Inhaltsübersicht für heute:

Anwendungsbeispiel: Rundreiseprobleme (TSP)

Finden „guter“ Lösungen, Heuristiken

Gemischt-ganzzahlige Optimierung

Inhaltsübersicht für heute:

Anwendungsbeispiel: Rundreiseprobleme (TSP)

Finden „guter“ Lösungen, Heuristiken

Gemischt-ganzzahlige Optimierung

Anwendungsbeispiel: Rundreiseprobleme (TSP)

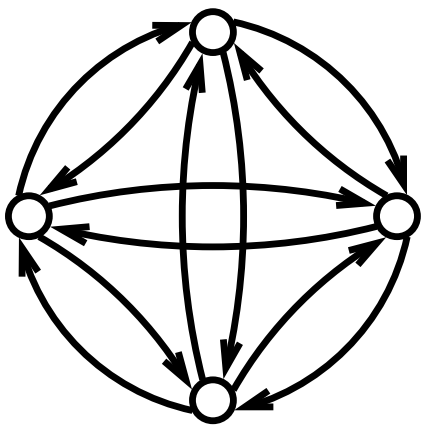
Problem des Handelsreisenden: Gegeben n Städte mit allen paarweisen Distanzen, finde eine kürzeste Rundreise, die jede genau einmal besucht.

Anwendungsbeispiel: Rundreiseprobleme (TSP)

Problem des Handelsreisenden: Gegeben n Städte mit allen paarweisen Distanzen, finde eine kürzeste Rundreise, die jede genau einmal besucht.

Komb. Opt.: $D = (V, E = \{(u, v) : u, v \in V, u \neq v\})$ vollst. Digraph, Kosten $c \in \mathbb{R}^E$, zul. Menge $\mathcal{F} = \{R \subset E : R \text{ (ger.) Kreis in } D, |R| = n\}$.
Finde $R \in \operatorname{Argmin}\{c(R) = \sum_{e \in R} c_e : R \in \mathcal{F}\}$

NP-vollständiges Problem
Anzahl Rundreisen?



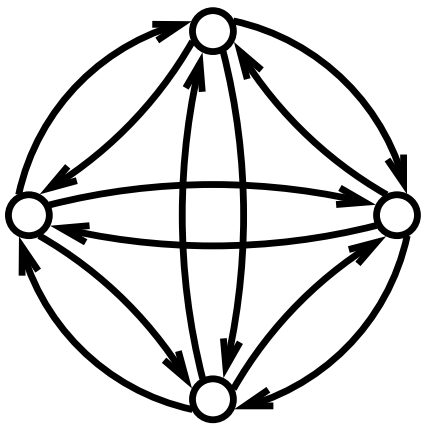
Anwendungsbeispiel: Rundreiseprobleme (TSP)

Problem des Handelsreisenden: Gegeben n Städte mit allen paarweisen Distanzen, finde eine kürzeste Rundreise, die jede genau einmal besucht.

Komb. Opt.: $D = (V, E = \{(u, v) : u, v \in V, u \neq v\})$ vollst. Digraph, Kosten $c \in \mathbb{R}^E$, zul. Menge $\mathcal{F} = \{R \subset E : R \text{ (ger.) Kreis in } D, |R| = n\}$.
Finde $R \in \operatorname{Argmin}\{c(R) = \sum_{e \in R} c_e : R \in \mathcal{F}\}$

NP-vollständiges Problem
Anzahl Rundreisen? $(n-1)!$

$\rightarrow$kombinatorische Explosion



$n = 3$:	2
$n = 4$:	$3 \cdot 2 = 6$
$n = 5$:	$4 \cdot 3 \cdot 2 = 24$
$n = 10$:	362880
$n = 11$:	3628800
$n = 12$:	39916800
$n = 13$:	479001600
$n = 14$:	6227020800
$n = 100$:	10^{158}

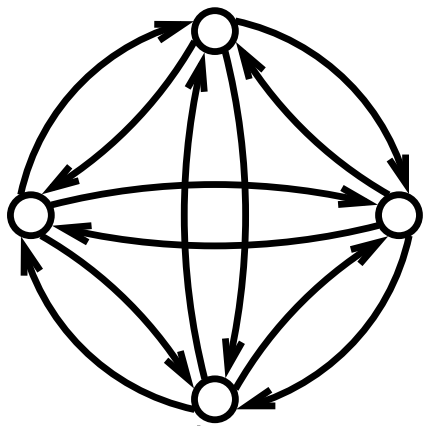
Anwendungsbeispiel: Rundreiseprobleme (TSP)

Problem des Handelsreisenden: Gegeben n Städte mit allen paarweisen Distanzen, finde eine kürzeste Rundreise, die jede genau einmal besucht.

Komb. Opt.: $D = (V, E = \{(u, v) : u, v \in V, u \neq v\})$ vollst. Digraph, Kosten $c \in \mathbb{R}^E$, zul. Menge $\mathcal{F} = \{R \subset E : R \text{ (ger.) Kreis in } D, |R| = n\}$.
Finde $R \in \operatorname{Argmin}\{c(R) = \sum_{e \in R} c_e : R \in \mathcal{F}\}$

NP-vollständiges Problem
Anzahl Rundreisen? $(n-1)!$

$\rightarrow$kombinatorische Explosion



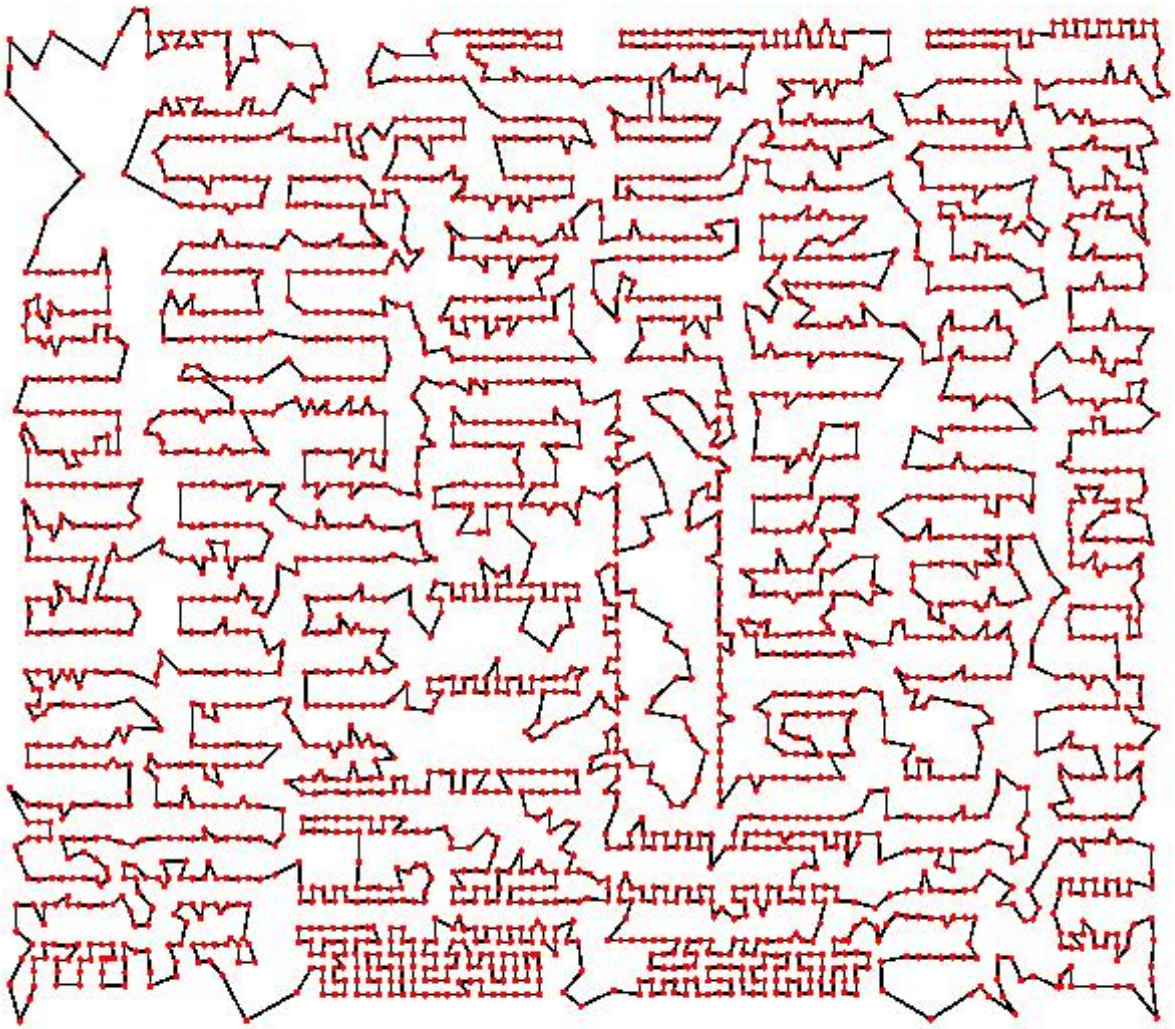
$n = 3$:	2
$n = 4$:	$3 \cdot 2 = 6$
$n = 5$:	$4 \cdot 3 \cdot 2 = 24$
$n = 10$:	362880
$n = 11$:	3628800
$n = 12$:	39916800
$n = 13$:	479001600
$n = 14$:	6227020800
$n = 100$:	10^{158}

Typisches Grundproblem in:

- Zustelldiensten (Paket-, Arznei-Transporte)
- Taxiservice, Havarie-Dienste
- Auftragsplanung mit Rüstkosten [z.B. Autos färben]
- Robotersteuerung (meist nichtlinear)

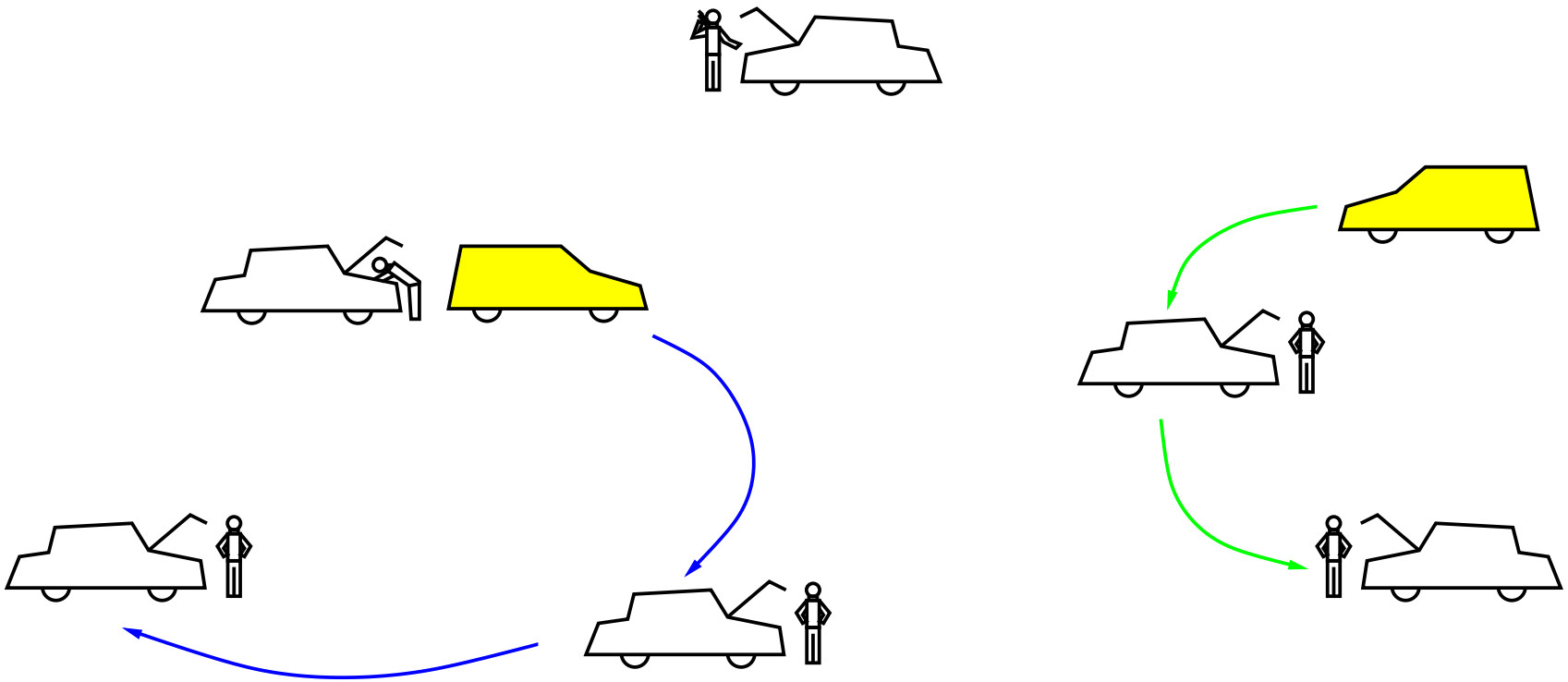
[oft mit mehreren „Fahrzeugen“ und Zeitfenstern]

Löcher in Platinen bohren



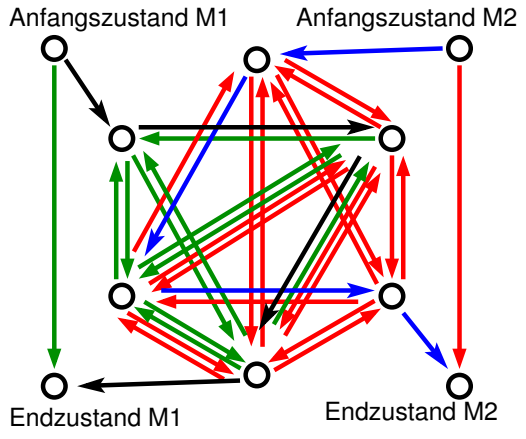
[http://www.math.princeton.edu/tsp]

Mit Online-Aspekten: Pannenhilfe

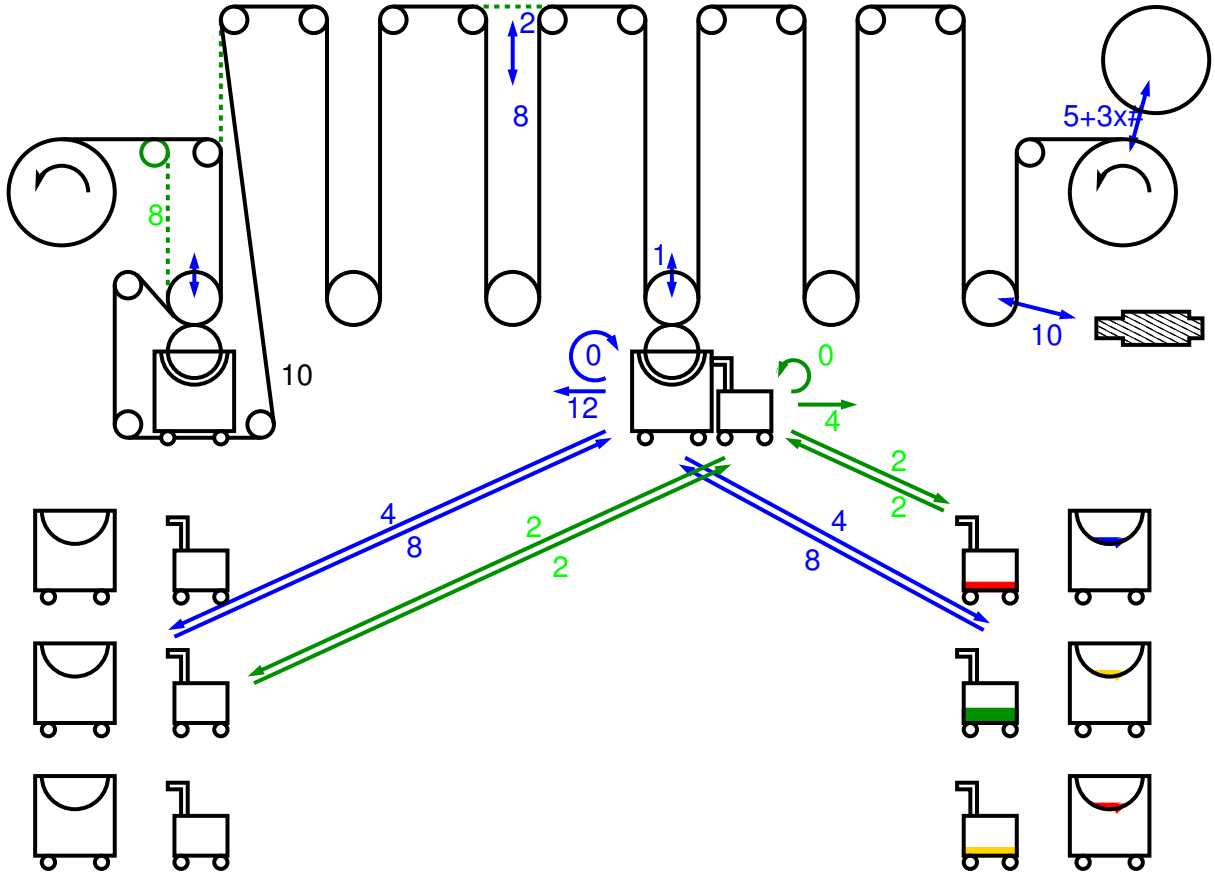


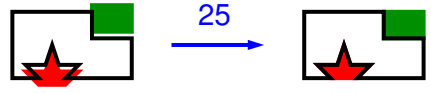
Bestimme Autozuweisung und Reihenfolge für jedes Pannenfahrzeug so, dass bereits versprochene Wartezeiten nicht überschritten werden.

Reihenfolgeoptimierung bei langen Rüstzeiten



Zwei Rotationstiefdruckmaschinen
für den Druck von Geschenkpapier



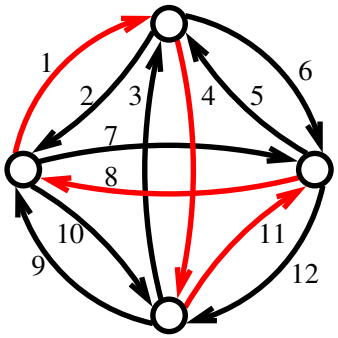


Passer

Farbanpassung neue Farben: 20

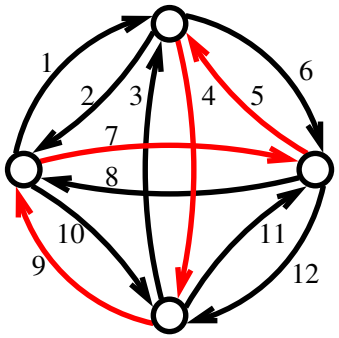
Modellierung als ganzzahliges Programm

Abstrakte Formulierung über konvexe Hülle der Inzidenzvektoren:



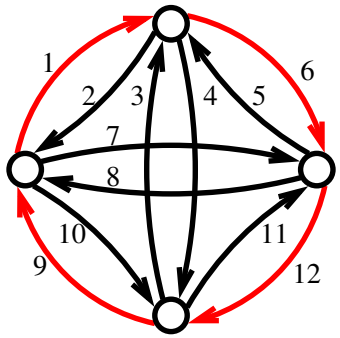
$R_1 = \{1, 4, 8, 11\}$

$$\chi(R_1) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



$R_2 = \{4, 5, 7, 9\}$

$$\chi(R_2) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



$R_3 = \{1, 6, 9, 12\}$

$$\chi(R_3) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Für Kantenlängen $c \in \mathbb{R}^E$ ist die Länge von Rundreise R: $\sum_{e \in R} c_e = c^T \chi(R)$.

(TSP) min $c^T x$ s.t. $x \in \operatorname{conv}\{\chi(R) : R \text{ Rundreise in } D = (V, E)\} =: P_{TSP}$

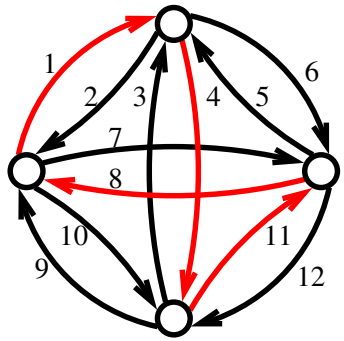
Wäre exakt, aber keine lineare Beschreibung $A_I x \leq b_I$ von P_{TSP} bekannt!

Ganzzahlige Formulierung für (TSP)

Ziel: P_{TSP} durch ein größeres Polytop $P = \{x \in \mathbb{R}^n : Ax \leq b\} \supseteq P_{TSP}$ möglichst eng einfassen, sodass wenigstens $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ Rundreise}\}$.

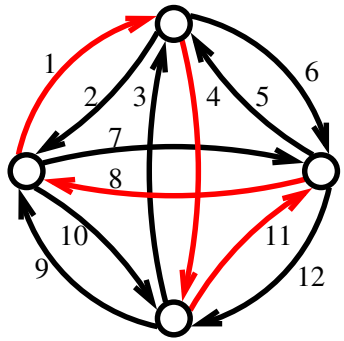
Eine Gleichung/Ungleichung heißt gültig für P_{TSP}, wenn sie für alle $x \in \{\chi(R) : R \text{ Rundreise}\}$ erfüllt ist.

Vorschläge?



Ganzzahlige Formulierung für (TSP)

Ziel: P_{TSP} durch ein größeres Polytop $P = \{x \in \mathbb{R}^n : Ax \leq b\} \supseteq P_{TSP}$ möglichst eng einfassen, sodass wenigstens $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ Rundreise}\}$.



Eine Gleichung/Ungleichung heißt gültig für P_{TSP}, wenn sie für alle $x \in \{\chi(R) : R \text{ Rundreise}\}$ erfüllt ist.

0-1 Würfel: $0 \leq x \leq 1$ ist gültig

Gradgleichungen:
aus jedem und in jeden Knoten geht genau ein Pfeil,

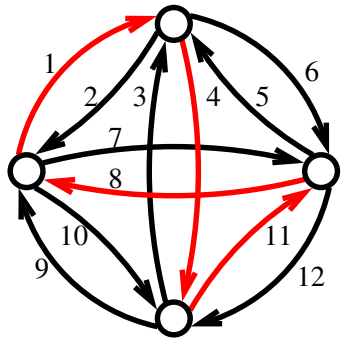
$$\text{für } v \in V: \quad \sum_{(v,u)\in E} x_{(v,u)} = 1, \quad \sum_{(u,v)\in E} x_{(u,v)} = 1$$

(pro Zeile/Spalte genau eine 1 $\leftrightarrow$ Zuweisungsproblem)

Schon Formulierung? $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ Rundreise}\}$?

Ganzzahlige Formulierung für (TSP)

Ziel: P_{TSP} durch ein größeres Polytop $P = \{x \in \mathbb{R}^n : Ax \leq b\} \supseteq P_{TSP}$ möglichst eng einfassen, sodass wenigstens $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ Rundreise}\}$.



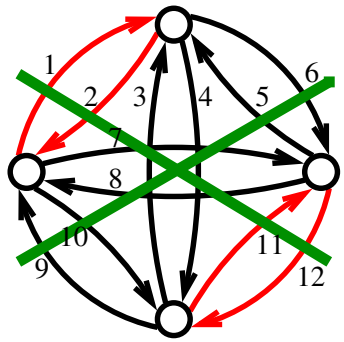
Eine Gleichung/Ungleichung heißt gültig für P_{TSP}, wenn sie für alle $x \in \{\chi(R) : R \text{ Rundreise}\}$ erfüllt ist.

0-1 Würfel: $0 \leq x \leq 1$ ist gültig

Gradgleichungen:
aus jedem und in jeden Knoten geht genau ein Pfeil,

$$\text{für } v \in V: \quad \sum_{(v,u)\in E} x_{(v,u)} = 1, \quad \sum_{(u,v)\in E} x_{(u,v)} = 1$$

(pro Zeile/Spalte genau eine 1 $\leftrightarrow$ Zuweisungsproblem)



Schon Formulierung? $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ Rundreise}\}$?

Kurzzyklusungleichungen:
Aus jeder Knoten-Teilmenge muss mindestens eine Kante hinausführen,

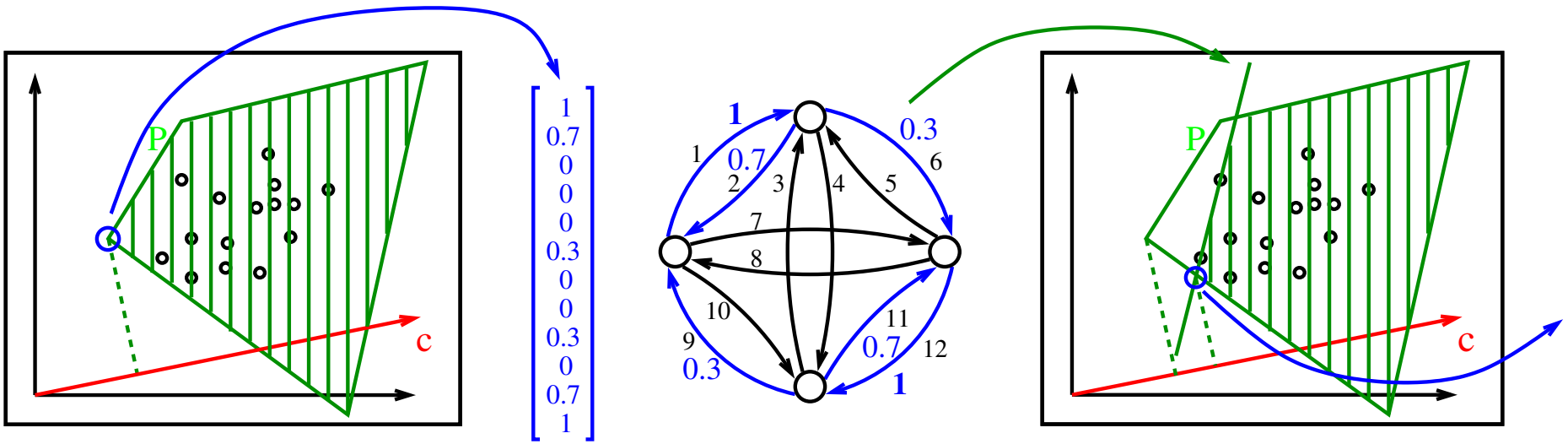
$$\text{für } S \subset V, 2 \leq |S| \leq n-2: \quad \sum_{e\in\delta^+(S)} x_e \geq 1$$

Ist nun eine Formulierung, aber mit etwa 2^n Ungleichungen!

Lösen der TSP LP-Relaxation

Geht nur mit Schnittebenenverfahren:
Die 1. Relaxation bildet das Zuweisungsproblem (Box+Gradgleichungen)
Dessen Lösung ist ganzzahlig und besteht meist aus getrennten Kreisen.

Ab jetzt die Schranke iterativ durch Kurzzyklus-Schnittebenen verbessern

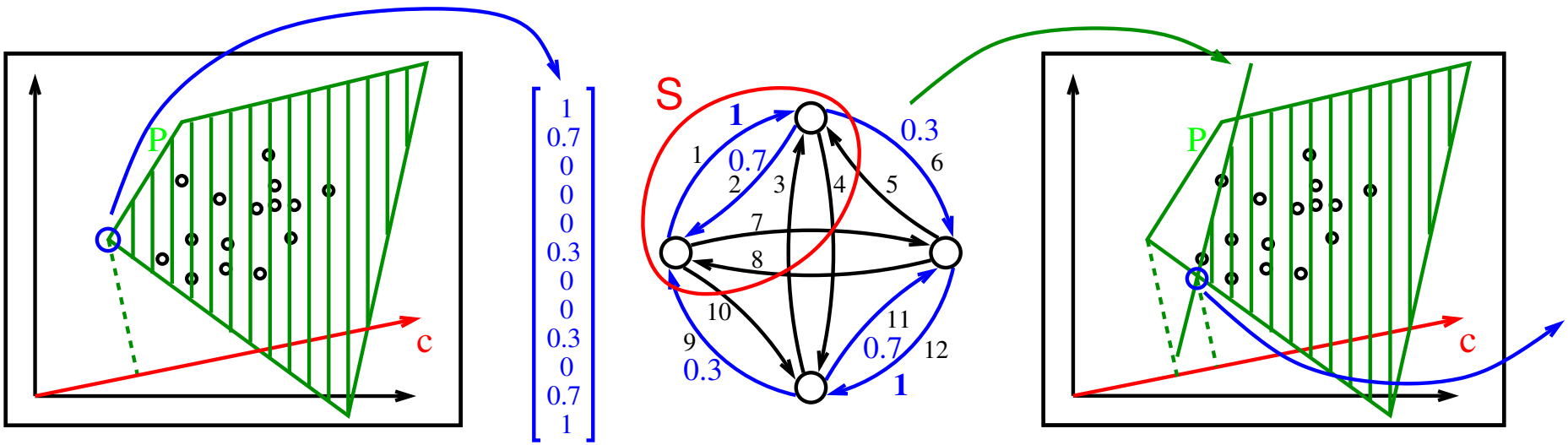


Separationsproblem: Finde ein $S \subset V, 2 \leq |S| \leq n-2$: $\sum_{e \in \delta^+(S)} x_e \not\geq 1$.

Lösen der TSP LP-Relaxation

Geht nur mit Schnittebenenverfahren:
Die 1. Relaxation bildet das Zuweisungsproblem (Box+Gradgleichungen)
Dessen Lösung ist ganzzahlig und besteht meist aus getrennten Kreisen.

Ab jetzt die Schranke iterativ durch Kurzzyklus-Schnittebenen verbessern



Separationsproblem: Finde ein $S \subset V, 2 \leq |S| \leq n-2 : \quad \sum_{e \in \delta^+(S)} x_e \not\geq 1.$

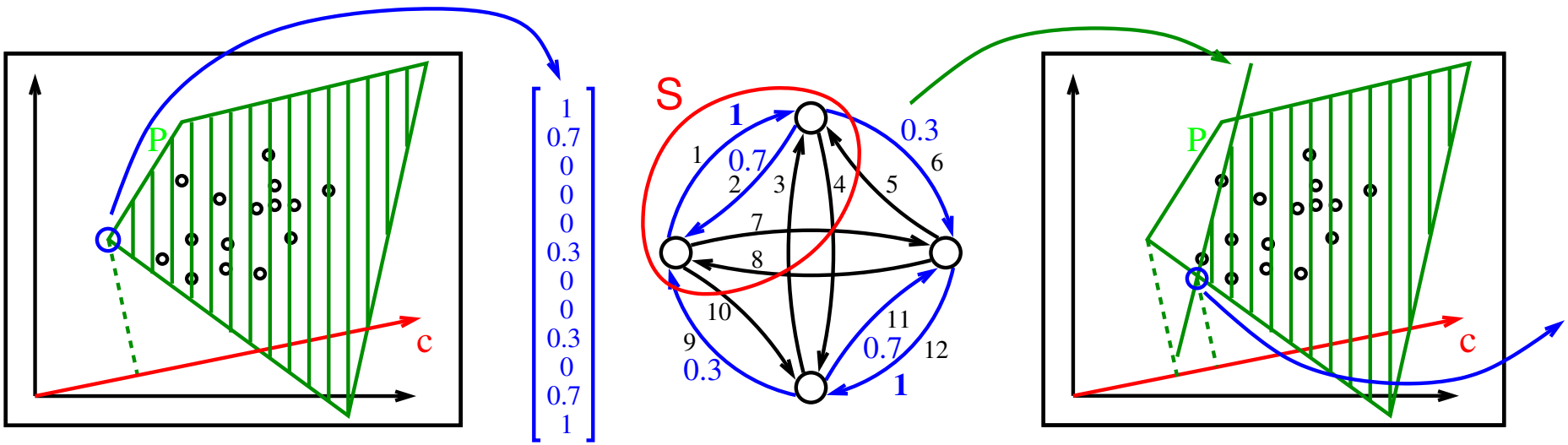
$\Leftrightarrow$ Finde im Netzwerk $D = (V, E)$ mit Kantenkap. x einen Schnitt $x(\delta^+(S)) < 1$.

$\rightarrow$ Maximale s-t-Flüsse/minimale s-t-Schnitte für $s, t \in V$, exakt lösbar!

Lösen der TSP LP-Relaxation

Geht nur mit Schnittebenenverfahren:
Die 1. Relaxation bildet das Zuweisungsproblem (Box+Gradgleichungen)
Dessen Lösung ist ganzzahlig und besteht meist aus getrennten Kreisen.

Ab jetzt die Schranke iterativ durch Kurzzyklus-Schnittebenen verbessern



Separationsproblem: Finde ein $S \subset V, 2 \leq |S| \leq n-2 : \quad \sum_{e \in \delta^+(S)} x_e \not\geq 1.$

$\Leftrightarrow$ Finde im Netzwerk $D = (V, E)$ mit Kantenkap. x einen Schnitt $x(\delta^+(S)) < 1$.

$\rightarrow$ Maximale s-t-Flüsse/minimale s-t-Schnitte für $s, t \in V$, exakt lösbar!

Grad+Kurzz. liefern sehr gute Schranken, aber noch lange nicht P_{TSP}!
Schranke mit weiteren Unglg. verbesserbar (Kamm-, etc.),
die Lösung der Relaxation wird aber fast nie ganzzahlig!

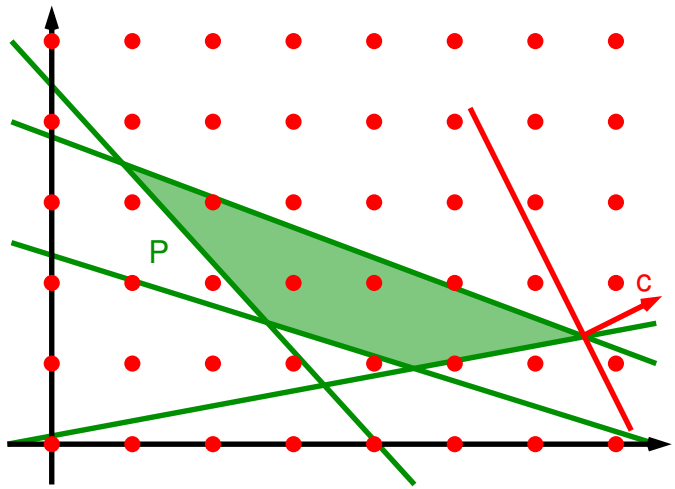
Allgemeine Schnittebenen

Es gibt auch problemübergreifende allgemeine Schnittebenen, die in state-of-the-art Lösern für ganzz. Opt. eingesetzt werden:

- Gomory-Schnitte [Abrunden der Koeffizienten durch $\lfloor \cdot \rfloor$]

  Ist $a^T x \leq \beta$ gültig für $x \in P \cap \mathbb{Z}^n_+$ dann ist wegen $\lfloor a \rfloor^T x \leq a^T x$ auch $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ gültig.

  So eine verletzte Unglg. ist automatisch aus nicht ganzz. OLs erzeugbar.



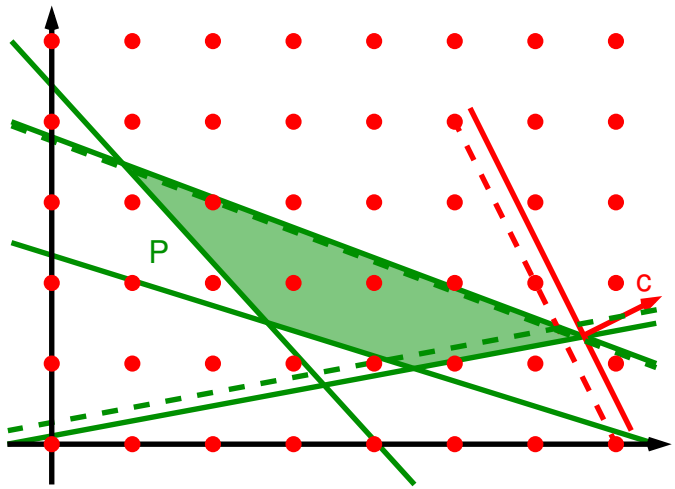
Allgemeine Schnittebenen

Es gibt auch problemübergreifende allgemeine Schnittebenen, die in state-of-the-art Lösern für ganzz. Opt. eingesetzt werden:

- Gomory-Schnitte [Abrunden der Koeffizienten durch $\lfloor \cdot \rfloor$]

  Ist $a^T x \leq \beta$ gültig für $x \in P \cap \mathbb{Z}^n_+$ dann ist wegen $\lfloor a \rfloor^T x \leq a^T x$ auch $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ gültig.

  So eine verletzte Unglg. ist automatisch aus nicht ganzz. OLs erzeugbar.



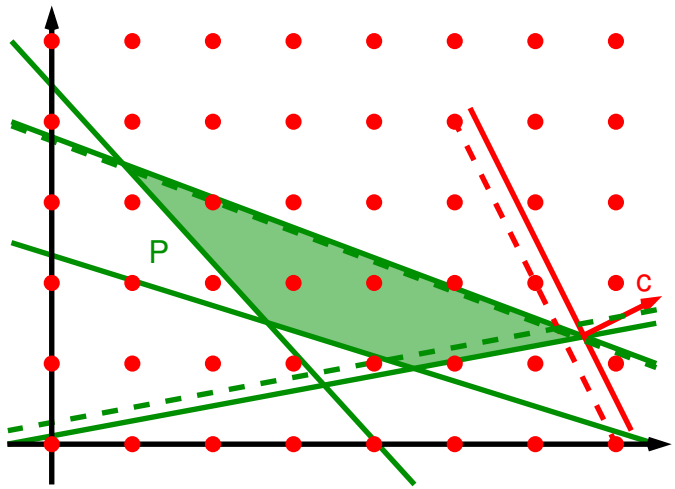
Allgemeine Schnittebenen

Es gibt auch problemübergreifende allgemeine Schnittebenen, die in state-of-the-art Lösern für ganzz. Opt. eingesetzt werden:

- Gomory-Schnitte [Abrunden der Koeffizienten durch $\lfloor \cdot \rfloor$]

  Ist $a^T x \leq \beta$ gültig für $x \in P \cap \mathbb{Z}^n_+$ dann ist wegen $\lfloor a \rfloor^T x \leq a^T x$ auch $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ gültig.

  So eine verletzte Unglg. ist automatisch aus nicht ganzz. OLs erzeugbar.



- Lift-and-Project Cuts,
- Clique-Ungleichungen,

etc.

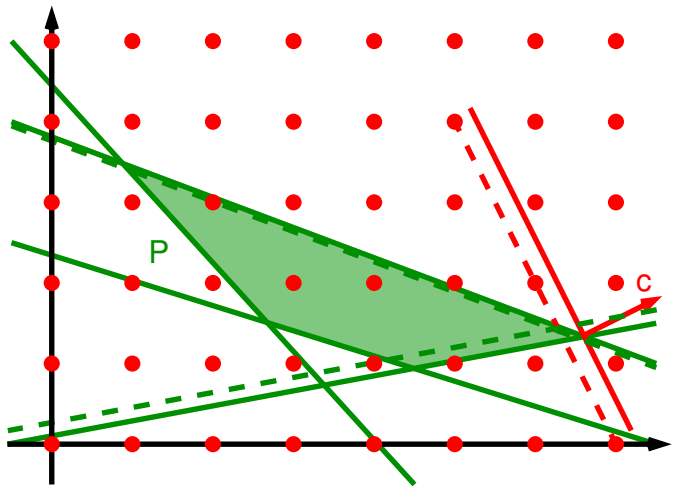
Allgemeine Schnittebenen

Es gibt auch problemübergreifende allgemeine Schnittebenen, die in state-of-the-art Lösern für ganzz. Opt. eingesetzt werden:

- Gomory-Schnitte [Abrunden der Koeffizienten durch $\lfloor \cdot \rfloor$]

  Ist $a^T x \leq \beta$ gültig für $x \in P \cap \mathbb{Z}^n_+$ dann ist wegen $\lfloor a \rfloor^T x \leq a^T x$ auch $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ gültig.

  So eine verletzte Unglg. ist automatisch aus nicht ganzz. OLs erzeugbar.



- Lift-and-Project Cuts,
- Clique-Ungleichungen,

etc.

LP-Relaxation mit Schnittebenen erzeugt gute Schranken (obere für Maximierungs-, untere für Minimierungsprobleme), die Lösungen der Relaxation sind (fast) nie ganzzahlig, führen aber oft in die Nähe guter ganzzahliger Lösungen.

Inhaltsübersicht für heute:

Anwendungsbeispiel: Rundreiseprobleme (TSP)

Finden „guter“ Lösungen, Heuristiken

Gemischt-ganzzahlige Optimierung

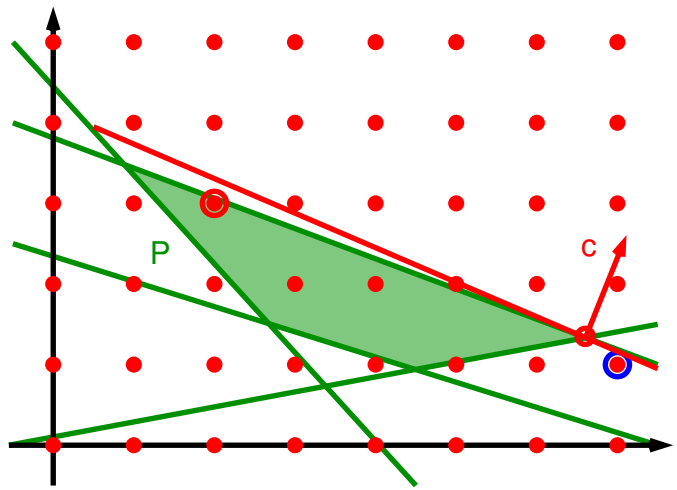
Finden „guter“ Lösungen, Heuristiken

[Heuristik hat den griechischen Wortstamm finden/erfinden]

Für „kleine“ $x \in \mathbb{Z}^n$ liefert normales Runden der LP-Lösung meist unzulässige oder schlechte Lösungen (selbst bei guter Schranke).

Es kann vorkommen, dass kein zulässiger Punkt in der Nähe der LP-Lösung ist!

In state-of-the-art Lösern gibt es aufwendige Standard-Rundeheuristiken (feasibility pump).



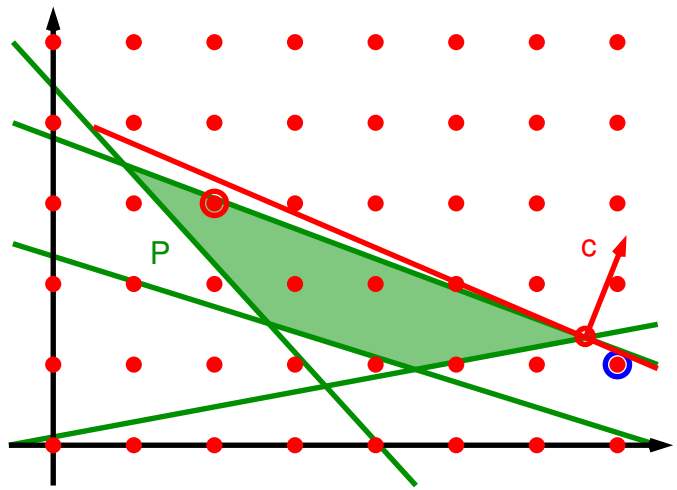
Finden „guter“ Lösungen, Heuristiken

[Heuristik hat den griechischen Wortstamm finden/erfinden]

Für „kleine“ $x \in \mathbb{Z}^n$ liefert normales Runden der LP-Lösung meist unzulässige oder schlechte Lösungen (selbst bei guter Schranke).

Es kann vorkommen, dass kein zulässiger Punkt in der Nähe der LP-Lösung ist!

In state-of-the-art Lösern gibt es aufwendige Standard-Rundeheuristiken (feasibility pump).



Generell: Ganzzahlige Probleme sind meist *NP*-schwer, haben viele „lokale Optima“ (keine benachbarte bessere Lösung) und es ist unwahrscheinlich, dass man die Optimallösung ohne Enumeration findet. Es kann sogar schwer sein, überhaupt eine zulässige Lösung zu finden!

$\rightarrow$ In Anwendungen nützt man möglichst viel problemspezifisches Wissen!

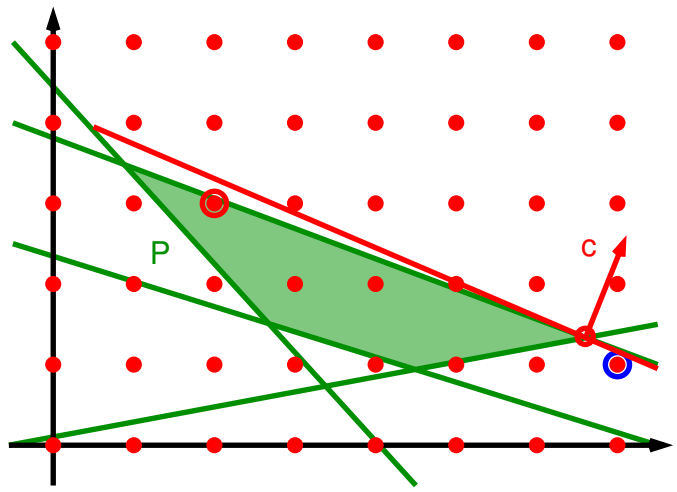
Finden „guter“ Lösungen, Heuristiken

[Heuristik hat den griechischen Wortstamm finden/erfinden]

Für „kleine“ $x \in \mathbb{Z}^n$ liefert normales Runden der LP-Lösung meist unzulässige oder schlechte Lösungen (selbst bei guter Schranke).

Es kann vorkommen, dass kein zulässiger Punkt in der Nähe der LP-Lösung ist!

In state-of-the-art Lösern gibt es aufwendige Standard-Rundeheuristiken (feasibility pump).



Generell: Ganzzahlige Probleme sind meist *NP*-schwer, haben viele „lokale Optima“ (keine benachbarte bessere Lösung) und es ist unwahrscheinlich, dass man die Optimallösung ohne Enumeration findet. Es kann sogar schwer sein, überhaupt eine zulässige Lösung zu finden!

$\rightarrow$ In Anwendungen nützt man möglichst viel problemspezifisches Wissen!

Grober Ablauf:

- (zulässige?) Startlösung erzeugen (meist aus der LP-Lösung)
- Iterative Verbesserung der Lösung durch lokale Suche (lokal exakt, Simulated Annealing, Tabu Search, Genetische Algorithmen, etc.)

Startlösung aus der LP-Relaxation

Typische Ansätze:

- Oft darf aus mehreren $\{0,1\}$-Variablen nur eine gewählt werden:

  LP-Relaxation: $\sum_{i \in N} x_i = 1,\ x_i \in [0,1]$

  Interpretiere x_i als Wahrscheinlichkeit, dass x_i auf 1 gesetzt werden soll und erzeuge damit mehrere Lösungen zufällig, nimm die beste.

Startlösung aus der LP-Relaxation

Typische Ansätze:

- Oft darf aus mehreren $\{0,1\}$-Variablen nur eine gewählt werden:

  LP-Relaxation: $\sum_{i \in N} x_i = 1,\ x_i \in [0,1]$

  Interpretiere x_i als Wahrscheinlichkeit, dass x_i auf 1 gesetzt werden soll und erzeuge damit mehrere Lösungen zufällig, nimm die beste.
- Abweichungen von Nebenbedingungen mit großen Dualvariablen erzeugen große Verluste im Zielfunktionswert.

  $\Rightarrow$ Versuche, diese beim Runden zu erfüllen.

Startlösung aus der LP-Relaxation

Typische Ansätze:

- Oft darf aus mehreren $\{0,1\}$-Variablen nur eine gewählt werden:

  LP-Relaxation: $\sum_{i \in N} x_i = 1,\ x_i \in [0,1]$

  Interpretiere x_i als Wahrscheinlichkeit, dass x_i auf 1 gesetzt werden soll und erzeuge damit mehrere Lösungen zufällig, nimm die beste.
- Abweichungen von Nebenbedingungen mit großen Dualvariablen erzeugen große Verluste im Zielfunktionswert.

  $\Rightarrow$ Versuche, diese beim Runden zu erfüllen.
- Sukzessives Fixieren: Setze eine oder mehrere Variablen, deren Wert „fast“ ganzzahlig ist, auf den gerundeten Wert und löse das LP für die restlichen Variablen erneut (u.U. rückgängig machen, falls nun unzulässig).

Startlösung aus der LP-Relaxation

Typische Ansätze:

- Oft darf aus mehreren $\{0,1\}$-Variablen nur eine gewählt werden:
  $$\text{LP-Relaxation:} \qquad \sum_{i \in N} x_i = 1,\ x_i \in [0,1]$$
  Interpretiere x_i als Wahrscheinlichkeit, dass x_i auf 1 gesetzt werden soll und erzeuge damit mehrere Lösungen zufällig, nimm die beste.
- Abweichungen von Nebenbedingungen mit großen Dualvariablen erzeugen große Verluste im Zielfunktionswert.
  $\Rightarrow$ Versuche, diese beim Runden zu erfüllen.
- Sukzessives Fixieren: Setze eine oder mehrere Variablen, deren Wert „fast“ ganzzahlig ist, auf den gerundeten Wert und löse das LP für die restlichen Variablen erneut (u.U. rückgängig machen, falls nun unzulässig).

Für einige grundlegende Probleme gibt es Rundungsverfahren, die aus LP-Lösung gerundete Lösungen mit Gütegarantie erzeugen (Approximationsalgorithmen), diese sind gute Ideenlieferanten für eigene Rundungsverfahren.

Verbesserungsverfahren allgemein

Gemeinsame Grundelemente:

- **Suchumgebung/Nachbarschaft erklären:** beschreibt, welche Lösungen ausgehend von der derzeitigen untersucht werden können oder sollen (etwa alle durch gewisse Austauschschritte erreichbaren Lösungen, oder Freigeben gewisser Variablen für lokales Nachoptimieren, etc.)
  Mathematisch: Jeder Lösung $\hat{x}$ wird eine (Nachbarschafts-) Menge $\mathcal{N}(\hat{x})$ von benachbarten Lösungen zugeordnet.

Verbesserungsverfahren allgemein

Gemeinsame Grundelemente:

- **Suchumgebung/Nachbarschaft erklären:** beschreibt, welche Lösungen ausgehend von der derzeitigen untersucht werden können oder sollen (etwa alle durch gewisse Austauschschritte erreichbaren Lösungen, oder Freigeben gewisser Variablen für lokales Nachoptimieren, etc.)
  Mathematisch: Jeder Lösung $\hat{x}$ wird eine (Nachbarschafts-) Menge $\mathcal{N}(\hat{x})$ von benachbarten Lösungen zugeordnet.
- **Fortschrittsmaß definieren:** Bewertungsfunktion $f(x)$ für neu erzeugte Lösungen, die Zielfunktion und Bestrafung von Unzulässigkeiten kombiniert
  [s. Merit- und Filter-Ansatz der nichtlin. Opt.]

Verbesserungsverfahren allgemein

Gemeinsame Grundelemente:

- **Suchumgebung/Nachbarschaft erklären:** beschreibt, welche Lösungen ausgehend von der derzeitigen untersucht werden können oder sollen (etwa alle durch gewisse Austauschschritte erreichbaren Lösungen, oder Freigeben gewisser Variablen für lokales Nachoptimieren, etc.)
  Mathematisch: Jeder Lösung $\hat{x}$ wird eine (Nachbarschafts-) Menge $\mathcal{N}(\hat{x})$ von benachbarten Lösungen zugeordnet.
- **Fortschrittsmaß definieren:** Bewertungsfunktion $f(x)$ für neu erzeugte Lösungen, die Zielfunktion und Bestrafung von Unzulässigkeiten kombiniert
  [s. Merit- und Filter-Ansatz der nichtlin. Opt.]
- **Akzeptanz-Schema festlegen:** Gibt an, welche neu erzeugten Lösungen zur Fortsetzung der Suche verwendet werden sollen. Um lokale Optima verlassen zu können, werden ab und zu auch Verschlechterungen akzeptiert.

Lokal exakte Verfahren/lokales Enumerieren

Bestimme $\mathcal{N}(\cdot)$ so, dass $(P_{\hat{x}})$ $\min f(x)$ s.t. $x \in \mathcal{N}(\hat{x})$ für jedes $\hat{x}$ durch polynomiale Verfahren oder vollständige Enumeration exakt lösbar ist.

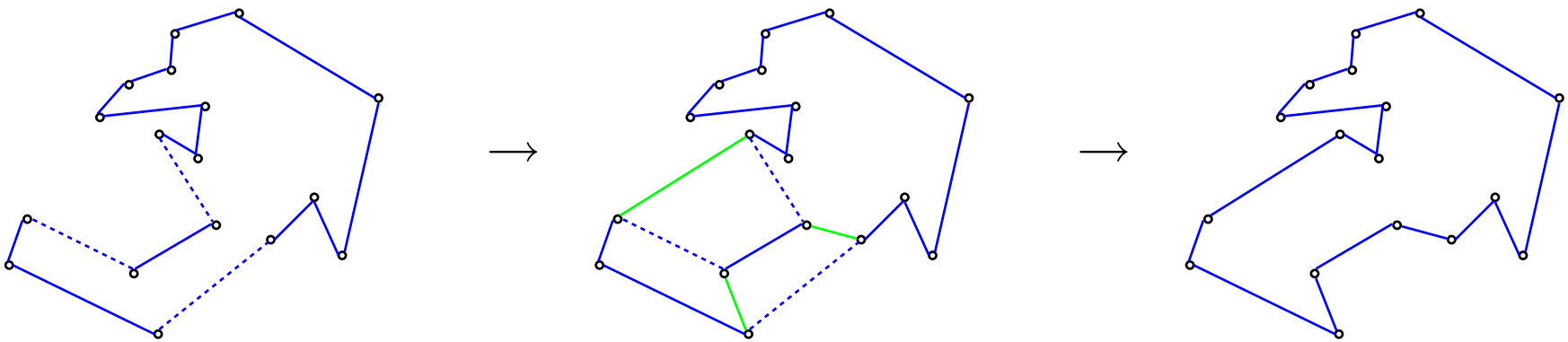
0. Bestimme eine Startlösung $\hat{x}$
1. Löse $(P_{\hat{x}}) \rightarrow \bar{x}$
2. Ist $f(\bar{x})$ besser als $f(\hat{x})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1., sonst STOP.

Lokal exakte Verfahren/lokales Enumerieren

Bestimme $\mathcal{N}(\cdot)$ so, dass $(P_{\hat{x}})$ $\min f(x)$ s.t. $x \in \mathcal{N}(\hat{x})$ für jedes $\hat{x}$ durch polynomiale Verfahren oder vollständige Enumeration exakt lösbar ist.

0. Bestimme eine Startlösung $\hat{x}$
1. Löse $(P_{\hat{x}}) \rightarrow \bar{x}$
2. Ist $f(\bar{x})$ besser als $f(\hat{x})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1., sonst STOP.

Bsp: 3-opt für TSP: 3 Kanten entfernen und neu zusammensetzen

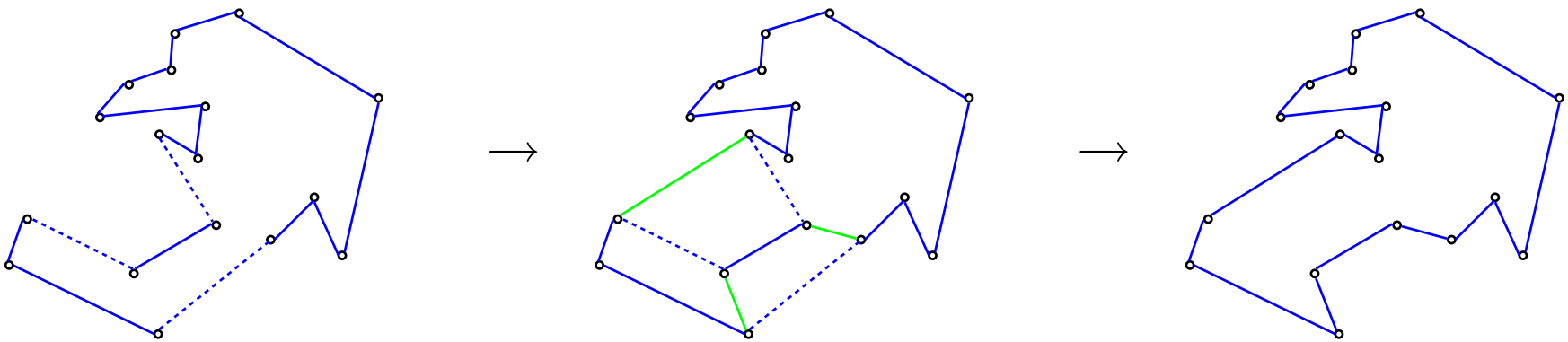


Lokal exakte Verfahren/lokales Enumerieren

Bestimme $\mathcal{N}(\cdot)$ so, dass $(P_{\hat{x}})$ $\min f(x)$ s.t. $x \in \mathcal{N}(\hat{x})$ für jedes $\hat{x}$ durch polynomiale Verfahren oder vollständige Enumeration exakt lösbar ist.

0. Bestimme eine Startlösung $\hat{x}$
1. Löse $(P_{\hat{x}}) \rightarrow \bar{x}$
2. Ist $f(\bar{x})$ besser als $f(\hat{x})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1., sonst STOP.

Bsp: 3-opt für TSP: 3 Kanten entfernen und neu zusammensetzen



---

die Kunst: Finde eine möglichst mächtige Nachbarschaft, für die $(P_{\hat{x}})$ noch polynomial lösbar ist.

Die Anzahl der Iterationen kann dennoch exponentiell sein!

Simulated Annealing (simuliertes langsames Abkühlen)

Erzeuge, in Schritt k, zufällig ein $\bar{x}$ aus $\mathcal{N}(\hat{x})$. Akzeptiere es, falls $f(\bar{x})$ besser als $f(\hat{x})$, sonst akzeptiere es nur mit Wahrscheinlichkeit

$$\exp\left(\frac{-|f(\hat{x}) - f(\bar{x})|}{T_k}\right) \qquad (0 < T_k \to 0 \text{ für } k \to \infty).$$

0. Bestimme Startlösung $\hat{x}$, Nullfolge $\{T_k > 0\}_{k\in\mathbb{N}}$, setze $k = 0$.
1. Wähle zufällig (gleichverteilt) $\bar{x} \in \mathcal{N}(\hat{x})$ setze $k \leftarrow k + 1$.
2. Ist $f(\bar{x})$ besser als $f(\hat{x})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1.
3. Ziehe gleichverteilt eine Zufallszahl $\zeta \in [0, 1]$.
   Ist $\zeta < \exp(\frac{-|f(\hat{x})-f(\bar{x})|}{c_k})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1.
4. Gehe zu 1. (ohne $\hat{x}$ zu ändern).

Wähle $\mathcal{N}(\cdot)$ so, dass jedes x über Zwischenstationen erreichbar ist.

Simulated Annealing (simuliertes langsames Abkühlen)

Erzeuge, in Schritt k, zufällig ein $\bar{x}$ aus $\mathcal{N}(\hat{x})$. Akzeptiere es, falls $f(\bar{x})$ besser als $f(\hat{x})$, sonst akzeptiere es nur mit Wahrscheinlichkeit

$$\exp\left(\frac{-|f(\hat{x}) - f(\bar{x})|}{T_k}\right) \qquad (0 < T_k \to 0 \text{ für } k \to \infty).$$

0. Bestimme Startlösung $\hat{x}$, Nullfolge $\{T_k > 0\}_{k\in\mathbb{N}}$, setze $k = 0$.
1. Wähle zufällig (gleichverteilt) $\bar{x} \in \mathcal{N}(\hat{x})$ setze $k \leftarrow k + 1$.
2. Ist $f(\bar{x})$ besser als $f(\hat{x})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1.
3. Ziehe gleichverteilt eine Zufallszahl $\zeta \in [0, 1]$.
   Ist $\zeta < \exp(\frac{-|f(\hat{x})-f(\bar{x})|}{c_k})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1.
4. Gehe zu 1. (ohne $\hat{x}$ zu ändern).

Wähle $\mathcal{N}(\cdot)$ so, dass jedes x über Zwischenstationen erreichbar ist.

Geht die (Temperatur-/Abkühlungs-)Folge T_k sehr langsam gegen Null, wird jedes x mit positiver Wahrscheinlichkeit irgendwann besucht (vollständige Enumeration), also auch das Optimum. (Aber wann?)

Tabu-Search

Idee: Versuche möglichst unterschiedliche Lösungen zu erzeugen.

Beschreibe $\mathcal{N}(\cdot)$ durch Änderungsregeln $r \in \mathcal{R}$ und speichere verwendete Regeln in einer Tabuliste $\mathcal{L}$. Für ein neues $\bar{x}$ sollte wenigstens eine Regel $r \in \mathcal{R} \setminus \mathcal{L}$ verwendet werden oder es muss besseren Wert haben.

0. Bestimme Startlösung $\hat{x}$, setze $\mathcal{L} = \emptyset$.
1. Erzeuge einige $x \in \mathcal{N}(\hat{x})$ durch mehrmaliges zufälliges Anwenden von Regeln aus $\mathcal{R}$, sammle diese in einer Menge S.
2. Wähle ein $\bar{x}$ aus S nach Tabuliste $\mathcal{L}$ und $f(\cdot)$.
3. Aktualisiere die Tabuliste $\mathcal{L}$, setze $\hat{x} \leftarrow \bar{x}$, gehe zu 1.

Tabu-Search

Idee: Versuche möglichst unterschiedliche Lösungen zu erzeugen.

Beschreibe $\mathcal{N}(\cdot)$ durch Änderungsregeln $r \in \mathcal{R}$ und speichere verwendete Regeln in einer Tabuliste $\mathcal{L}$. Für ein neues $\bar{x}$ sollte wenigstens eine Regel $r \in \mathcal{R} \setminus \mathcal{L}$ verwendet werden oder es muss besseren Wert haben.

0. Bestimme Startlösung $\hat{x}$, setze $\mathcal{L} = \emptyset$.
1. Erzeuge einige $x \in \mathcal{N}(\hat{x})$ durch mehrmaliges zufälliges Anwenden von Regeln aus $\mathcal{R}$, sammle diese in einer Menge S.
2. Wähle ein $\bar{x}$ aus S nach Tabuliste $\mathcal{L}$ und $f(\cdot)$.
3. Aktualisiere die Tabuliste $\mathcal{L}$, setze $\hat{x} \leftarrow \bar{x}$, gehe zu 1.

Bsp. TSP: $\mathcal{R} = \{r_{ij} := \text{vertausche Reihenfolge von Stadt } i \text{ und } j\}$.
$\mathcal{L} = \{r_{ij} : r_{ij} \text{ wurde in den letzten } n/10 \text{ Schritten verwendet}\}$

Tabu-Search

Idee: Versuche möglichst unterschiedliche Lösungen zu erzeugen.

Beschreibe $\mathcal{N}(\cdot)$ durch Änderungsregeln $r \in \mathcal{R}$ und speichere verwendete Regeln in einer Tabuliste $\mathcal{L}$. Für ein neues $\bar{x}$ sollte wenigstens eine Regel $r \in \mathcal{R} \setminus \mathcal{L}$ verwendet werden oder es muss besseren Wert haben.

0. Bestimme Startlösung $\hat{x}$, setze $\mathcal{L} = \emptyset$.
1. Erzeuge einige $x \in \mathcal{N}(\hat{x})$ durch mehrmaliges zufälliges Anwenden von Regeln aus $\mathcal{R}$, sammle diese in einer Menge S.
2. Wähle ein $\bar{x}$ aus S nach Tabuliste $\mathcal{L}$ und $f(\cdot)$.
3. Aktualisiere die Tabuliste $\mathcal{L}$, setze $\hat{x} \leftarrow \bar{x}$, gehe zu 1.

Bsp. TSP: $\mathcal{R} = \{r_{ij} := \text{vertausche Reihenfolge von Stadt } i \text{ und } j\}$.
$\mathcal{L} = \{r_{ij} : r_{ij} \text{ wurde in den letzten } n/10 \text{ Schritten verwendet}\}$

Über die Regeln $\mathcal{R}$ sollte jede Lösung erreichbar sein.
Allgemeine theoretische Resultate oder Qualitätsgarantien gibt es nicht.

Genetische Algorithmen

Idee: Lasse die Evolution für Dich arbeiten (und warte derweilen).

Erzeuge aus einer Population von Lösungen durch Selektion (Auswahl der nächsten Eltern), Rekombination/Vermehrung (Austausch von Teillösungen), Mutation (zufälliges Verändern) die nächste Population.

0. Wähle $k \in \mathbb{N}$ und bestimme eine Startpopulation $\mathcal{P}$, $|\mathcal{P}| \geq 2k$.
1. Bestimme die durchschnittliche Fitness $\bar{f} = \sum_{x \in \mathcal{P}} f(x)/|\mathcal{P}|$
2. Lösche x aus $\mathcal{P}$ mit Wahrscheinlichkeit prop. zu $\frac{f(x)}{\bar{f}}$, bis $|P| = 2k$.
3. Bilde zufällig k Paare aus $\mathcal{P}$, erzeuge für jedes Paar einige Nachkommen durch Rekombination und Mutation $\rightarrow \bar{\mathcal{P}}$
4. Setze $\mathcal{P} \leftarrow \bar{\mathcal{P}}$, gehe zu 1.

Genetische Algorithmen

Idee: Lasse die Evolution für Dich arbeiten (und warte derweilen).

Erzeuge aus einer Population von Lösungen durch Selektion (Auswahl der nächsten Eltern), Rekombination/Vermehrung (Austausch von Teillösungen), Mutation (zufälliges Verändern) die nächste Population.

0. Wähle $k \in \mathbb{N}$ und bestimme eine Startpopulation $\mathcal{P}$, $|\mathcal{P}| \geq 2k$.
1. Bestimme die durchschnittliche Fitness $\bar{f} = \sum_{x \in \mathcal{P}} f(x)/|\mathcal{P}|$
2. Lösche x aus $\mathcal{P}$ mit Wahrscheinlichkeit prop. zu $\frac{f(x)}{\bar{f}}$, bis $|P| = 2k$.
3. Bilde zufällig k Paare aus $\mathcal{P}$, erzeuge für jedes Paar einige Nachkommen durch Rekombination und Mutation $\rightarrow \bar{\mathcal{P}}$
4. Setze $\mathcal{P} \leftarrow \bar{\mathcal{P}}$, gehe zu 1.

- Viele Experimente mit Population der Größe 1 (!!!)
- Theorie besagt, dass Simulated Annealing eher Optima findet.

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)
- Fast alle hoffen, mit (leicht gesteuertem) Zufall eine Optimallösung zu finden (passt zu *NP*!).

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)
- Fast alle hoffen, mit (leicht gesteuertem) Zufall eine Optimallösung zu finden (passt zu *NP*!).

Vorteile:

- Auch bei wenig Problemverständnis ist rasch etwas implementiert, erste Lösungen sind schnell erzeugt, Regeln schnell geändert.

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)
- Fast alle hoffen, mit (leicht gesteuertem) Zufall eine Optimallösung zu finden (passt zu *NP*!).

Vorteile:

- Auch bei wenig Problemverständnis ist rasch etwas implementiert, erste Lösungen sind schnell erzeugt, Regeln schnell geändert.
- Man kann viele Parameter einstellen.

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)
- Fast alle hoffen, mit (leicht gesteuertem) Zufall eine Optimallösung zu finden (passt zu *NP*!).

Vorteile:

- Auch bei wenig Problemverständnis ist rasch etwas implementiert, erste Lösungen sind schnell erzeugt, Regeln schnell geändert.
- Man kann viele Parameter einstellen.

Nachteile:

- Man muss viele Parameter einstellen, ohne gute Richtschnur!

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)
- Fast alle hoffen, mit (leicht gesteuertem) Zufall eine Optimallösung zu finden (passt zu *NP*!).

Vorteile:

- Auch bei wenig Problemverständnis ist rasch etwas implementiert, erste Lösungen sind schnell erzeugt, Regeln schnell geändert.
- Man kann viele Parameter einstellen.

Nachteile:

- Man muss viele Parameter einstellen, ohne gute Richtschnur!
- Konvergenz der Verfahren sagt nichts über die Qualität der Lösung. Ohne dazupassende Relaxation hat man keine Ahnung, wie weit man vom Optimum entfernt ist (manchmal sehr weit).

Inhaltsübersicht für heute:

Anwendungsbeispiel: Rundreiseprobleme (TSP)

Finden „guter“ Lösungen, Heuristiken

Gemischt-ganzzahlige Optimierung

Gemischt-ganzzahlige Optimierung (MIP)

Für eine Teilmenge der Indices $G \subseteq \{1, \ldots, n\}$ soll x ganzzahlig sein.

$$\max\ c^T x \text{ s.t. } Ax \geq b,\ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Enthält ganzz. Optimierung als Spezialfall, ist aber deutlich breiter.

Gemischt-ganzzahlige Optimierung (MIP)

Für eine Teilmenge der Indices $G \subseteq \{1, \ldots, n\}$ soll x ganzzahlig sein.

$$\max\ c^T x \text{ s.t. } Ax \geq b,\ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Enthält ganzz. Optimierung als Spezialfall, ist aber deutlich breiter.

Anwendungsbeispiel: Standortoptimierung mit Fixkosten
Gegeben eine Menge K an Kunden mit Bedarfen b_k und eine Menge M möglicher Standorte für Versandlager, jeweils mit Mietkosten c_m, Kapazität b_m und Transportkosten c_{km} pro Einheit für $k \in K, m \in M$. Welche Standorte sollen geöffnet werden?

Variablen:

Gemischt-ganzzahlige Optimierung (MIP)

Für eine Teilmenge der Indices $G \subseteq \{1, \ldots, n\}$ soll x ganzzahlig sein.

$$\max\ c^T x \text{ s.t. } Ax \geq b,\ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Enthält ganzz. Optimierung als Spezialfall, ist aber deutlich breiter.

Anwendungsbeispiel: Standortoptimierung mit Fixkosten
Gegeben eine Menge K an Kunden mit Bedarfen b_k und eine Menge M möglicher Standorte für Versandlager, jeweils mit Mietkosten c_m, Kapazität b_m und Transportkosten c_{km} pro Einheit für $k \in K, m \in M$.
Welche Standorte sollen geöffnet werden?

Variablen:
$x_m \in \{0, 1\}$, $m \in M$ ... Versandlager wird in m errichtet.
$x_{km} \in \mathbb{R}_+$, $k \in K, m \in M$ ... Menge, die Kunde k von Standort m erhält.

Nebenbedingungen:

Gemischt-ganzzahlige Optimierung (MIP)

Für eine Teilmenge der Indices $G \subseteq \{1, \ldots, n\}$ soll x ganzzahlig sein.

$$\max\ c^T x \text{ s.t. } Ax \geq b,\ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Enthält ganzz. Optimierung als Spezialfall, ist aber deutlich breiter.

Anwendungsbeispiel: Standortoptimierung mit Fixkosten
Gegeben eine Menge K an Kunden mit Bedarfen b_k und eine Menge M möglicher Standorte für Versandlager, jeweils mit Mietkosten c_m, Kapazität b_m und Transportkosten c_{km} pro Einheit für $k \in K, m \in M$.
Welche Standorte sollen geöffnet werden?

Variablen:
$x_m \in \{0, 1\}$, $m \in M$ ... Versandlager wird in m errichtet.
$x_{km} \in \mathbb{R}_+$, $k \in K, m \in M$ ... Menge, die Kunde k von Standort m erhält.

Nebenbedingungen:
$\sum_{m \in M} x_{km} = b_k$, $k \in K$ ... Kunde k erhält seine Bestellung
$\sum_{k \in K} x_{km} \leq b_m x_m$, $m \in M$ ... Standort m verteilt maximal b_m.

Zielfunktion:

Gemischt-ganzzahlige Optimierung (MIP)

Für eine Teilmenge der Indices $G \subseteq \{1, \ldots, n\}$ soll x ganzzahlig sein.

$$\max\ c^T x \text{ s.t. } Ax \geq b,\ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Enthält ganzz. Optimierung als Spezialfall, ist aber deutlich breiter.

Anwendungsbeispiel: Standortoptimierung mit Fixkosten

Gegeben eine Menge K an Kunden mit Bedarfen b_k und eine Menge M möglicher Standorte für Versandlager, jeweils mit Mietkosten c_m, Kapazität b_m und Transportkosten c_{km} pro Einheit für $k \in K, m \in M$.
Welche Standorte sollen geöffnet werden?

Variablen:
$x_m \in \{0, 1\}$, $m \in M$ ... Versandlager wird in m errichtet.
$x_{km} \in \mathbb{R}_+$, $k \in K, m \in M$ ... Menge, die Kunde k von Standort m erhält.

Nebenbedingungen:
$\sum_{m \in M} x_{km} = b_k$, $k \in K$ ... Kunde k erhält seine Bestellung
$\sum_{k \in K} x_{km} \leq b_m x_m$, $m \in M$ ... Standort m verteilt maximal b_m.

Zielfunktion:
$\min \sum_{k \in K, m \in M} c_{km} x_{km} + \sum_{m \in M} c_m x_m = c^T x$

MIP-Modellierungstechniken:

Bedingte Ungleichungen: Ungleichungen, die nur bei bestimmten Entscheidungen beachtet werden müssen, können bei entgegengesetzter Entscheidung durch einen big M-Term erfüllt werden.

MIP-Modellierungstechniken:

Bedingte Ungleichungen: Ungleichungen, die nur bei bestimmten Entscheidungen beachtet werden müssen, können bei entgegengesetzter Entscheidung durch einen big M-Term erfüllt werden.

Bsp: Fährt Zug A mit Abfahrtszeit t_A vor Zug B mit Abfahrtszeit t_B, ist eine Mindestzugfolgezeit $t_{AB} > 0$ einzuhalten, also $t_B \geq t_A + t_{AB}$. Für Zug B vor Zug A muss wiederum $t_A \geq t_B + t_{BA}$ erfüllt sein. Die nachfolgenden Züge dürfen natürlich auch viel später fahren.

Variablen

MIP-Modellierungstechniken:

Bedingte Ungleichungen: Ungleichungen, die nur bei bestimmten Entscheidungen beachtet werden müssen, können bei entgegengesetzter Entscheidung durch einen big M-Term erfüllt werden.

Bsp: Fährt Zug A mit Abfahrtszeit t_A vor Zug B mit Abfahrtszeit t_B, ist eine Mindestzugfolgezeit $t_{AB} > 0$ einzuhalten, also $t_B \geq t_A + t_{AB}$. Für Zug B vor Zug A muss wiederum $t_A \geq t_B + t_{BA}$ erfüllt sein. Die nachfolgenden Züge dürfen natürlich auch viel später fahren.

Variablen ($M \gg 0$, größer als späteste Abfahrtszeit von t_A und t_B):
$x_{AB} \in \{0, 1\}$ . . . 1 falls A vor B fährt, sonst 0
$t_A, t_B \in [0, M]$. . . Abfahrtszeit

Nebenbedingungen:
$t_A + t_{AB} \leq t_B + M(1 - x_{AB})$ . . . nur wichtig, wenn $x_{AB} = 1$
$t_B + t_{BA} \leq t_A + Mx_{AB}$ . . . nur wichtig, wenn $x_{AB} = 0$.

MIP-Modellierungstechniken:

Bedingte Ungleichungen: Ungleichungen, die nur bei bestimmten Entscheidungen beachtet werden müssen, können bei entgegengesetzter Entscheidung durch einen big M-Term erfüllt werden.

Bsp: Fährt Zug A mit Abfahrtszeit t_A vor Zug B mit Abfahrtszeit t_B, ist eine Mindestzugfolgezeit $t_{AB} > 0$ einzuhalten, also $t_B \geq t_A + t_{AB}$. Für Zug B vor Zug A muss wiederum $t_A \geq t_B + t_{BA}$ erfüllt sein. Die nachfolgenden Züge dürfen natürlich auch viel später fahren.

Variablen ($M \gg 0$, größer als späteste Abfahrtszeit von t_A und t_B):
$x_{AB} \in \{0, 1\}$ . . . 1 falls A vor B fährt, sonst 0
$t_A, t_B \in [0, M]$. . . Abfahrtszeit

Nebenbedingungen:
$t_A + t_{AB} \leq t_B + M(1 - x_{AB})$ . . . nur wichtig, wenn $x_{AB} = 1$
$t_B + t_{BA} \leq t_A + Mx_{AB}$ . . . nur wichtig, wenn $x_{AB} = 0$.

- M zu groß $\rightarrow$ Unglg. in LP-Relaxation wirkungslos $\rightarrow$ schlechte Schranke
- gut, falls Verletzungsspielraum der Unglg. gut abschätzbar (siehe das Standortplanungsbeispiel)
- in Branch&Bound hilfreich, wenn auf der Variable gebrancht wird

Modellierung logischer Bedingungen

Für ein $x_i \in \{0, 1\}$ steht $x_i = 1$ oft für „Aussage i ist wahr“.
Logische Verknüpfungen sind dann so erzeugbar:

Log. Bedingung	Formulierung
$x_2 = (\text{nicht } x_1)$	

Modellierung logischer Bedingungen

Für ein $x_i \in \{0, 1\}$ steht $x_i = 1$ oft für „Aussage i ist wahr“.
Logische Verknüpfungen sind dann so erzeugbar:

Log. Bedingung	Formulierung
$x_2 = (\text{nicht } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ oder } x_2)$	

Modellierung logischer Bedingungen

Für ein $x_i \in \{0, 1\}$ steht $x_i = 1$ oft für „Aussage i ist wahr“.
Logische Verknüpfungen sind dann so erzeugbar:

Log. Bedingung	Formulierung
$x_2 = (\text{nicht } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ oder } x_2)$	$x_3 \geq x_1,\ x_3 \geq x_2,\ x_3 \leq x_1 + x_2$
$x_3 = (x_1 \text{ und } x_2)$	$x_3 \leq x_1,\ x_3 \leq x_2,\ x_3 \geq x_1 + x_2 - 1$
$x_1 \Rightarrow x_2$	

Modellierung logischer Bedingungen

Für ein $x_i \in \{0, 1\}$ steht $x_i = 1$ oft für „Aussage i ist wahr“.
Logische Verknüpfungen sind dann so erzeugbar:

Log. Bedingung	Formulierung
$x_2 = (\text{nicht } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ oder } x_2)$	$x_3 \geq x_1,\ x_3 \geq x_2,\ x_3 \leq x_1 + x_2$
$x_3 = (x_1 \text{ und } x_2)$	$x_3 \leq x_1,\ x_3 \leq x_2,\ x_3 \geq x_1 + x_2 - 1$
$x_1 \Rightarrow x_2$	$x_1 \leq x_2$
$x_1 \Leftrightarrow x_2$	

Modellierung logischer Bedingungen

Für ein $x_i \in \{0,1\}$ steht $x_i = 1$ oft für „Aussage i ist wahr“.
Logische Verknüpfungen sind dann so erzeugbar:

Log. Bedingung	Formulierung
$x_2 = (\text{nicht } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ oder } x_2)$	$x_3 \geq x_1,\ x_3 \geq x_2,\ x_3 \leq x_1 + x_2$
$x_3 = (x_1 \text{ und } x_2)$	$x_3 \leq x_1,\ x_3 \leq x_2,\ x_3 \geq x_1 + x_2 - 1$
$x_1 \Rightarrow x_2$	$x_1 \leq x_2$
$x_1 \Leftrightarrow x_2$	$x_1 = x_2$

Bemerkung: Zusammen mit $0 \leq x_i \leq 1$ beschreiben die Nebenbedinungen

$$\operatorname{conv}\left\{ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \{0,1\}^3 : \text{die } x_i \text{ erfüllen die logische Bedingung} \right\}.$$

Mit dieser Technik sind Formulierungen weiterer Beziehungen ableitbar.
Übung: $x_3 = (x_1 \text{ xor } x_2)$

Schnittebenen für MIP

Wie in der ganzz. Optimierung ist „conv“ die beste lineare Relaxation,

$$P_G := \mathrm{conv}\{x \in \mathbb{R}^n : Ax \leq b, x_G \in \mathbb{Z}^G\}.$$

$Ax \leq b$ ist die LP-Relaxation, P_G wird durch Schnittebenen angenähert.

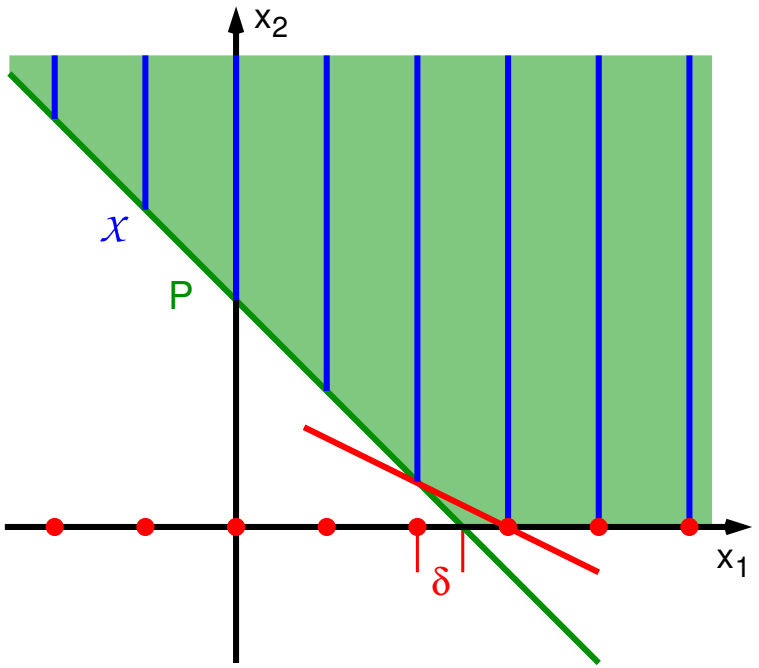
Schnittebenen für MIP

Wie in der ganzz. Optimierung ist „conv“ die beste lineare Relaxation,

$$P_G := \text{conv}\{x \in \mathbb{R}^n : Ax \leq b, x_G \in \mathbb{Z}^G\}.$$

$Ax \leq b$ ist die LP-Relaxation, P_G wird durch Schnittebenen angenähert.

Bsp: Mixed Integer Rounding Ungleichung (MIR)
Einfachste Form: $\mathcal{X} = \{(x_1, x_2) \in \mathbb{Z} \times \mathbb{R}_+ : x_1 + x_2 \geq \beta\}$, $\beta \in \mathbb{R}$



Setze $\delta := \beta - \lfloor \beta \rfloor$,
dann ist die Unglg.

$$x_1 + \frac{1}{\delta} x_2 \geq \lceil \beta \rceil$$

gültig für $\mathcal{X}$.

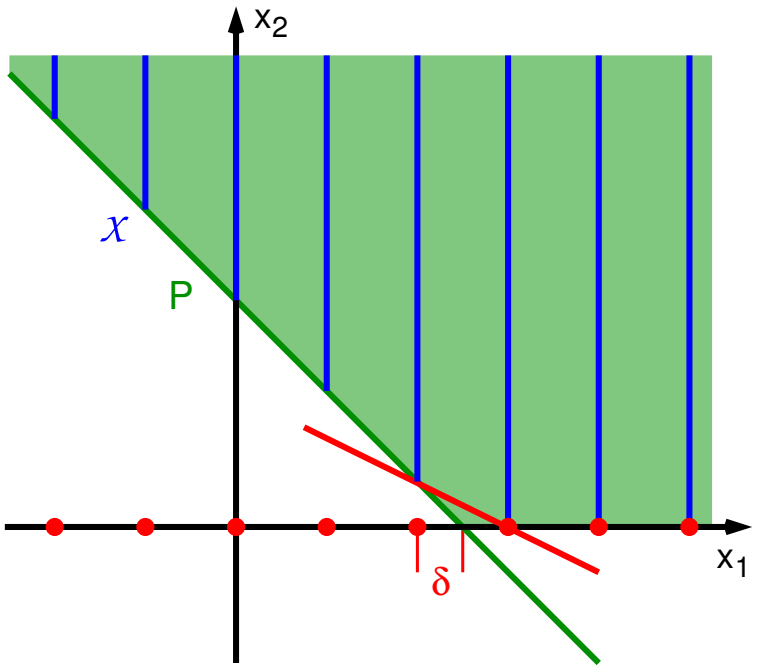
Schnittebenen für MIP

Wie in der ganzz. Optimierung ist „conv“ die beste lineare Relaxation,

$$P_G := \text{conv}\{x \in \mathbb{R}^n : Ax \leq b, x_G \in \mathbb{Z}^G\}.$$

$Ax \leq b$ ist die LP-Relaxation, P_G wird durch Schnittebenen angenähert.

Bsp: Mixed Integer Rounding Ungleichung (MIR)
Einfachste Form: $\mathcal{X} = \{(x_1, x_2) \in \mathbb{Z} \times \mathbb{R}_+ : x_1 + x_2 \geq \beta\}$, $\beta \in \mathbb{R}$



Setze $\delta := \beta - \lfloor \beta \rfloor$,
dann ist die Unglg.

$$x_1 + \frac{1}{\delta} x_2 \geq \lceil \beta \rceil$$

gültig für $\mathcal{X}$.

In state-of-the-art Paketen sind viele weitere enthalten
(flow cover, cliques, etc.)

Branch-and-Cut Frameworks

Wird in einem Branch&Bound-Verfahren für jedes Teilproblem die LP-Relaxation durch Schnittebenen verschärft, spricht man von Branch&Cut-Verfahren. Dies sind derzeit die besten Verfahren für allgemeine gemischt-ganzzahlige Optimierungsprobleme.

Branch-and-Cut Frameworks

Wird in einem Branch&Bound-Verfahren für jedes Teilproblem die LP-Relaxation durch Schnittebenen verschärft, spricht man von Branch&Cut-Verfahren. Dies sind derzeit die besten Verfahren für allgemeine gemischt-ganzzahlige Optimierungsprobleme.

Eine effiziente Branch&Cut Implementierung ist sehr schwer:

- Auswahl des nächsten Teilproblems
- Auswahl der Verzweigungsvariable, Fixieren von Variablen
- Teilprobleme effizient inkrementell speichern
- Schnittebenen über mehrere Teilprobleme verwalten
- effiziente Heuristiken für zulässige Lösungen

etc.

Branch-and-Cut Frameworks

Wird in einem Branch&Bound-Verfahren für jedes Teilproblem die LP-Relaxation durch Schnittebenen verschärft, spricht man von Branch&Cut-Verfahren. Dies sind derzeit die besten Verfahren für allgemeine gemischt-ganzzahlige Optimierungsprobleme.

Eine effiziente Branch&Cut Implementierung ist sehr schwer:

- Auswahl des nächsten Teilproblems
- Auswahl der Verzweigungsvariable, Fixieren von Variablen
- Teilprobleme effizient inkrementell speichern
- Schnittebenen über mehrere Teilprobleme verwalten
- effiziente Heuristiken für zulässige Lösungen

etc.

Es gibt Software-Pakete, die alle Verwaltungsaspekte übernehmen und einem den Einbau weiterer Schnittebenen und Heuristiken ermöglichen.

z.B.: SCIP, Cplex, Gurobi, Abacus ...