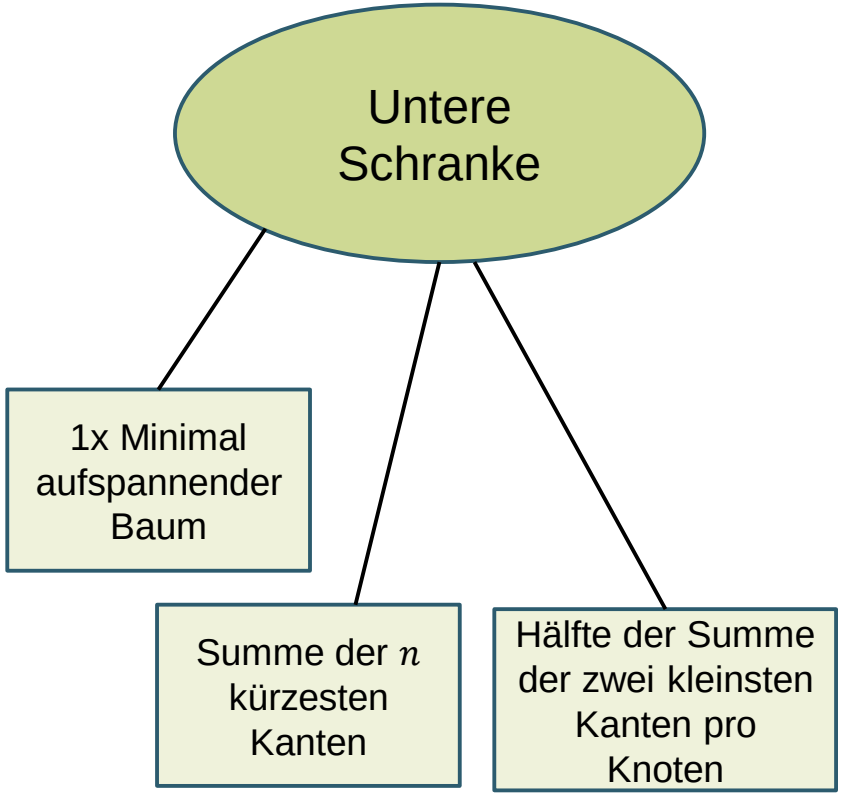
ETSP – Schranken



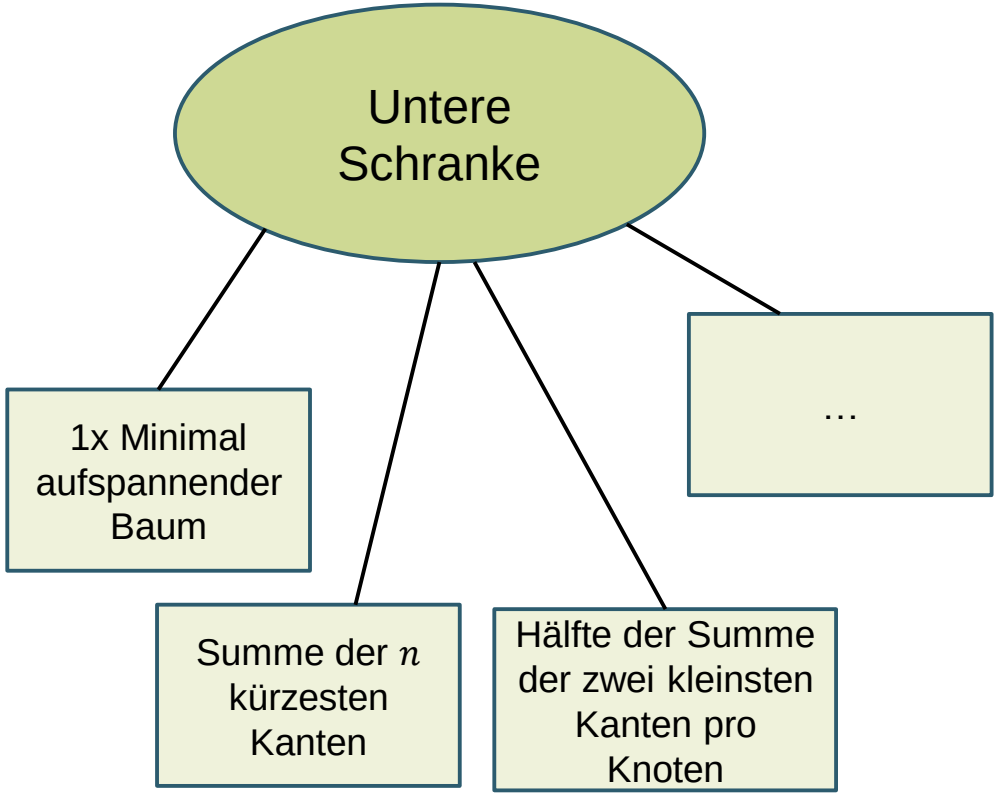
ETSP – Schranken



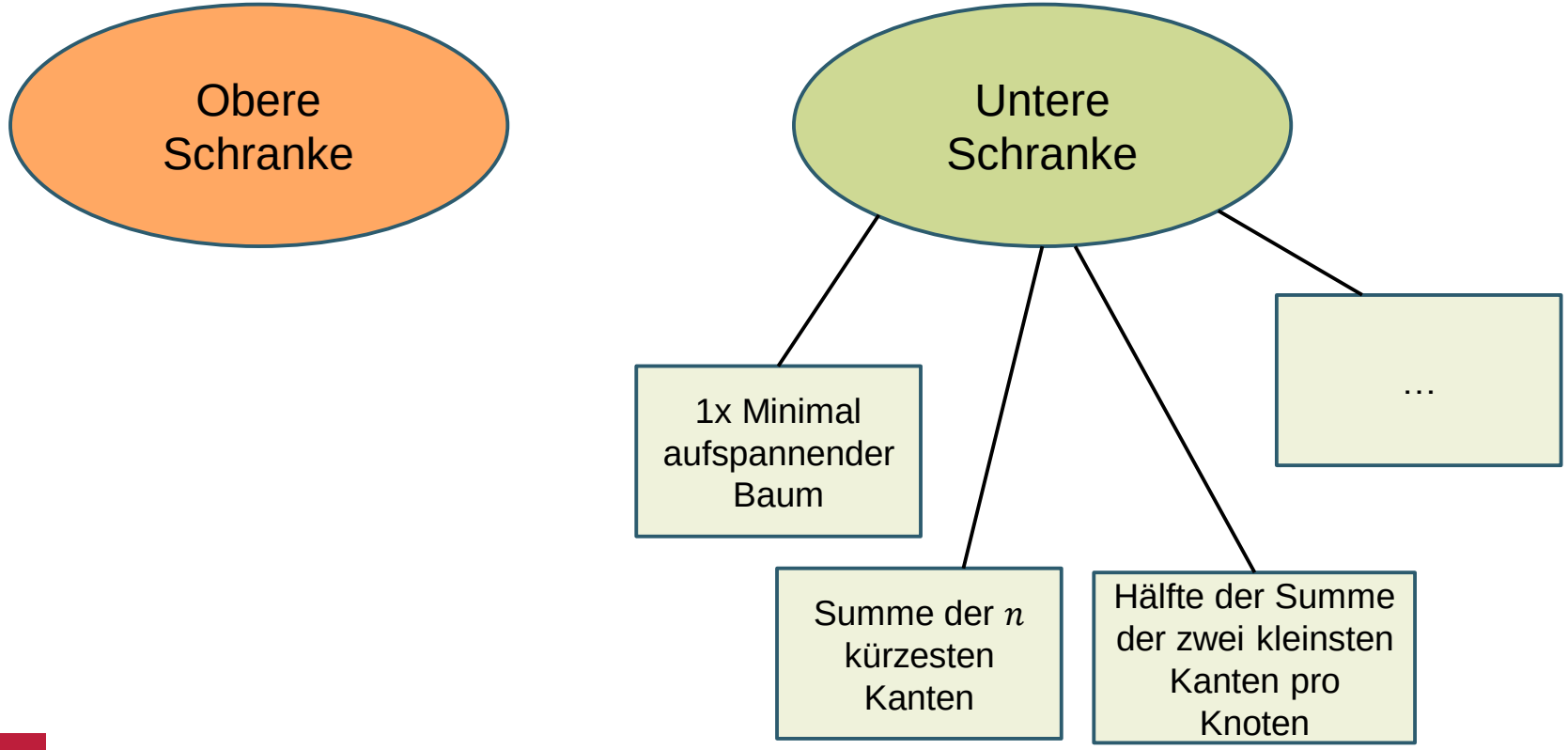
ETSP – Schranken



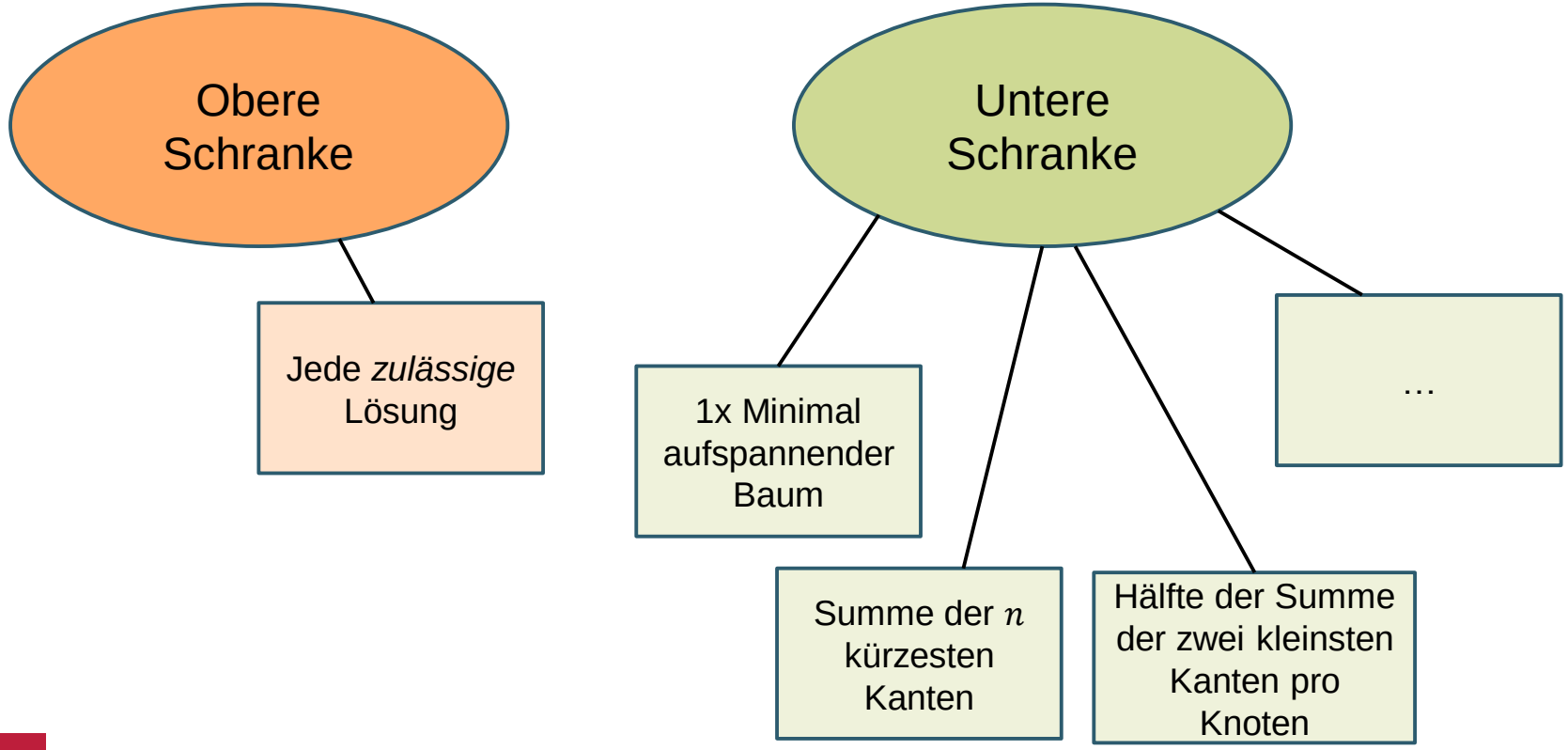
ETSP – Schranken



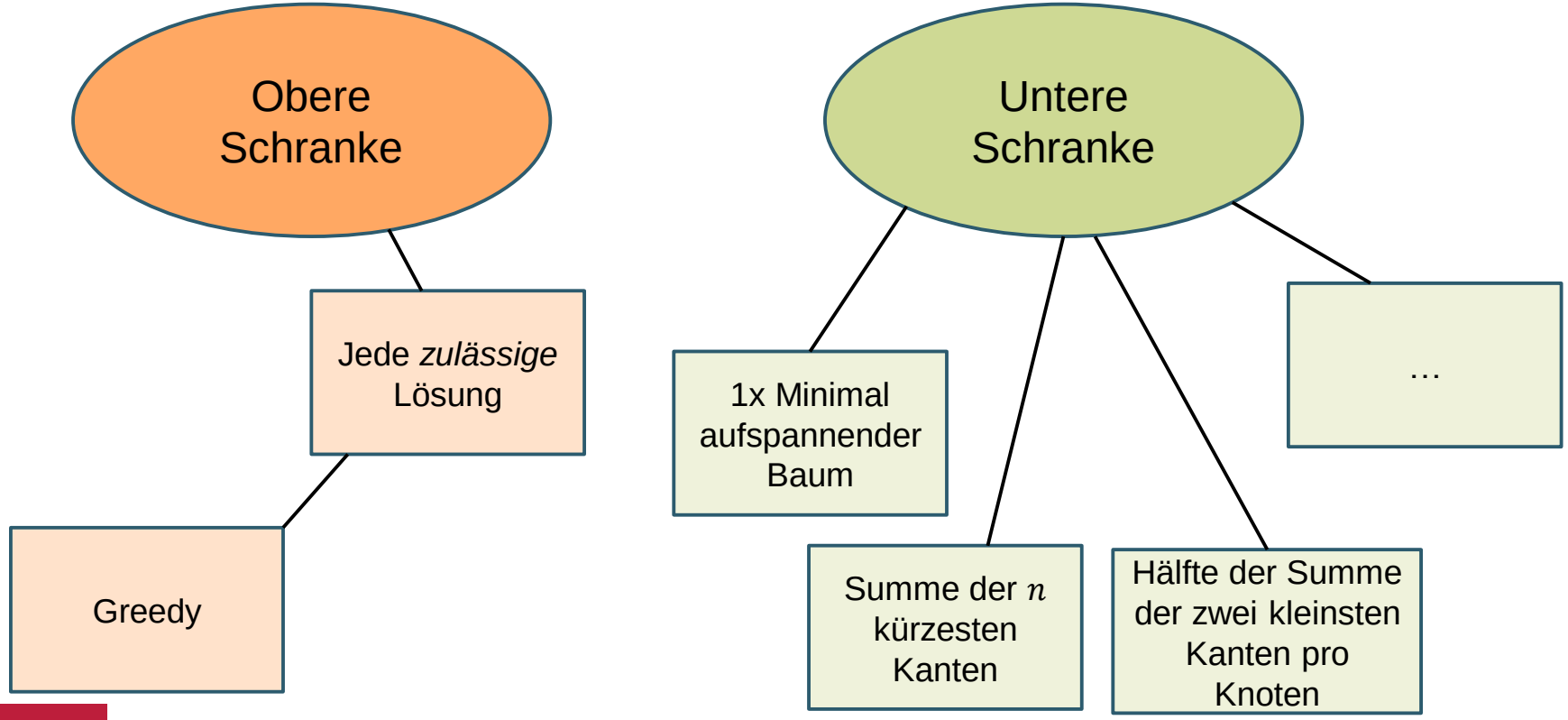
ETSP – Schranken



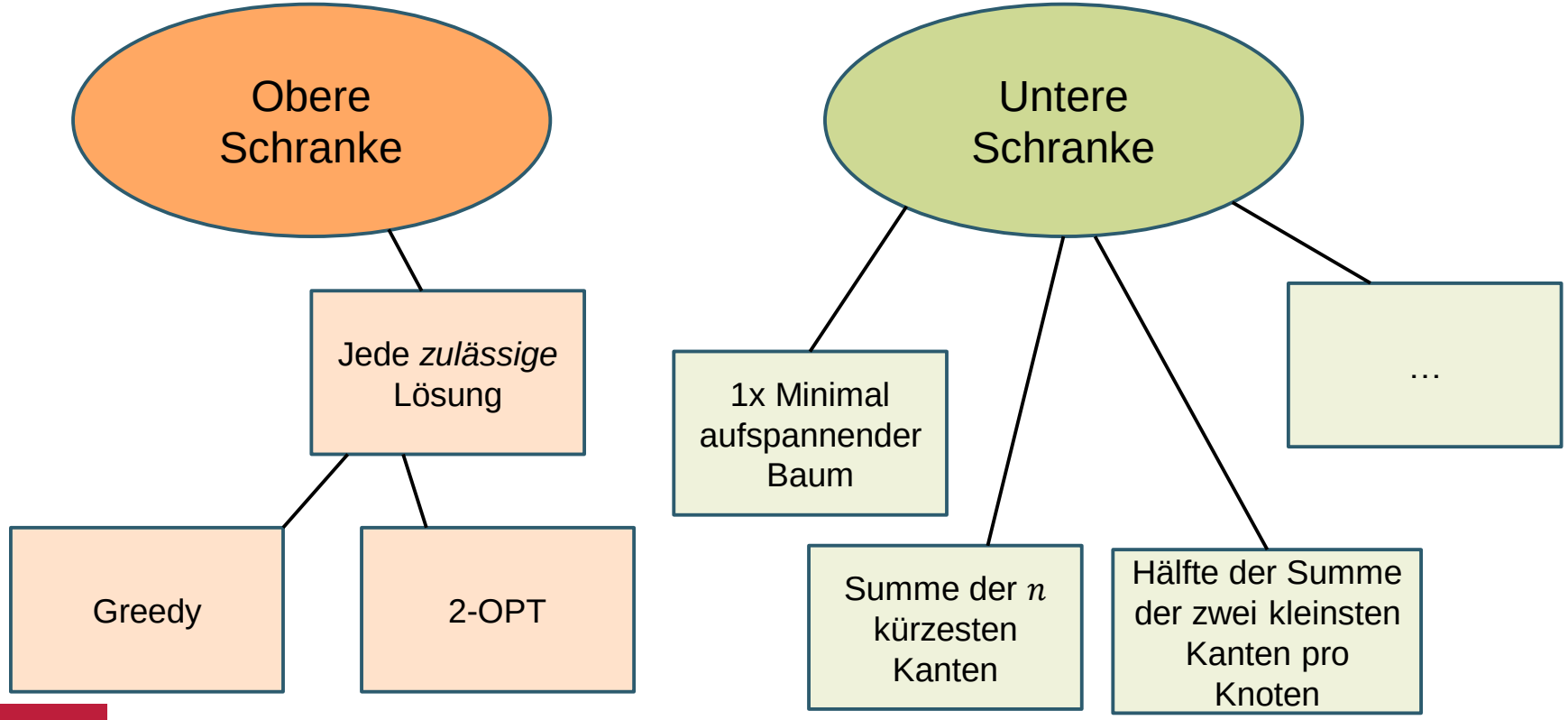
ETSP – Schranken



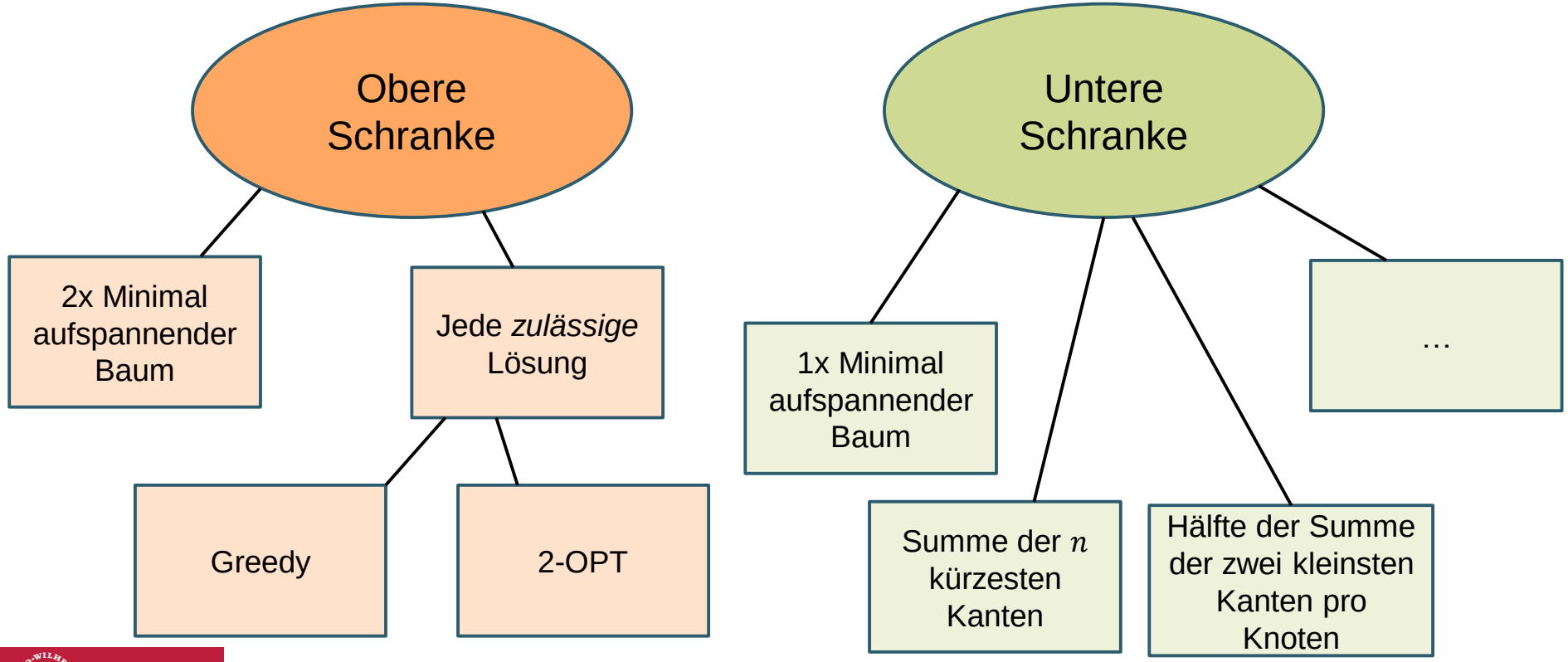
ETSP – Schranken



ETSP – Schranken

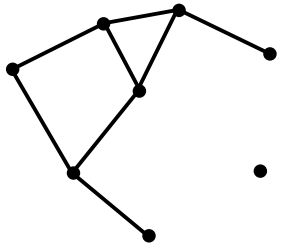


ETSP – Schranken

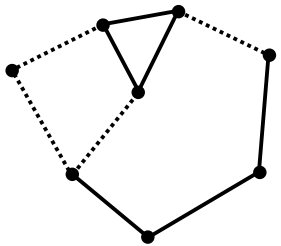


ETSP – Untere Schranken

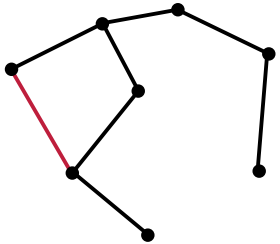
Summe der n kürzesten Kanten



Hälfte der Summe der zwei kleinsten Kanten pro Knoten



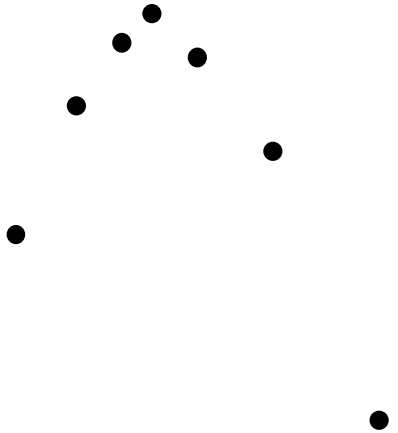
1x Minimal aufspannender Baum
(+n. kürzeste Kante)



ETSP – Obere Schranken

Greedy
Nearest-Neighbor

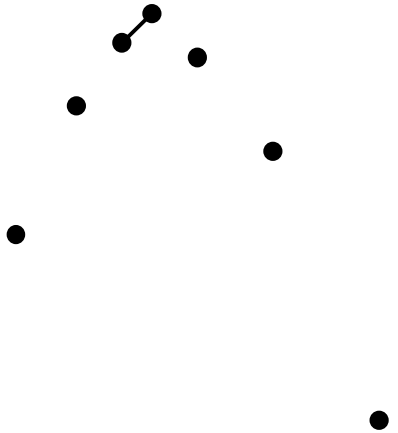
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.



ETSP – Obere Schranken

Greedy
Nearest-Neighbor

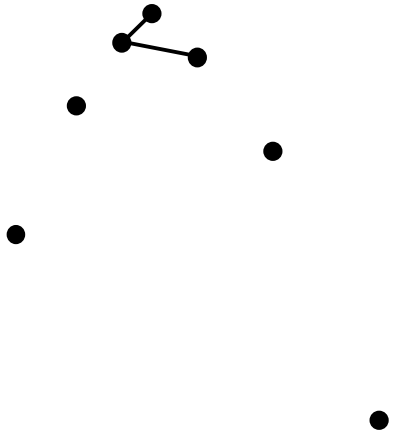
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.



ETSP – Obere Schranken

Greedy
Nearest-Neighbor

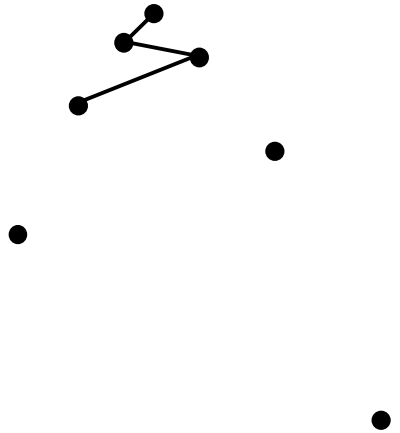
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.



ETSP – Obere Schranken

Greedy
Nearest-Neighbor

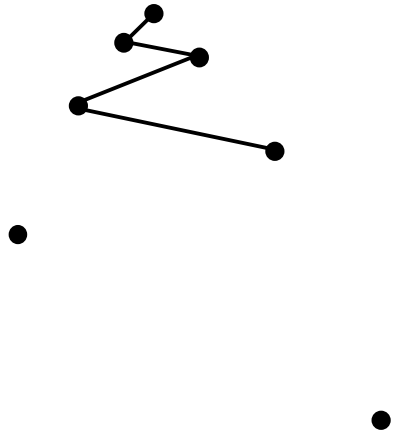
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.



ETSP – Obere Schranken

Greedy
Nearest-Neighbor

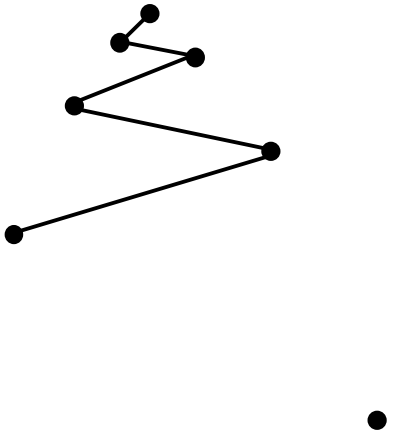
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.



ETSP – Obere Schranken

Greedy
Nearest-Neighbor

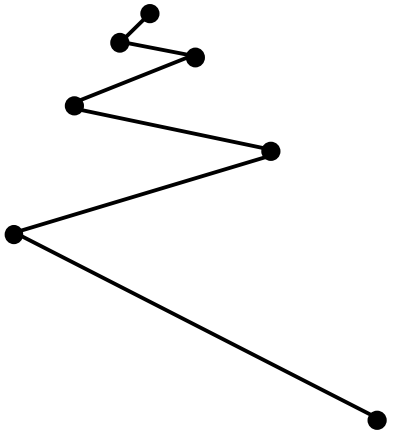
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.



ETSP – Obere Schranken

Greedy
Nearest-Neighbor

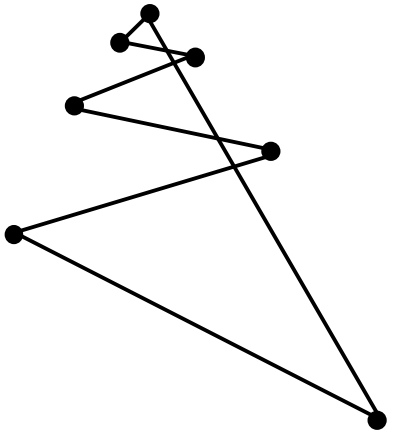
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.



ETSP – Obere Schranken

Greedy
Nearest-Neighbor

Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.



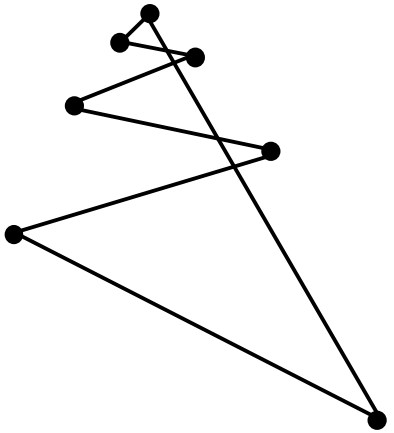
ETSP – Obere Schranken

Greedy
Nearest-Neighbor

Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.

2-OPT

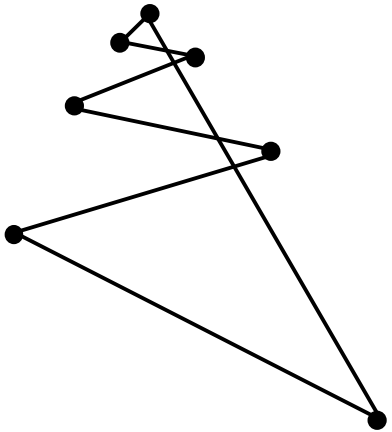
In einer beliebigen Tour, tausche zwei existierende Kanten mit zwei nicht-existierenden Kanten, um eine bessere Tour zu erhalten.



ETSP – Obere Schranken

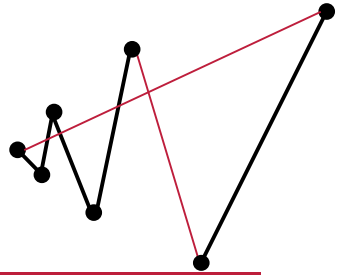
Greedy
Nearest-Neighbor

Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.



2-OPT

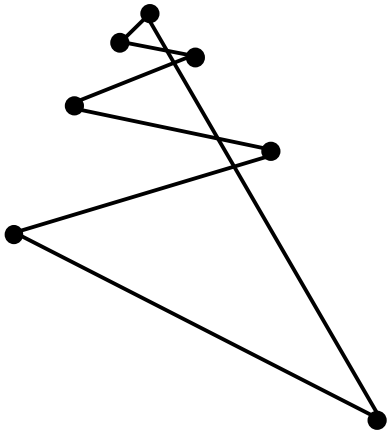
In einer beliebigen Tour, tausche zwei existierende Kanten mit zwei nicht-existierenden Kanten, um eine bessere Tour zu erhalten.



ETSP – Obere Schranken

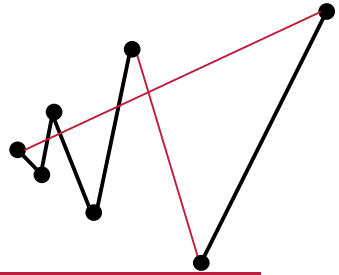
Greedy
Nearest-Neighbor

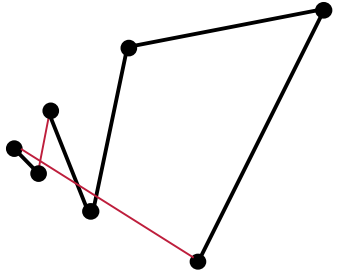
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.



2-OPT

In einer beliebigen Tour, tausche zwei existierende Kanten mit zwei nicht-existierenden Kanten, um eine bessere Tour zu erhalten.

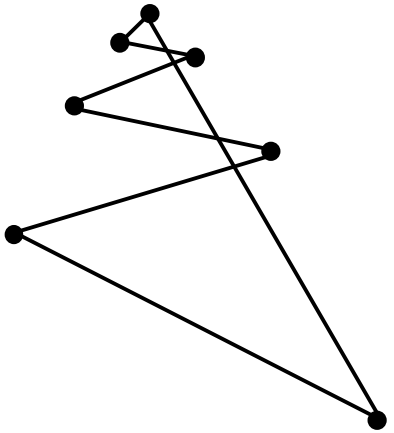




ETSP – Obere Schranken

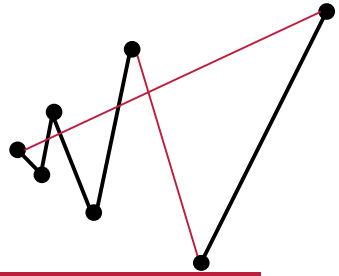
Greedy
Nearest-Neighbor

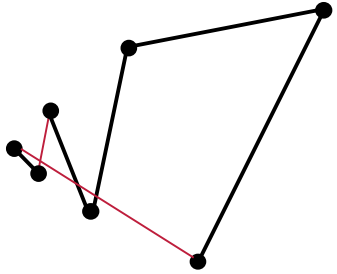
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.

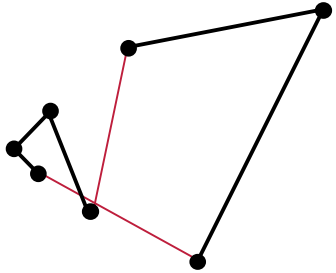


2-OPT

In einer beliebigen Tour, tausche zwei existierende Kanten mit zwei nicht-existierenden Kanten, um eine bessere Tour zu erhalten.



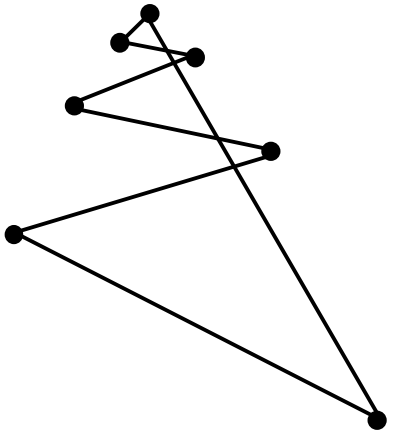




ETSP – Obere Schranken

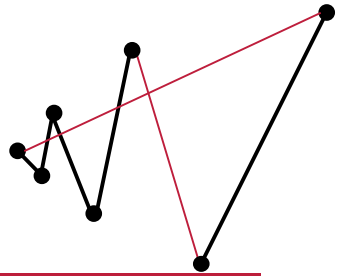
Greedy
Nearest-Neighbor

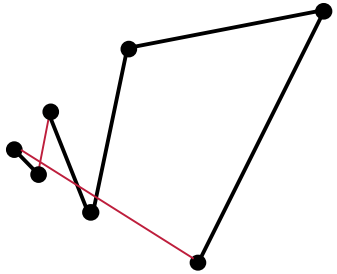
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.

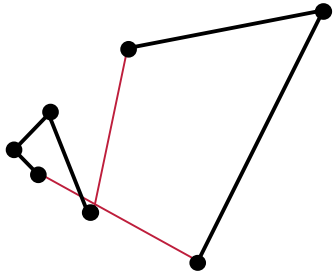


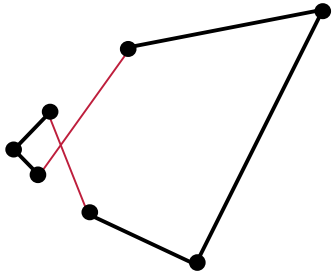
2-OPT

In einer beliebigen Tour, tausche zwei existierende Kanten mit zwei nicht-existierenden Kanten, um eine bessere Tour zu erhalten.





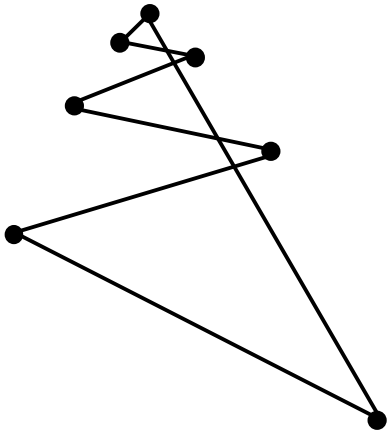




ETSP – Obere Schranken

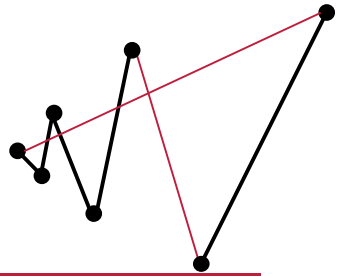
Greedy
Nearest-Neighbor

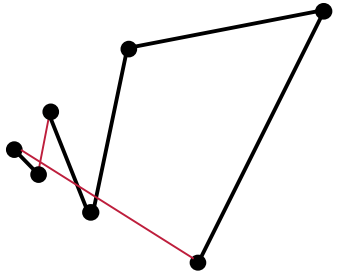
Starte bei p_1 und gehe iterativ zum nächstgelegenen Punkt.

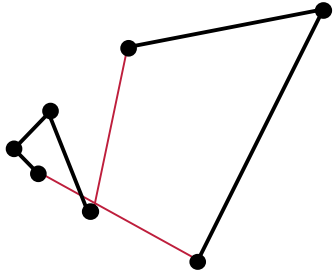


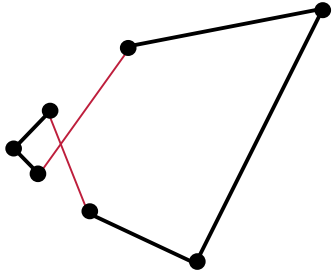
2-OPT

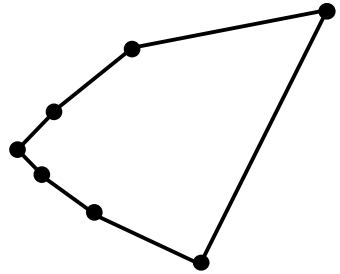
In einer beliebigen Tour, tausche zwei existierende Kanten mit zwei nicht-existierenden Kanten, um eine bessere Tour zu erhalten.











ETSP – Branch-and-Bound

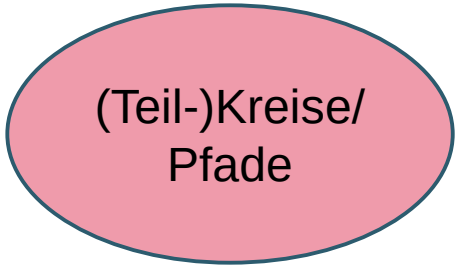
Zunächst: Worüber verzweigen? Wo können wir Entscheidungen treffen?

ETSP – Branch-and-Bound

Zunächst: Worüber verzweigen? Wo können wir Entscheidungen treffen?
Knoten? Kanten? Teilkreise?

ETSP – Branch-and-Bound

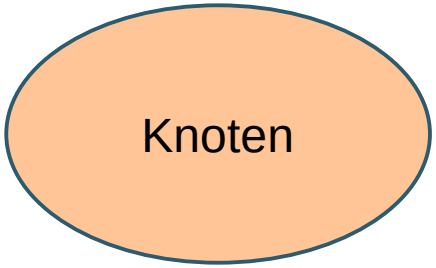
Zunächst: Worüber verzweigen? Wo können wir Entscheidungen treffen? Knoten? Kanten? Teilkreise?

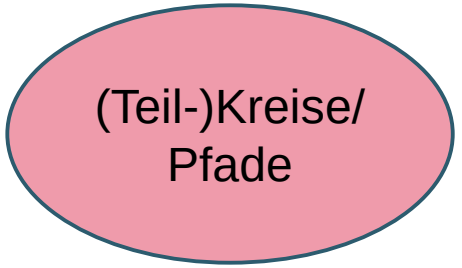


Also alle Permutationen testen?
Vorgänger von Knoten raten?
Wie kann ich darüber Schranken finden?

ETSP – Branch-and-Bound

Zunächst: Worüber verzweigen? Wo können wir Entscheidungen treffen?
Knoten? Kanten? Teilkreise?

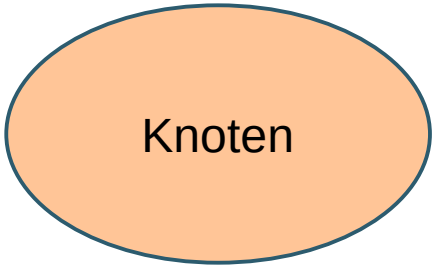




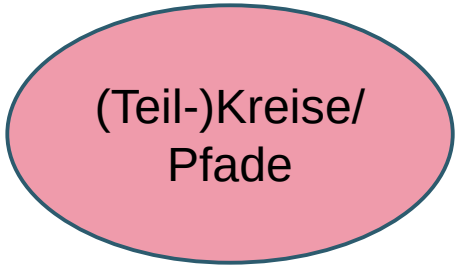
Also alle Permutationen testen?
Vorgänger von Knoten raten?
Wie kann ich darüber Schranken finden?

ETSP – Branch-and-Bound

Zunächst: Worüber verzweigen? Wo können wir Entscheidungen treffen?
Knoten? Kanten? Teilkreise?



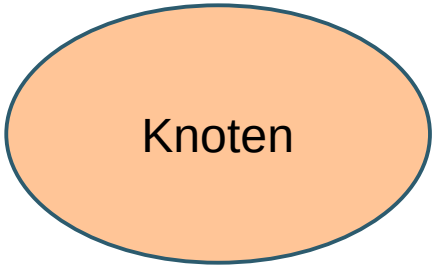
Entscheide für jeden Knoten, wann er besucht wird.
Verschiedene Knoten dürfen nicht gleichzeitig besucht werden.



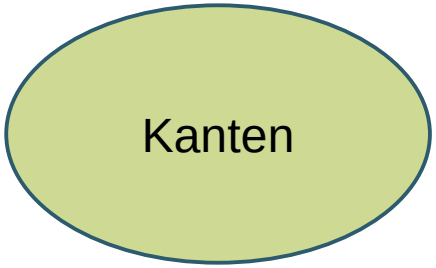
Also alle Permutationen testen?
Vorgänger von Knoten raten?
Wie kann ich darüber Schranken finden?

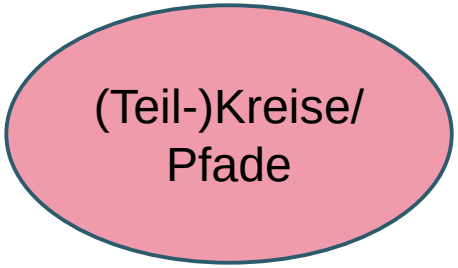
ETSP – Branch-and-Bound

Zunächst: Worüber verzweigen? Wo können wir Entscheidungen treffen?
Knoten? Kanten? Teilkreise?



Entscheide für jeden Knoten, wann er besucht wird.
Verschiedene Knoten dürfen nicht gleichzeitig besucht werden.

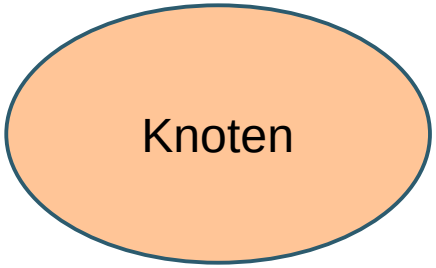




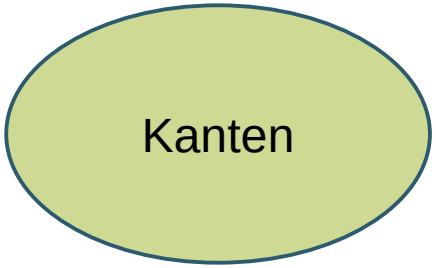
Also alle Permutationen testen?
Vorgänger von Knoten raten?
Wie kann ich darüber Schranken finden?

ETSP – Branch-and-Bound

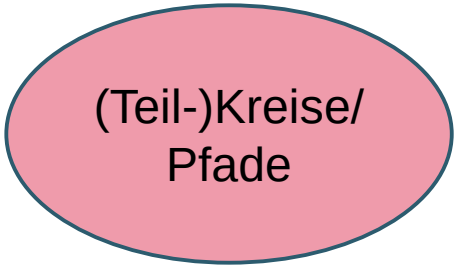
Zunächst: Worüber verzweigen? Wo können wir Entscheidungen treffen?
Knoten? Kanten? Teilkreise?



Entscheide für jeden Knoten, wann er besucht wird.
Verschiedene Knoten dürfen nicht gleichzeitig besucht werden.



Kante ist entweder da, oder nicht.
Entscheidungen sind unabhängig.
Einfach Schranken zu finden.



Also alle Permutationen testen?
Vorgänger von Knoten raten?
Wie kann ich darüber Schranken finden?

ETSP – zulässige Teillösungen

Betrachte eine beliebige Reihenfolge der Kanten. Sei $x_i \in \{0,1\}$ die Entscheidungsvariable für die i-te Kante.
Wann ist eine Auswahl von Kanten nicht zulässig?

ETSP – zulässige Teillösungen

Betrachte eine beliebige Reihenfolge der Kanten. Sei $x_i \in \{0,1\}$ die Entscheidungsvariable für die i-te Kante.
Wann ist eine Auswahl von Kanten nicht zulässig?

1. Maximal n Kanten ausgewählt, also
   $\sum_i x_i \leq n$

ETSP – zulässige Teillösungen

Betrachte eine beliebige Reihenfolge der Kanten. Sei $x_i \in \{0,1\}$ die Entscheidungsvariable für die i-te Kante.
Wann ist eine Auswahl von Kanten nicht zulässig?

1. Maximal n Kanten ausgewählt, also
   $\sum_i x_i \leq n$
2. An jedem Punkt v dürfen nur zwei Kanten liegen, also
   $\sum_{i \in \delta(v)} x_i \leq 2$

ETSP – zulässige Teillösungen

Betrachte eine beliebige Reihenfolge der Kanten. Sei $x_i \in \{0,1\}$ die Entscheidungsvariable für die i-te Kante.
Wann ist eine Auswahl von Kanten nicht zulässig?

1. Maximal n Kanten ausgewählt, also
   $\sum_i x_i \leq n$
2. An jedem Punkt v dürfen nur zwei Kanten liegen, also
   $\sum_{i \in \delta(v)} x_i \leq 2$
3. Falls $< n$ Kanten ausgewählt sind, darf kein Kreis existieren. (Erkennbar mit BFS/DFS aus AuD)

ETSP – zulässige Teillösungen

Betrachte eine beliebige Reihenfolge der Kanten. Sei $x_i \in \{0,1\}$ die Entscheidungsvariable für die i-te Kante.
Wann ist eine Auswahl von Kanten nicht zulässig?

1. Maximal n Kanten ausgewählt, also
   $\sum_i x_i \leq n$
2. An jedem Punkt v dürfen nur zwei Kanten liegen, also
   $\sum_{i \in \delta(v)} x_i \leq 2$
3. Falls $< n$ Kanten ausgewählt sind, darf kein Kreis existieren. (Erkennbar mit BFS/DFS aus AuD)
4. Falls $= n$ Kanten ausgewählt sind, muss exakt ein Kreis existieren. (Wieder mit BFS/DFS)

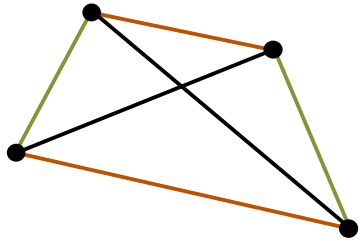
ETSP – zulässige Teillösungen

Betrachte eine beliebige Reihenfolge der Kanten. Sei $x_i \in \{0,1\}$ die Entscheidungsvariable für die i-te Kante.
Wann ist eine Auswahl von Kanten nicht zulässig?

1. Maximal n Kanten ausgewählt, also
   $\sum_i x_i \leq n$
2. An jedem Punkt v dürfen nur zwei Kanten liegen, also
   $\sum_{i \in \delta(v)} x_i \leq 2$
3. Falls $< n$ Kanten ausgewählt sind, darf kein Kreis existieren. (Erkennbar mit BFS/DFS aus AuD)
4. Falls $= n$ Kanten ausgewählt sind, muss exakt ein Kreis existieren. (Wieder mit BFS/DFS)
5. Es dürfen sich keine zwei Kanten schneiden.

ETSP – zulässige Teillösungen

Betrachte eine beliebige Reihenfolge der Kanten. Sei $x_i \in \{0,1\}$ die Entscheidungsvariable für die i-te Kante.
Wann ist eine Auswahl von Kanten nicht zulässig?

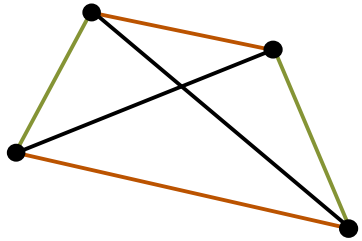


1. Maximal n Kanten ausgewählt, also
   $\sum_i x_i \leq n$
2. An jedem Punkt v dürfen nur zwei Kanten liegen, also
   $\sum_{i \in \delta(v)} x_i \leq 2$
3. Falls $< n$ Kanten ausgewählt sind, darf kein Kreis existieren. (Erkennbar mit BFS/DFS aus AuD)
4. Falls $= n$ Kanten ausgewählt sind, muss exakt ein Kreis existieren. (Wieder mit BFS/DFS)
5. Es dürfen sich keine zwei Kanten schneiden.

ETSP – zulässige Teillösungen

Betrachte eine beliebige Reihenfolge der Kanten. Sei $x_i \in \{0,1\}$ die Entscheidungsvariable für die i-te Kante.
Wann ist eine Auswahl von Kanten nicht zulässig?

1. Maximal n Kanten ausgewählt, also
   $\sum_i x_i \leq n$
2. An jedem Punkt v dürfen nur zwei Kanten liegen, also
   $\sum_{i \in \delta(v)} x_i \leq 2$
3. Falls $< n$ Kanten ausgewählt sind, darf kein Kreis existieren. (Erkennbar mit BFS/DFS aus AuD)
4. Falls $= n$ Kanten ausgewählt sind, muss exakt ein Kreis existieren. (Wieder mit BFS/DFS)
5. Es dürfen sich keine zwei Kanten schneiden.



Die grünen bzw. braunen Kanten sind höchstens so lang wie die schwarzen.
Grund: Dreiecksungleichung
$d(a,b) + d(b,c) \geq d(a,c)$

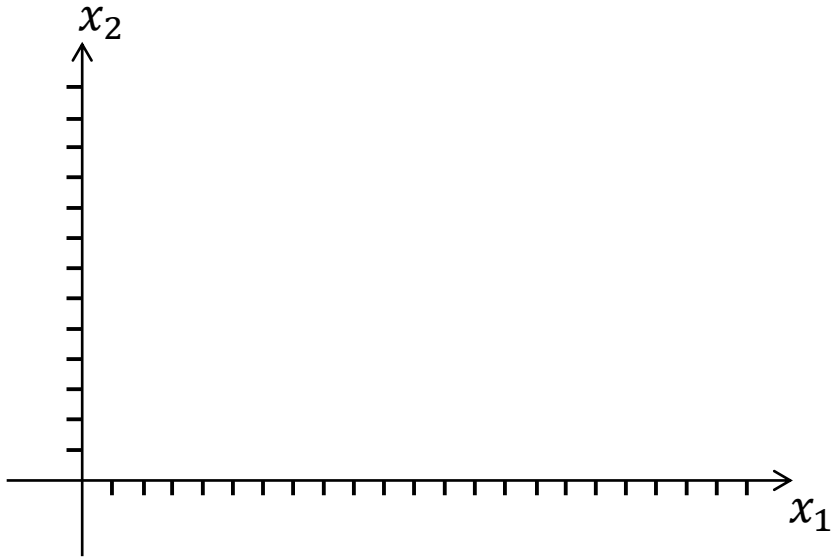
ETSP – Linear/Integer Programming

Wie löst man ETSP in der Praxis? Dazu eignet sich lineare Optimierung. Man stellt ein (Un-)Gleichungssystem mit m Variablen in der folgenden Form auf:

$$\begin{aligned} &\min c^T x \\ &\text{s.t. } Ax \leq b \\ &\qquad x \in \mathbb{R}^m \end{aligned}$$

Beispiel:

$$\min 2x_1 - 5x_2$$



ETSP – Linear/Integer Programming

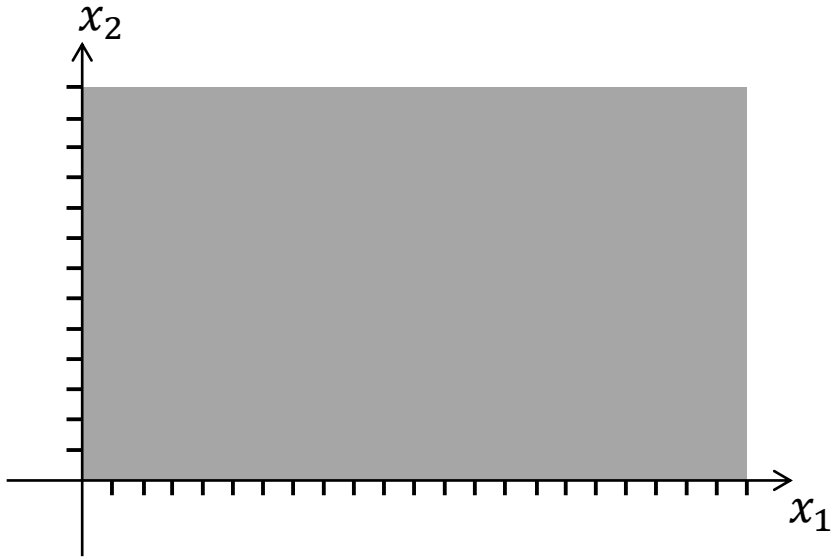
Wie löst man ETSP in der Praxis? Dazu eignet sich lineare Optimierung. Man stellt ein (Un-)Gleichungssystem mit m Variablen in der folgenden Form auf:

$$\min c^T x$$
$$\text{s.t. } Ax \leq b$$
$$x \in \mathbb{R}^m$$

Beispiel:

$$\min 2x_1 - 5x_2$$

$$x_1, x_2 \geq 0$$



ETSP – Linear/Integer Programming

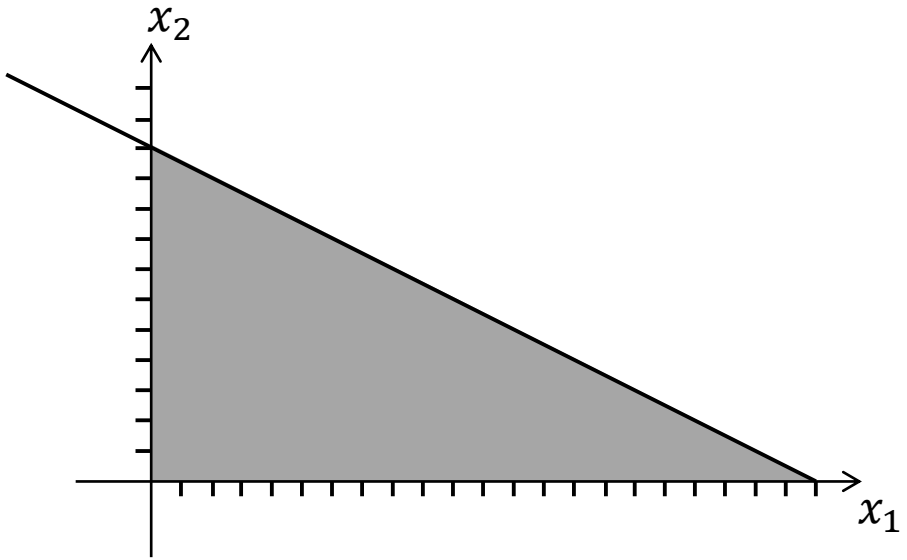
Wie löst man ETSP in der Praxis? Dazu eignet sich lineare Optimierung. Man stellt ein (Un-)Gleichungssystem mit m Variablen in der folgenden Form auf:

$$\min c^T x$$
$$\text{s.t. } Ax \leq b$$
$$x \in \mathbb{R}^m$$

Beispiel:

$$\min 2x_1 - 5x_2$$
$$\text{s.t. } x_1 + 2x_2 \leq 22$$

$$x_1, x_2 \geq 0$$



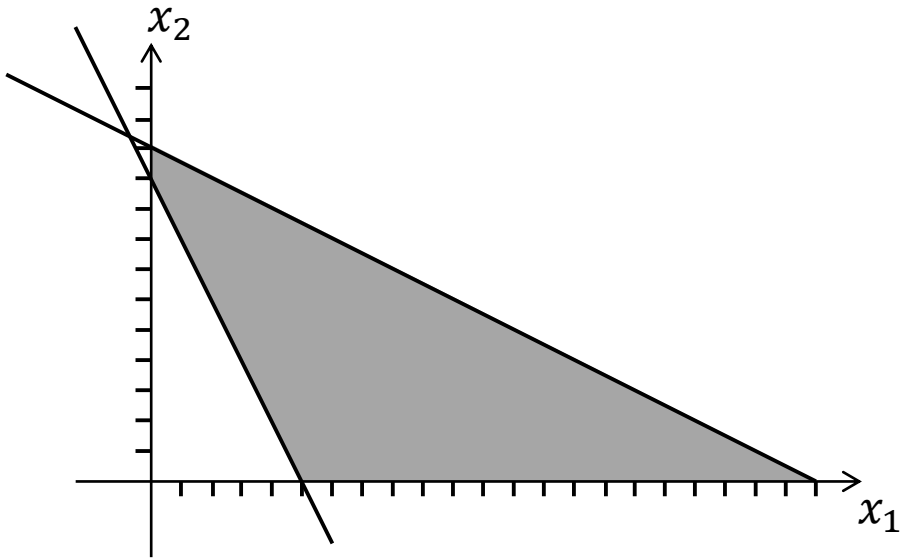
ETSP – Linear/Integer Programming

Wie löst man ETSP in der Praxis? Dazu eignet sich lineare Optimierung. Man stellt ein (Un-)Gleichungssystem mit m Variablen in der folgenden Form auf:

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t. } Ax \leq b \\ &\qquad x \in \mathbb{R}^m \end{aligned}$$

Beispiel:

$$\begin{aligned} &\min\ 2x_1 - 5x_2 \\ &\text{s.t.}\quad x_1 + 2x_2 \leq 22 \\ &\qquad 2x_1 + \ x_2 \geq 10 \\ &\qquad x_1, x_2 \geq 0 \end{aligned}$$



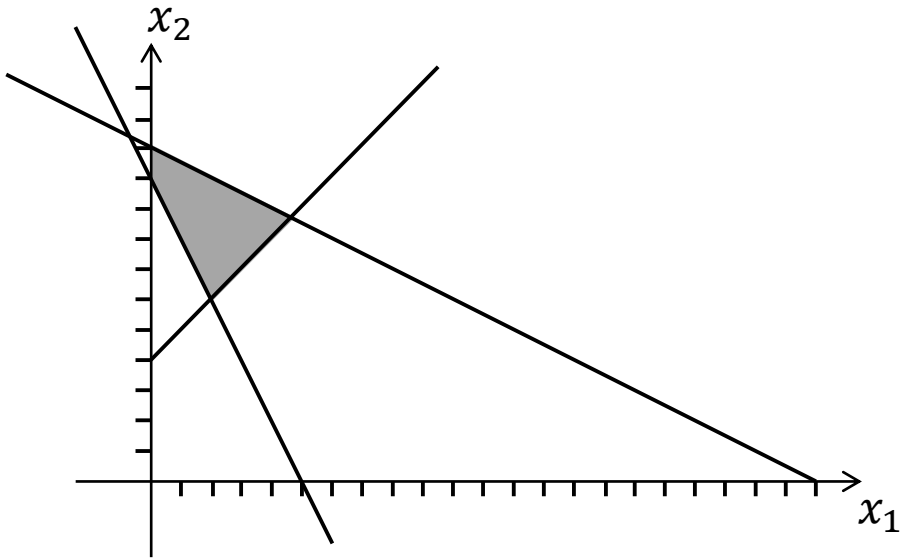
ETSP – Linear/Integer Programming

Wie löst man ETSP in der Praxis? Dazu eignet sich lineare Optimierung. Man stellt ein (Un-)Gleichungssystem mit m Variablen in der folgenden Form auf:

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t. } Ax \leq b \\ &\qquad x \in \mathbb{R}^m \end{aligned}$$

Beispiel:

$$\begin{aligned} &\min\ 2x_1 - 5x_2 \\ &\text{s.t.}\quad x_1 + 2x_2 \leq 22 \\ &\qquad 2x_1 + x_2 \geq 10 \\ &\qquad -x_1 + x_2 \geq 4 \\ &\qquad x_1, x_2 \geq 0 \end{aligned}$$



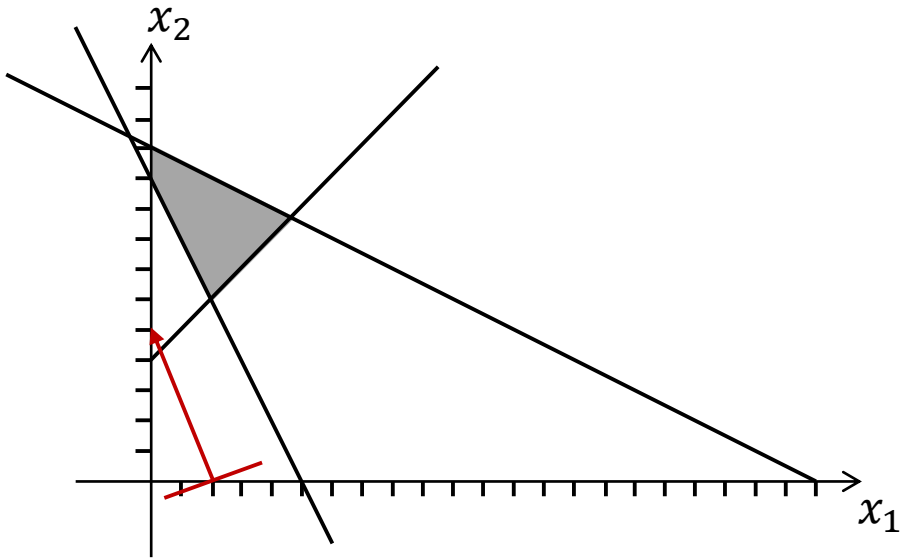
ETSP – Linear/Integer Programming

Wie löst man ETSP in der Praxis? Dazu eignet sich lineare Optimierung. Man stellt ein (Un-)Gleichungssystem mit m Variablen in der folgenden Form auf:

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t. } Ax \leq b \\ &\qquad x \in \mathbb{R}^m \end{aligned}$$

Beispiel:

$$\begin{aligned} &\min\ 2x_1 - 5x_2 \\ &\text{s.t.}\quad x_1 + 2x_2 \leq 22 \\ &\qquad 2x_1 + x_2 \geq 10 \\ &\qquad -x_1 + x_2 \geq 4 \\ &\qquad x_1, x_2 \geq 0 \end{aligned}$$



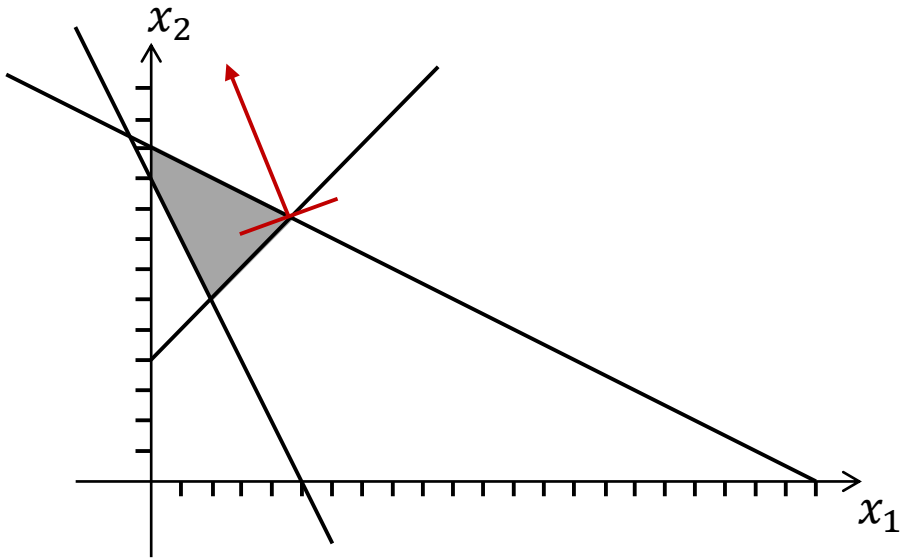
ETSP – Linear/Integer Programming

Wie löst man ETSP in der Praxis? Dazu eignet sich lineare Optimierung. Man stellt ein (Un-)Gleichungssystem mit m Variablen in der folgenden Form auf:

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t. } Ax \leq b \\ &\qquad x \in \mathbb{R}^m \end{aligned}$$

Beispiel:

$$\begin{aligned} &\min\ 2x_1 - 5x_2 \\ &\text{s.t.}\quad x_1 + 2x_2 \leq 22 \\ &\qquad 2x_1 + x_2 \geq 10 \\ &\qquad -x_1 + x_2 \geq 4 \\ &\qquad x_1, x_2 \geq 0 \end{aligned}$$



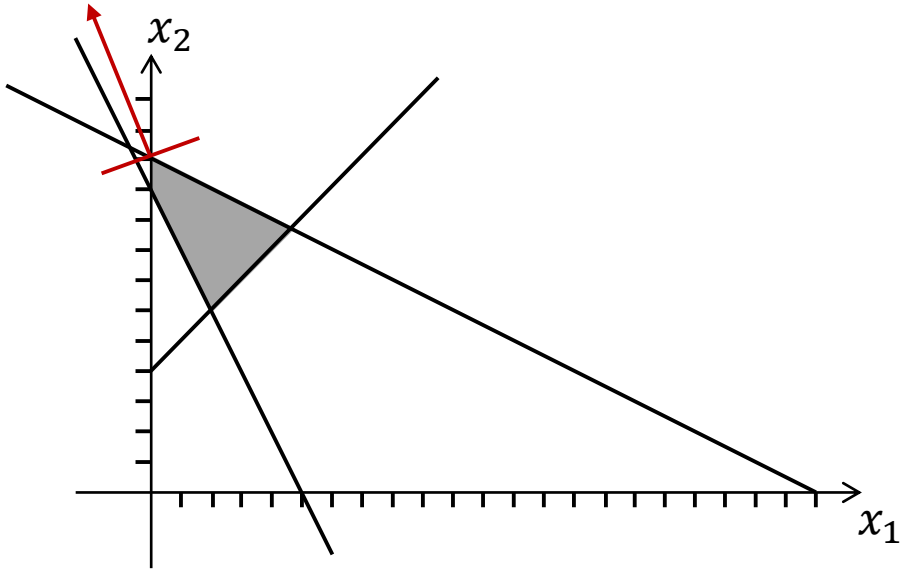
ETSP – Linear/Integer Programming

Wie löst man ETSP in der Praxis? Dazu eignet sich lineare Optimierung. Man stellt ein (Un-)Gleichungssystem mit m Variablen in der folgenden Form auf:

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t. } Ax \leq b \\ &\qquad x \in \mathbb{R}^m \end{aligned}$$

Beispiel:

$$\begin{aligned} &\min\ 2x_1 - 5x_2 \\ &\text{s.t.}\quad x_1 + 2x_2 \leq 22 \\ &\qquad 2x_1 + x_2 \geq 10 \\ &\qquad -x_1 + x_2 \geq 4 \\ &\qquad x_1, x_2 \geq 0 \end{aligned}$$



ETSP – Linear/Integer Programming

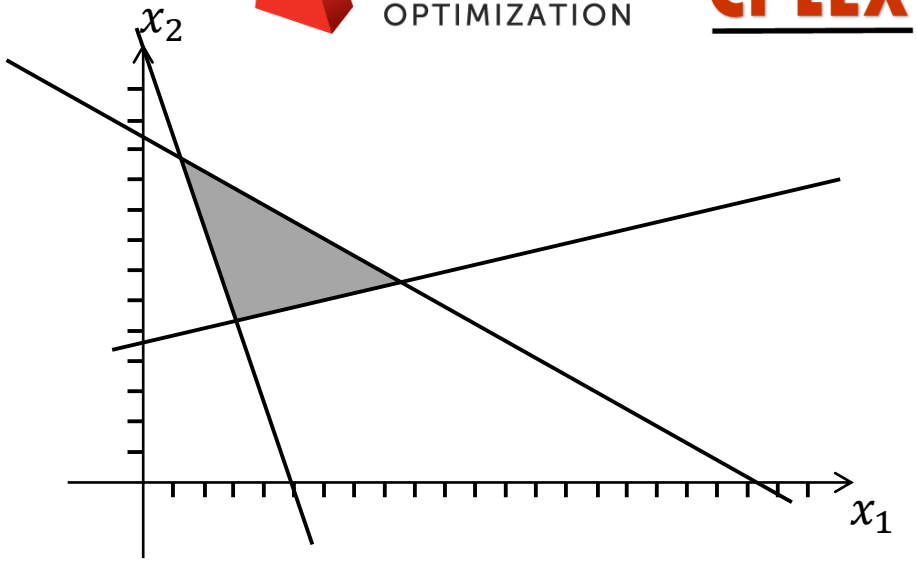
Für ETSP:

$$\begin{aligned} \min\ & \sum_{e \in E} c_e x_e \\ \text{s.t. } & \sum_{e \in \delta(v)} x_e = 2,\ \forall v \in P \\ & \sum_{e \in \delta(S)} x_e \geq 2,\ \forall \emptyset \neq S \subsetneq P \\ & x \in \{0,1\}^m \end{aligned}$$

Solver für LPs/IPs:

GUROBI OPTIMIZATION

IBM CPLEX



ETSP – Linear/Integer Programming

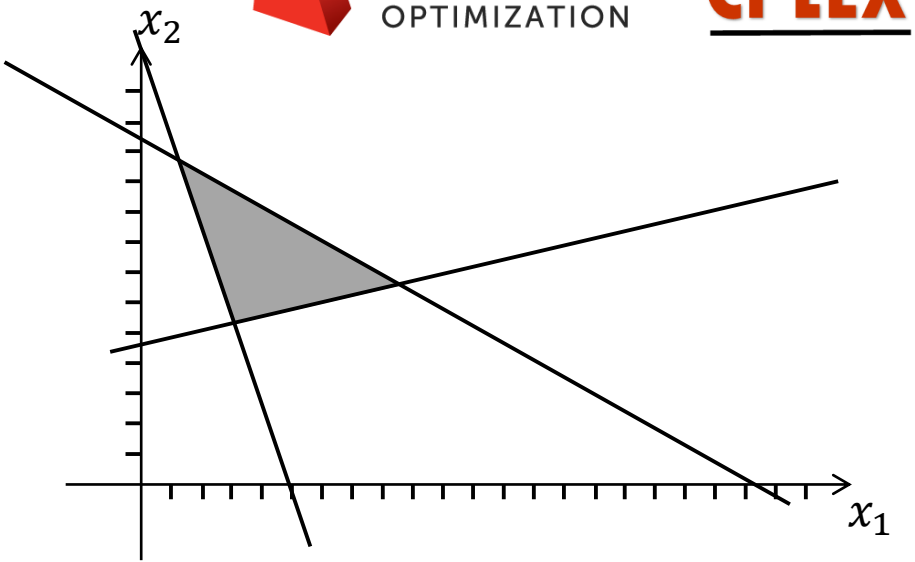
Für ETSP:

$$\min \sum_{e \in E} c_e x_e$$
$$\text{s.t. } \sum_{e \in \delta(v)} x_e = 2, \forall v \in P$$
$$\sum_{e \in \delta(S)} x_e \geq 2, \forall \emptyset \neq S \subsetneq P$$
$$x \in \{0,1\}^m$$

Solver für LPs/IPs:

GUROBI OPTIMIZATION

IBM CPLEX



ETSP – Linear/Integer Programming

Für ETSP:

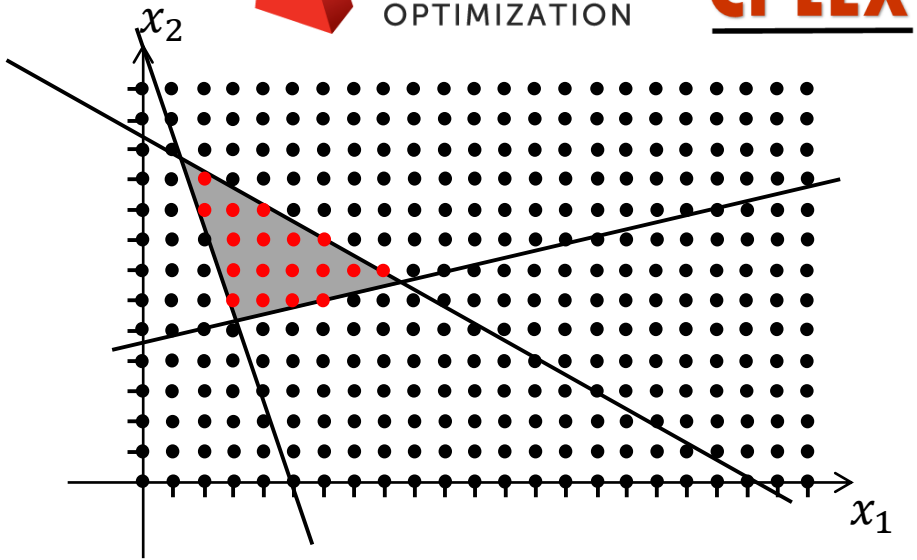
$$\min \sum_{e \in E} c_e x_e$$
$$\text{s.t. } \sum_{e \in \delta(v)} x_e = 2, \forall v \in P$$
$$\sum_{e \in \delta(S)} x_e \geq 2, \forall \emptyset \neq S \subsetneq P$$
$$x \in \{0,1\}^m$$

Wie findet man die roten Punkte?
Antwort: Branch-and-Cut

Solver für LPs/IPs:

GUROBI OPTIMIZATION

IBM CPLEX



ETSP – Linear/Integer Programming

Für ETSP:

$$\min \sum_{e \in E} c_e x_e$$
$$\text{s.t. } \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in P$$
$$\sum_{e \in \delta(S)} x_e \geq 2, \ \forall \emptyset \neq S \subsetneq P$$
$$x \in \{0,1\}^m$$

Wie findet man die roten Punkte?
Antwort: Branch-and-Cut

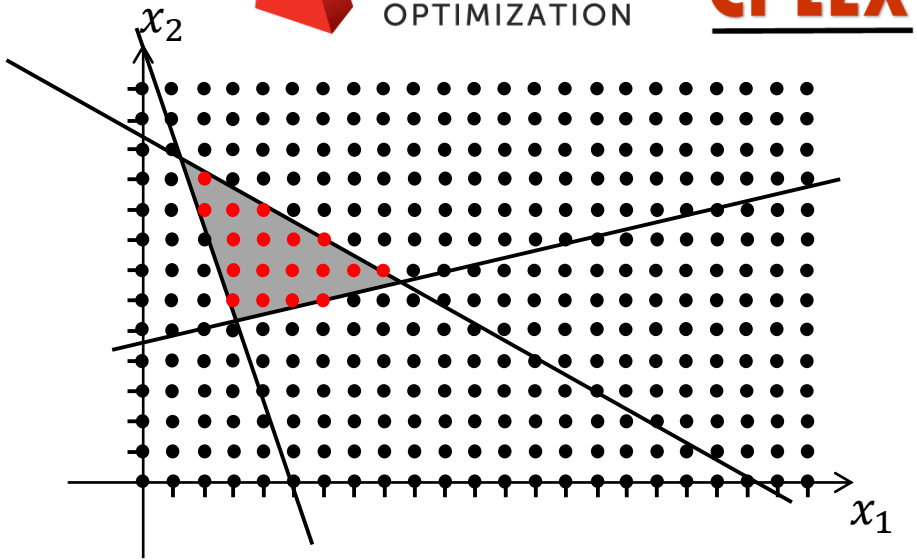
Für eine untere Schranke benutzt man $x \in [0,1]^m$ statt $x \in \{0,1\}^m$

Wie man solche LPs/IPs lösen kann, erfahrt ihr in MMA!

Solver für LPs/IPs:

GUROBI OPTIMIZATION

IBM CPLEX



ETSP – Wie gut seid ihr?

Human performance on the traveling salesman problem

J. N. MacGREGOR and T. ORMEROD
Loughborough University of Technology, Loughborough, England

Two experiments on performance on the traveling salesman problem (TSP) are reported. The TSP consists of finding the shortest path through a set of points, returning to the origin. It appears to be an intransigent mathematical problem, and heuristics have been developed to find approximate solutions. The first experiment used 10-point, the second, 20-point problems. The experiments tested the hypothesis that complexity of TSPs is a function of number of nonboundary points, not total number of points. Both experiments supported the hypothesis. The experiments provided information on the quality of subjects' solutions. Their solutions clustered close to the best known solutions, were an order of magnitude better than solutions produced by three well-known heuristics, and on average fell beyond the 99.9th percentile in the distribution of random solutions. The solution process appeared to be perceptually based.

Table 5
Minimum Observed Path Length and Percentage Above the Minimum for Human and Heuristic Solutions, Experiment 2

Number of Interior Points	Optimal Solution	Percentage Above Optimal					
		Subject Minimum	NN Minimum	LA	CHCI	Subject Mean	NN Mean
4	703.81	3.0	4.1	3.1	2.5	5.4	8.4
6	703.89	1.2	1.2	20.1	5.3	7.3	8.8
8	725.31	1.7	3.3	13.7	5.8	5.2	9.4
10	698.83	1.4	0	6.0	4.8	3.3	9.5
12	688.33	0.6	0	23.4	3.6	4.6	7.1
14	663.61	1.9	1.5	10.0	4.4	9.6	15.4
16	593.81	1.0	15.0	0.8	0.05	8.5	21.7

Note—NN, Nearest Neighbor; LA, Largest Interior Angle; CHCI, Convex Hull, with cheapest insertion criterion.

Jetzt seid ihr dran!