11 Ganzzahlige und kombinatorische Optimierung

Bisher behandelt:

- Optimierung linearer Funktionen unter linearen Nebenbedingungen
- Auswahl des Optimums aus einem Kontinuum möglicher Lösungen

Viele Probleme erfordern die Auswahl optimaler Lösungen aus einer endlichen/abzählbaren Menge von Lösungen, z.B.

- Optimale Ressourcenzahl
- Optimale Reihenfolge
- Auswahl aus einer diskreten Menge von Alternativen
- Optimale Zuordnungen
- …

Formale Darstellung:

- Menge von Entscheidungsvariablen $\mathbf{x} \in \mathbb{G}^n$ (oft auch $\mathbf{x} \in \mathbb{N}_0^n$)
- Zielfunktion $Z=f(\mathbf{x})$ ist zu minimieren
- unter den Nebenbedingungen $\mathbf{h}(\mathbf{x}) \leq \mathbf{0}$
  (Nebenbedingungen definieren den zulässigen Bereich $W \subseteq \mathbb{G}^n$)

Einige Spezialfälle:

- $f(\mathbf{x})$ und $\mathbf{h}(\mathbf{x})$ lineare Funktionen $\Rightarrow$
  ganzzahlige (lineare) Programmierung
- $\mathbf{x} \in /^n \Rightarrow$ binäres Optimierungsproblem
- Auswahl aus diskreter Menge von Alternativen
  (z.B. Reihenfolge, Zuordnungen, Gruppierung, ...)
  $\Rightarrow$ kombinatorische Optimierung
- $\mathbf{x}=(\mathbf{x}_D,\mathbf{x}_K)$ mit $\mathbf{x}_D \in \mathbb{G}^{n1}$ und $\mathbf{x}_K \in ?^{n2} \Rightarrow$ gemischte Probleme

Ganzzahlige Parameter oft realistischer als kontinuierliche, da

- Variablen in der Realität ganzzahlig,
- auch kontinuierlichen Parameter nur mit beschränkter Genauigkeit einstellbar sind und
- Rechnerarithmetik mit diskreten Werten arbeitet

Aber ganzzahlige Optimierung ist schwierig

z.B. lineare Optimierung mit polynomiellem Aufwand lösbar, ganzzahlige lineare Optimierung NP-schwer

Hier nur eine kurze Übersicht über ganzzahlige Probleme
(weitere Details in anderen Vorlesungen, sehr aktives Gebiet an der TU Do.):

11.1 Ganzzahlige Programmierung

11.2 Lösungsansätze der kombinatorischen Optimierung

11.3 Lösungsmethoden für typische Beispielprobleme

11.4 Gemischt ganzzahlige Modellierung und Optimierung

11.1 Ganzzahlige Programmierung

Einfachste Form: Lineare ganzzahlige Programmierung

$Z=f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ und $\mathbf{Ax} \leq \mathbf{b}$ und $\mathbf{x} \in \mathbb{G}_+^n$

mit $\mathbf{A} \in \mathbb{G}^{m,n}$, $\mathbf{b} \in \mathbb{G}^m$ und $\mathbf{c} \in \mathbb{G}^n$

} **ILP**

n Entscheidungsvariablen und m Nebenbedingungen

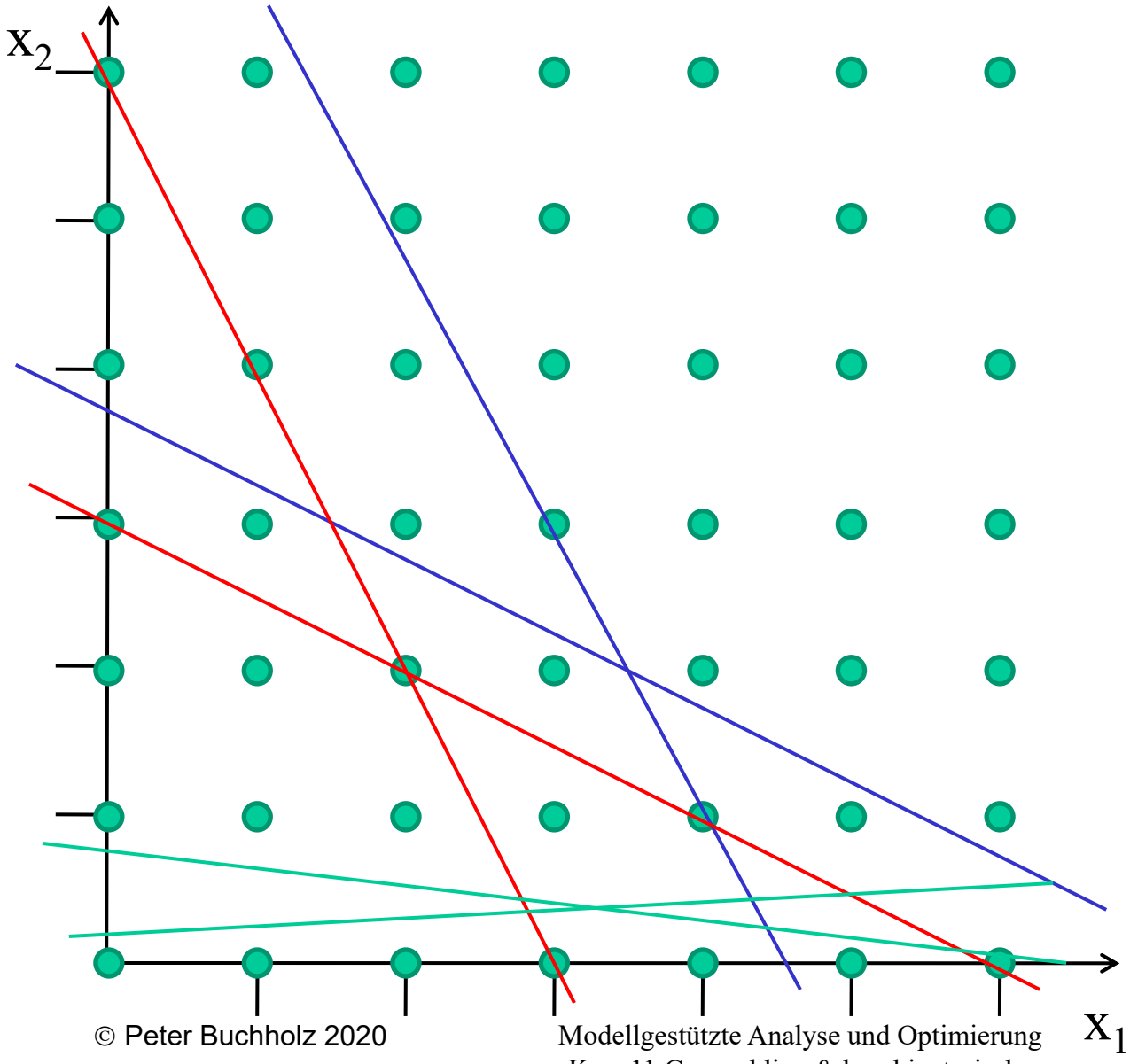
Sei $\mathbf{x}^* \in ?^n$ Lösung des LPs (ohne Ganzzahligkeit der Lösung)

Sei $\mathbf{x}' \in \mathbb{G}^n$ die gesuchte Lösung des ganzzahligen Problems

Was haben $\mathbf{x}^*$ und $\mathbf{x}'$ miteinander zu tun?

- Falls LP keine Lösung hat, so hat auch ILP keine Lösung
- Falls x^* ganzzahlig, so ist $x' = x^*$
- Ansonsten gibt es keine Beziehung
  (d.h. die Lösungsmenge des ILPs kann leer sein oder das Optimum „weit weg“ von x^* liegen $\Rightarrow$ Runden ist problematisch)

Beispiel

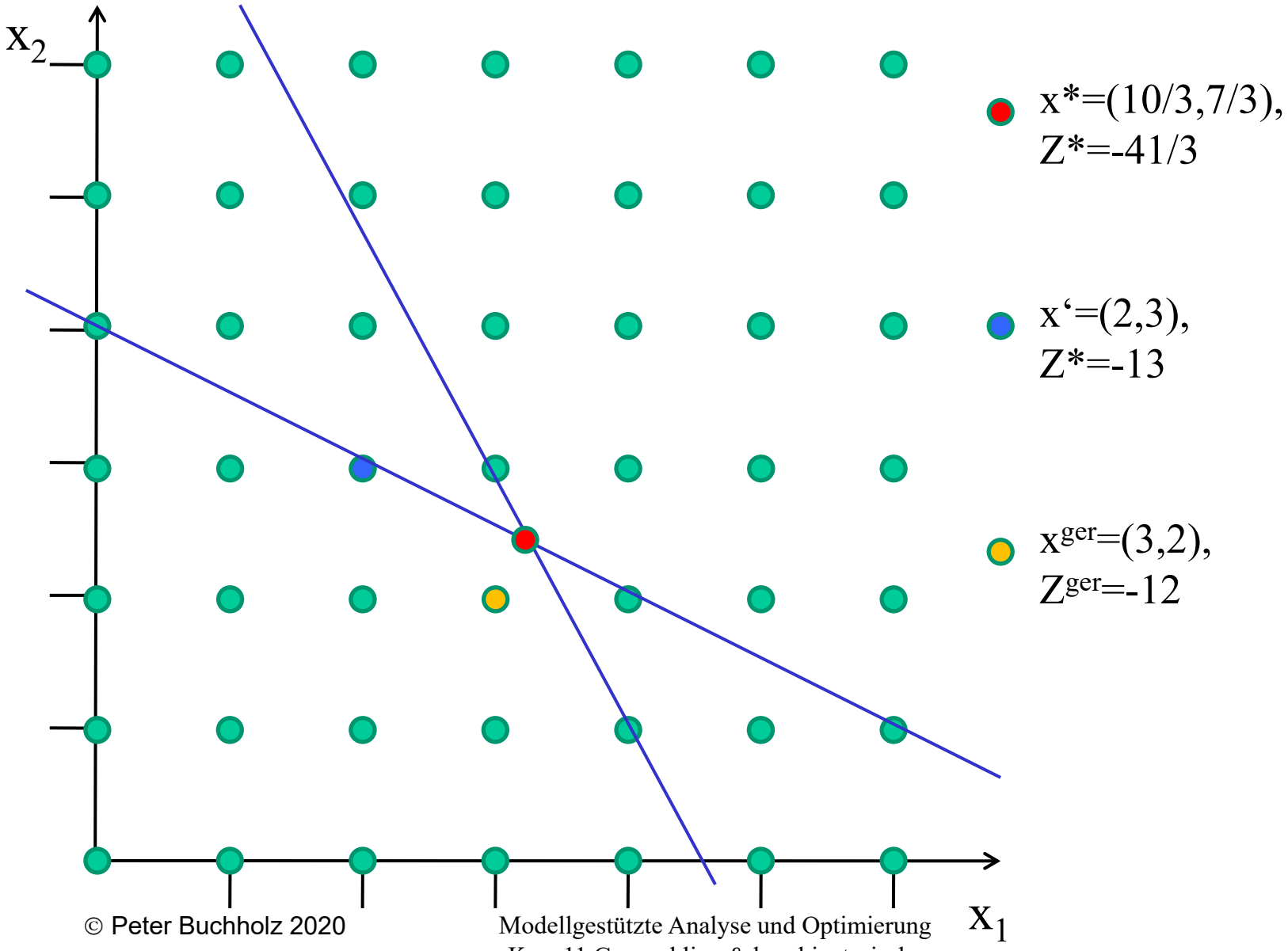


W≠∅
x* = **x‘** ??

W≠∅
x* = **x‘**

W≠∅ für LP
W=∅ für ILP
x* ≠ **x‘**

Beispiel: min $Z = -2x_1-3x_2$ und $x_1+2x_2 \leq 8$, $2x_1+x_2 \leq 9$, $x_1,x_2 \in G_+$



LP ist relaxierte Form des ILP (relaxiert: gelockert/gestrichen)

Erweiterte Form des ganzzahligen Problems:

$Z=f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ und $\mathbf{Ax}=\mathbf{b}$ mit $rg(\mathbf{A})=m$ und $\mathbf{x}\in\mathbb{N}_0^{n+m}$

Erweiterung um m Schlupfvariablen (wie beim LP)

Wann ist $\mathbf{x}^*$ ganzzahlig?

- $\mathbf{B}\in\mathbb{G}^{n,n}$ heißt ***unimodular***, falls $|\det \mathbf{B}| = 1$
- $\mathbf{A}\in\mathbb{G}^{m,n}$ heißt ***total unimodular***, falls die Determinante jeder quadratischen Untermatrix 0, -1 oder 1 ist
  (d.h. jede quadratische Untermatrix ist singulär oder unimodular)

Satz 11.1

Falls **A** total unimodular und Vektor **b** ganzzahlig, so hat das ILP min $\mathbf{c}^T\mathbf{x}$ und $\mathbf{Ax}=\mathbf{b}$ nur ganzzahlige Basislösungen.

Beweis:

Sei $\mathbf{B}=(\mathbf{A}_{k\bullet})$ eine Basismatrix und $\mathbf{x}_B$ eine Basislösung (Ordnung m)

d.h. $\mathbf{B}\mathbf{x}_B=\mathbf{b} \Rightarrow \mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$

Nach der Cramer'schen Regel gilt $\mathbf{B}^{-1} = \mathbf{B}^{adj}/\det \mathbf{B}$

Wobei $\mathbf{B}^{adj}$ die adjungierte der Matrix **B** ist, d.h.

$\mathbf{B}^{adj}(i,j) = (-1)^{i+j} M_{ij}$

mit M_{ij} := Determinante der Matrix, die aus **B** durch Streichen der i-ten Zeile und j-ten Spalte entsteht

B total unimodular $\Rightarrow$ $\det(\mathbf{B}) = -1$ oder $+1$ und $\mathbf{B}^{adj}$ ganzzahlig

$\Rightarrow$ bei ganzzahligem **b**

ist auch $\mathbf{x}_B$ ganzzahlig!

Prüfung, ob **A** total unimodular ist zwar polynomiell, aber in der Praxis nicht empfehlenswert!

Schnittebenenverfahren (nach Gomory):

- Weitere Nutzung des Simplex-Algorithmus
- Aber schrittweise Beseitigung unzulässiger (d.h. nicht ganzzahliger) Lösungen durch Einführung neuer Nebenbedingungen, so dass
  - gefundene optimale (nicht ganzzahlige) Lösungen nicht mehr zulässig sind und
  - alle ganzzahligen Lösungen zulässig bleiben

Verfahren existiert in unterschiedlichen Ausprägungen,

hier erst einmal nur in Grundform vorgestellt

Wie kann man neue Nebenbedingungen einführen, die die Bedingungen erfüllen?

Beispiel (von Folie 6):

Ausgangstableau

x_1	x_2	-b	
1	2	-8	$-x_3$
2	1	-9	$-x_4$
-2	-3	0	Z

Endtableau nach zwei Austauschschritten

x_4	x_3	-b	
-1/3	2/3	-7/3	$-x_2$
2/3	-1/3	-10/3	$-x_1$
1/3	4/3	-41/3	Z

Lösung nicht ganzzahlig deshalb nicht zulässig für ILP.

Sei

- B := { i | x_i Basisvariable }
  die Indexmenge der (momentanen) Basisvariable
- N := { i | x_i Nichtbasisvariable }
  die Indexmenge der (momentanen) Nichtbasisvariable

Jede Zeile des Endtableaus liefert eine Gleichung

$$-b_k + \sum_{l \in N} a_{kl} \cdot x_l = -x_k \ (k \in B)$$ mit $x_k = b_k$, da $x_l = 0$ für $l \in N$.

Sei $b_k = \lfloor b_k \rfloor + r_k$ und $a_{kl} = \lfloor a_{kl} \rfloor + r_{kl}$ mit $0 \leq r_k, r_{kl} < 1$

Umschreiben der Gleichung im Tableau liefert:

$$x_k + \sum_{l \in N} \lfloor a_{kl} \rfloor \cdot x_l - \lfloor b_k \rfloor = -\sum_{l \in N} r_{kl} \cdot x_l + r_k$$

Ganzzahlige Lösung sorgt für ganzzahlige rechte Seite

Ferner gilt $-\sum_{l \in N} r_{kl} \cdot x_l + r_k \leq r_k$ für $x_l \geq 0$

Für ganzzahlige Lösungen gilt damit $-\sum_{l \in N} r_{kl} \cdot x_l + r_k \leq 0$

Einführung einer neuen Nebenbedingung: $x_{n+1} = -r_k + \sum_{l \in N} r_{kl} \cdot x_l$

Es gilt:

- $x_{n+1} \geq 0$ für alle ganzzahligen Lösungen
- $x_{n+1} < 0$ für **x***, da $x^*_l = 0$ für $l \in N$ und x^*_k nicht ganzzahlig

Im Beispiel:

Forme neue Restriktion aus Gleichung für x_1:

$x_5 = -1/3 + 2/3 \cdot x_3 + 2/3 \cdot x_4$

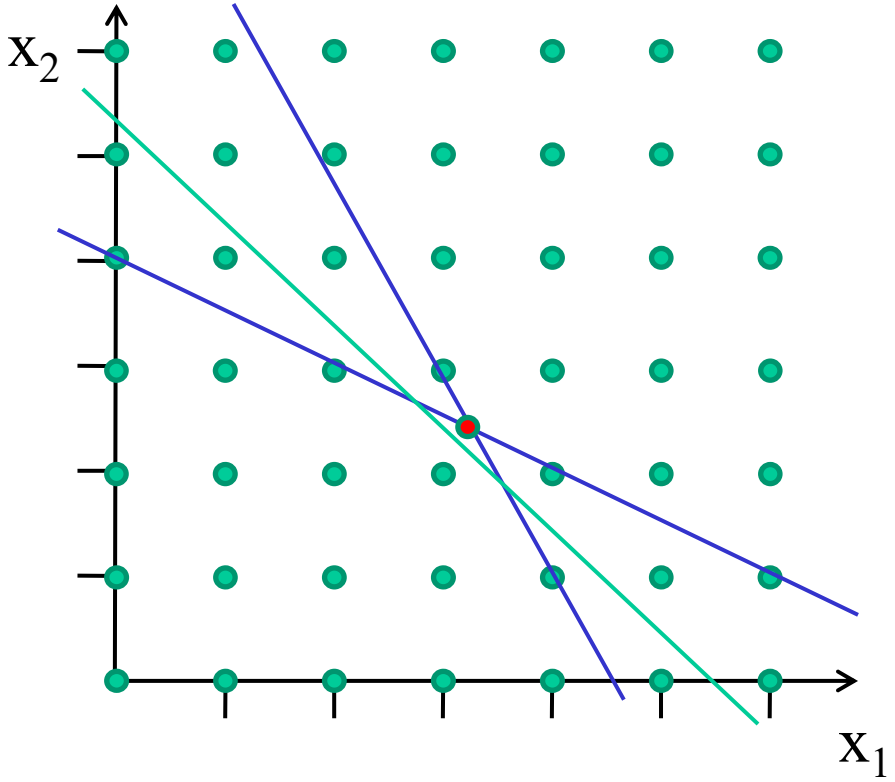
und $x_5 \geq 0$

Dies entspricht der Schnittebene

$2 \cdot x_1 + 2 \cdot x_2 = 11$

bzw. der Nebenbedingung

$2 \cdot x_1 + 2 \cdot x_2 \leq 11$



x_4	x_3	-b	
-1/3	2/3	-7/3	$-x_2$
2/3	-1/3	-10/3	$-x_1$
-2/3	-2/3	1/3	$-x_5$
1/3	4/3	-41/3	Z

Neue Basislösung nicht zulässig,
da $x_5 < 0$

Gilt allgemein, deshalb üblicherweise duale Lösung des linearen Problems (hier nicht behandelt)

Wir nutzen hier (den umständlicheren) Weg des allgemeinen Simplexverfahrens

Austauschschritt zur Erlangung einer Startlösung liefert

x_1	x_3	-b	
1/2	1/2	-4	$-x_2$
3/2	-1/2	-5	$-x_4$
1	-1	-3	$-x_5$
-1/2	3/2	-12	Z

(Pivotzeile 2, Pivotspalte 1)

Lösung ganzzahlig aber nicht optimal,
da $c_1 < 0$

Simplexschritt mit Pivotzeile 3, Pivotspalte 1 liefert:

x_5	x_3	-b	
-1/2	1	-5/2	$-x_2$
-3/2	1	-1/2	$-x_4$
1	-1	-3	$-x_1$
1/2	1	-13.5	Z

Lösung optimal, aber nicht ganzzahlig

- Einführung einer neuen Schnittebene
  $x_6 = -1/2 + 1/2 \cdot x_5$
- Dies entspricht der Nebenbedingung
  $x_1 + x_2 \leq 5$

Neues, erweitertes Tableau:

x_5	x_3	-b	
-1/2	1	-5/2	$-x_2$
-3/2	1	-1/2	$-x_4$
1	-1	-3	$-x_1$
-1/2	0	1/2	$-x_6$
1/2	1	-13.5	Z

Lösung nicht zulässig, da x_6 negativ

Simplexschritt mit Pivotzeile 3, Pivotspalte 1 liefert:

x_1	x_3	-b	
1/2	1/2	-4	$-x_2$
3/2	-1/2	-5	$-x_4$
1	-1	-3	$-x_5$
1/2	-1/2	-1	$-x_6$
-1/2	3/2	-12	Z

Lösung ganzzahlig aber nicht optimal

$\Rightarrow$ Simplexschritt mit Pivotzeile 4,
Pivotspalte 1

x_6	x_3	-b	
-1	1	-3	$-x_2$
-3	1	-2	$-x_4$
-2	0	-2	$-x_5$
2	-1	-2	$-x_1$
1	1	-13	Z

Lösung optimal und ganzzahlig

$\Rightarrow$ Lösung des ILP

Weitere Ergebnisse:

- Beweise für Endlichkeit des Verfahrens existieren
- Worst case Komplexität exponentiell (wie Simplex)
- In reiner Form kaum angewendet für praktische Probleme, da
  - viele zusätzlich Nebenbedingungen auftreten und komplexe Simplexlösungen erfordern und
  - die Lösungen oft numerisch problematisch sind

Es existieren 3 Schnittebenenverfahren von Gomory:

1. Vorgestelltes Verfahren (mit Nachteilen durch Rundungsfehler)
2. Ganzzahlige Rechnungen
3. Verfahren für gemischt ganzzahlige Probleme

Darstellung verschiedener Optimierungsprobleme als ILP:

Rucksackproblem:

- n Gegenstände mit Gewichten $a_j>0$ und Werten $c_j>0$ (j=1,…n)
- Maximalgewicht A>0

Ziel: Packe den Rucksack so, dass sein Gewicht ≤A und sein Wert möglichst groß ist!

ILP-Formulierung: $\min\left(-\sum_{i=1}^{n} x_i \cdot c_i\right)$

- n Entscheidungsvariablen $x_i \in \{0,1\}$ (i=1,…,n)
- Eine Nebenbedingung $\sum_{i=1}^{n} x_i \cdot a_i \leq A$
- Weitere Nebenbedingungen der Form $x_i \Rightarrow x_j$ durch $x_i \leq x_j$ integrierbar

Handlungsreisendenproblem:

- m Städte
- c_{ij} ($1 \leq i,j \leq m$) Entfernung/Kosten der Reise von i nach j

Ziele: Finde eine Rundereise, die alle Städte erreicht, in der Ausgangsstadt endet und minimale Kosten aufweist

ILP-Formulierung: $\min_{y_{ij}} \left(\sum_{i=1}^{m} \sum_{j=1}^{m} y_{ij} \cdot c_{ij} \right)$

- m^2 Entscheidungsvariablen $y_{ij} \in \{0,1\}$, m ganzzahlige Variablen u_i (i,j=1,…,m)
- m Nebenbedingungen der Form $\sum_{j=1}^{m} y_{ij} = 1$
- m Nebenbedingungen der Form $\sum_{i=1}^{m} y_{ij} = 1$
- Für jede Untermenge $V \subset \{1, \cdots, m\}$: $\sum_{i,j \in V \times V} y_{ij} \leq |V| - 1$

Zur „exakten“ ILP-Formulierung Indizes (i,j) linearisieren!

Versorgungsproblem:

- m mögliche Orte, um Bedienzentren zu eröffnen
- c_i (i=1,…,m) Kosten am Ort i ein Bedienzentrum zu eröffnen
- k Kunden
- d_{ij} (i=1,…,k; j=1,…,m) Kosten, um Kunde i aus Zentrum j zu bedienen

ILP-Formulierung: $$\min \left(\sum_{i=1}^{k} \sum_{j=1}^{m} y_{ij} \cdot d_{ij} + \sum_{j=1}^{m} z_j \cdot c_j \right)$$

- m·(k+1) Entscheidungsvariablen $y_{ij}, z_j \in \{0,1\}$ (i=1,…,k; j=1,…m)
- m·k Nebenbedingungen der Form $y_{ij} \leq z_j$ für alle i, j
- k Nebenbedingungen der Form $\sum_{j=1}^{m} y_{ij} = 1$ für alle i

Zur ILP-Formulierung Indizes (i,j) linearisieren!

11.2 Lösungsansätze der kombinatorischen Optimierung

In diesem Kapitel: Allgemeine Prinzipen + Methoden zur Bewältigung/Lösung schwerer Probleme, in Anwendbarkeit nicht auf komb. Optimierung beschränkt

Offensichtliche Methode (endlicher Zulässigkeitsraum) ist vollständige Enumeration (exakte Methode):

- Systematische Untersuchung aller zulässigen Punkte (systematisch: z.B. mit Entscheidungsbaum, s. später)

  bester Punkt liefert Lösung

  => exponentieller Aufwand (worst case und auch in praxi)

- Illustration: exponentiell (und damit „schwer“)

  Lösungsraum im Falle binäre Optimierung $M=\{0,1\}^n$

  Problemgröße n=50, Größenordnung Aufwand $2^{50} \approx 10^{15}$

  bei zeitlichem Aufwand je Punkt von 1 msec

  Größenordnung Zeitaufwand 2^{50} msec ≈ 32000 Jahre

  => nur für kleine Probleme praktikabel!

Allgemeine Ansätze zur Lösung schwerer Probleme

Exakte Methoden

worst case nach wie vor exponentiell - praktisch u.U. polynomiell

- Begrenzung des Lösungsraumes,
  u.U. schrittweise (z.B. Schnittebenenverfahren)
- Divide & Conquer, „Teile und Herrsche“ im allg. Sinn
  Zerlegung des Problems in kleinere (i. allg. repetitiv) derart, dass
  - aus Lösung der Teilprobleme die Lösung des Gesamtproblems konstruierbar ist, jedoch mit insgesamt geringerem Aufwand
  - die Lösung des Gesamtproblems „in“ der Lösung eines Teilproblems zu suchen ist, mit geringerem Aufwand für gewisse Teilprobleme (z.B. Branch-and-Bound: B&B)
  - in gewissem Sinne auch „dynamische Optimierung“ (später)

und intelligente Kombinationen aus obigen Ideen (z.B. Branch & Cut)

Heuristische Methoden

erstrebenswerterweise polynomiell,

aber „approximativ“ in folgendem Sinne

- ohne Garantie für eine optimale Lösung
- aber i.a. (also „praktisch häufig“) gute Lösung
  erstrebenswerterweise mit Schranken,
  vorzugsweise „engen“ Schranken

Heuristische Verfahren sind meist problemspezifisch,
häufig mit folgenden Charakteristika

- Eröffnung: Bestimmung initialer Lösung
  (oft: zulässiger Lösung)
- Verbesserung: iterative Suche
  (oft: lokal, in der Nachbarschaft)
- Abbruch: vorzeitige Beendigung
  (oft: Anspruchs-/Grenz-abhängig)

und intelligente Kombinationen aus diesen Ansätzen

Vorgehen bei heuristischen Verfahren

- Eröffnungsverfahren
  - (wenn vorgesehen) bestimmen initiale (zulässige) Lösung Verfahren: greedy vs vorausschauend
- Verbesserungsverfahren
  - starten mit zulässiger Lösung **x**
  - iterieren über Nachbarschaftsbestimmung
    N(**x**):={**y**| **y** nahe bei **x**} und Untersuchung der Nachbarschaft
- brechen ab bei (festzulegendem) Kriterium

✓ Eröffnungs- und Verbesserungsverfahren können deterministisch oder stochastisch ausgestaltet sein

✓ Typische Schwäche reiner Verbesserungsverfahren liegt in Lokalität der Suche (bestimmt durch Def N(**x**))
=> Beschränkung auf lokale Optima

✓ Abhilfe durch „Metastrategien“: Stichworte in diesem Umfeld sind Simulated Annealing, Threshold Accepting, Tabu Search, Genetische/Evolutionäre Algorithmen ...

Exakte Verfahren: Branch-and-Bound Methoden

Sei ein Maximierungsproblem P_0 zu lösen: (Minimierung analog)

Hier P_0: max f(**x**), $\mathbf{x} \in W_0 \subseteq \{0,1\}^n$ mit zulässiger Lösungsmenge W_0

B&B beruht auf den folgenden Lösungsprinzipien

1) Branching

- zerteile Problem P_0 in Teilprobleme (TPs) $P_1,...,P_k$ und Lösungsmengen $W_1,...,W_k$ derart dass

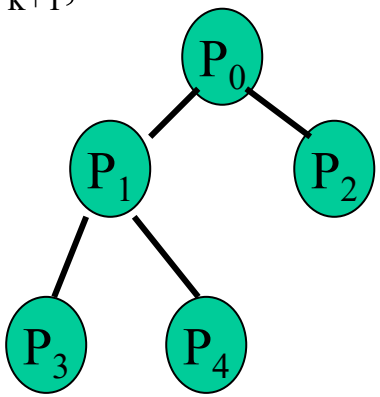
$$W_0 = \cup_{i=1}^{K} W_i \text{ und } W_i \cap W_j = \emptyset \quad (i \neq j)$$

- falls erforderlich, zerteile Probleme P_i weiter in Teilproblem $P_{k+1},...$ auf
- insgesamt resultiert Entscheidungs-(Wurzel)-Baum,

  in dem Teilprobleme als Knoten notiert und

  Teilprobleme eines Problems vom zugeordneten

  Knoten per branching erreicht werden
- Lösung aller Blattprobleme liefert (sicher) Gesamtlösung



2) Bounding

- für optimalen Lösungswert Z* sei untere Schranke U bekannt, $U \leq Z^*$
  - sowohl initial: schlimmstenfalls $-\infty$ oder besser aus Heuristik
  - als auch während des Ablaufs von B&B:
    U als unter Schranke für P_i

  fortlaufend bestmögliche Vergrößerung vorzunehmen
- für Teilprobleme P_i obere Z-Schranke $O_i \geq \max_{\mathbf{x} \in W_i} (f(\mathbf{x}))$
- sowie Optimum $O_i^* = \max_{\mathbf{x} \in W_i} (f(\mathbf{x}))$
- Schranken-/Wert-Ermittlung aus
  - entweder (direkt:) Heuristik
  - oder (Verzweigung:) Betrachtung P_i-Teilprobleme,

    wobei aus $W_i = \cup_k W_k$ folgt dass $O_i \leq \max_k O_k$ und $O_i^* = \max_k O_k^*$
- Teilprobleme gelten als weiter zu untersuchen, weiter zu verzweigen, ausgelotet (fathomed) oder abgearbeitet

- Teilproblem heißt ausgelotet (fathomed), wenn
  - $O_i < U$, d.h. die optimale Lösung kann nicht in W_i liegen

    ($O_i^* < U$) => P_i nicht weiter untersuchen / verzweigen, daher Abbruch der Bearbeitung des „Zweiges“ durch „Bounding“
  - $O_i^* > U$, d.h W_i-Lösung gefunden, besser als U

    => $U := O_i^*$ als Vergrößerung von U
  - $W_i = \varnothing$, d.h. P_i hat keine zulässige Lösung
- B&B-Varianten arbeiten zusätzlich mit

  oberer Schranke O und unteren Schranken U_i

Je besser die Schranken, desto mehr Zweige können auf Grund der Schranken verworfen werden

- B&B-Verfahren sind bzgl. folgender Komponenten zu konkretisieren (geschieht weitgehend problemspezifisch)
  - Regeln Initialisierung: initiale Schranken U, O
    (Heuristiken, z.B. Relaxation)
  - Regeln TP-Schranken: Schranken O_i, U_i
    (Heuristiken, z.B. Relaxation)
  - Regeln zur Reihenfolge der Auswahl zu verzweigender TP und zur Auslotung
    (Depth-First-Search, Breadth-First-Search, Maximum-Upper-Bound)
  - Regeln zur Problemaufteilung, Verzweigung
    (binäre, mehrere, wie ...)

11.3 Lösungsmethoden für typische Beispielprobleme

Beträchtliche Menge spezifischer Problemkreise „in diesem Kontext“ behandelt, u.a.

- Rucksackprobleme
- Travelling Salesman (Tourenplanungs-) Probleme
- Verschnittprobleme
- Scheduling- (Maschinenbelegungs-) Probleme
- Ressourcen-/Projekt-Planungsprobleme
- ...

Jeweils spezifische Ausprägungen der Lösungsalgorithmen, den vorgestellten Ideen folgend

Diese Veranstaltung fokussiert auf 2 Problemkreise:

1. Rucksackproblem
2. Scheduling-Probleme

Rucksackproblem (knapsack):

- n Gegenstände mit Gewichten $a_j>0$ und Wert $c_j>0$ (j=1,…n)
- Maximalgewicht A>0

Ziel: Packe den Rucksack so, dass sein Gewicht ≤A und sein Wert möglichst groß ist!

Problemlösungsmethoden

- von hoher theoretischer Bedeutung (Testfeld für Methodik, Methoden, Techniken, Aufwandsabschätzung, ...; breit untersucht)
- von gewisser praktischer Bedeutung (Frachtladung, Produktionsplanung, Maschinenbelegung)
- Problemformalisierung als binäres, endliches => kombinatorisches Optimierungsproblem
  (ILP-Formulierung siehe Folie 17)
- Wir betrachten nun problemangepasste Lösungsmethoden

Problem formal: max $\mathbf{c}^T\mathbf{x} = \min -\mathbf{c}^T\mathbf{x}$ udN $\mathbf{ax} \leq A$ und $\mathbf{x} \in \{0,1\}^n$

mit zulässiger Lösung $\mathbf{x}=\mathbf{0}$, d.h. Rucksack ist leer

Problem ist NP-schwer, Entscheidungsvariante ist NP-vollständig

Im folgenden: sinnvolle Ordnung der Gegenstände nach „spezifischem Wert“:

$c_1/a_1 \geq c_2/a_2 \geq ... \geq c_n/a_n$ in (WE/GE)

Heuristische Lösungen (hier typischerweise problemspezifisch)

Eröffnung:

Relaxation des Rucksackproblems gemäß $\mathbf{x} \in \{0,1\}^n$

$\Rightarrow 0 \leq x_j \leq 1$ ist lineares Optimierungsproblem („LP“)

$\Rightarrow$ Lösungsweg bekannt

Wir betrachten

- Einfache „Greedy“ Heuristik
- Branch and Bound-Verfahren

Optimale Lösung für LP sogar unmittelbar verfügbar

Auswahlkriterium: „wähle jede Volumeneinheit im Rucksack mit maximal möglichem spezifischem Wert“

=> (nach gew. Ordnung) erste k-1 Gegenstände im Rucksack, k-ten Gegenstand teilweise (Relaxation!), restliche nicht

Optimale Lösung LP formal mit Zielfunktionswert Z_{LP}, Gewicht A_{LP}:

$$k: \quad \sum_{j=1}^{k-1} a_j \leq A \quad \sum_{j=1}^{k} a_j > A$$

$$x_j^{LP} = \begin{cases} 1 & \text{falls } j < k \\ \frac{A - \sum_{j=1}^{k-1} a_j}{a_k} & \text{falls } j = k \\ 0 & \text{falls } j > k \end{cases}$$

$$Z_{LP} = \sum_{j=1}^{k-1} c_j + \left(A - \sum_{j=1}^{k-1} a_j\right) \frac{c_k}{a_k} \qquad A_{LP} = A$$

Daraus kann Näherungslösung für das ILP generiert werden

(Zielfunktionswert Z_{ILP} und Gewicht A_{ILP},

Z^*_{ILP} und Gewicht A^*_{ILP}, seien die optimalen Werte)

$$x_j^{ILP} = x_j^{LP} \text{ falls } j \neq k \quad x_k^{ILP} = 0$$

$$Z_{ILP} = \sum_{j=1}^{k-1} c_j \qquad A_{ILP} = \sum_{j=1}^{k-1} a_j$$

Es gilt $Z_{ILP} \leq Z^*_{ILP} \leq Z_{LP}$

<u>Verbesserung</u>

Freigebliebener Rest $A' := A - A_{ILP}$ kann mit Kandidaten aus der Gegenstandsmenge $\{j \mid j=k+1,...,n\}$ gefüllt werden

Greedy-Heuristik folgt „Richtung der Anordnung“

$A' = A - A_{ILP}$; $Z_H = Z_{ILP}$;
for j=k+1,…,n do
  if a[j] ≤ A′ then
    begin x[j] = 1 ; $Z_H = Z_H + c[j]$; $A' = A' - a[j]$; end
  else x[j] = 0 ;
$A_H = A - A'$;

Gesamtaufwand O(n log n) durch Initiale Sortierung

woraus insgesamt als Schranken folgt: $Z_H \leq Z_{ILP}^* \leq Z_{LP}$

Anmerkung: es gibt schärfere Schranken, hier nicht verfolgt

Rucksack-Problem: Branch-And-Bound Lösungen

(auch hier, typischerweise, problemspezifisch)

Notation wie zuvor:

Gegenstände sinnvollerweise nach spezifischem Wert geordnet indiziert, $c_1/a_1 \geq c_2/a_2 \geq \ldots \geq c_n/a_n$ mit i=1,...,n und
Indikatorvariable $x_i \in \{0,1\}$

<u>Problemaufteilung</u>

Problem/Teilproblem

= Knoten im Entscheidungsbaum

= Weg zu Knoten

wird charakterisiert durch (Teil-)Vektor Indikatorvariable $(x_1,...,x_s)$

wobei $1 \leq s \leq n$ oder Vektor ist leer (-)

Bedeutung: x_i in $(x_1,...,x_s)$ hat Wert

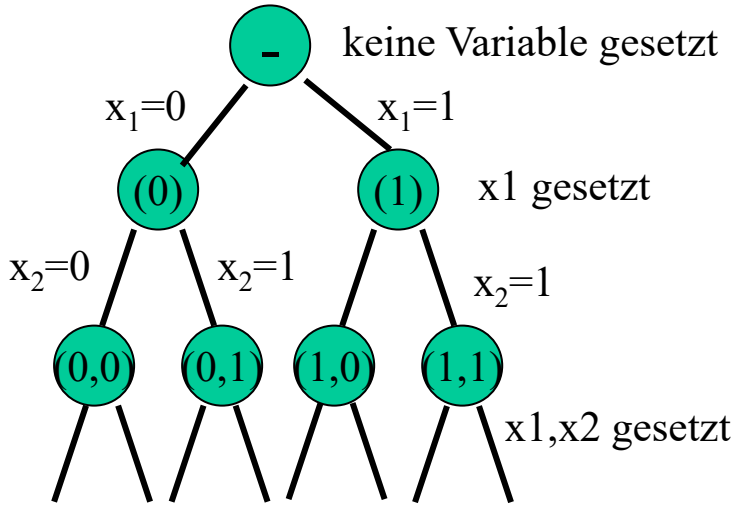
0 = Objekt i ist nicht im Rucksack, 1 = Objekt i ist im Rucksack

$(x_{s+1},...,x_n)$ frei: Entscheidung noch offen, für s=n ist Restliste leer

Initialisierung

- für Gesamtproblem
  (noch keine Variable gesetzt)
- wird mit Greedy Heuristik GH
  Anfangslösung $\mathbf{x}_H$, Z_H bestimmt
- dadurch untere Schranke $U=Z_H$ bekannt
- sei $(Z^+,\mathbf{x}^+) = (Z_H,\mathbf{x}_H)$ Variable für besten bisher berechneten Lösungskandidaten
- betrachte Zerlegung des Problems in (0) und (1)

binärer Entscheidungsbaum



Teilproblem-Schranken

- für $\mathbf{t}=(x_1,...,x_s)^T$
- berechne zugewiesenes Gewicht $A_{\mathbf{t}} = \sum_{i=1}^{s} x_i \cdot a_i$,
  erreichten Wert $Z_{\mathbf{t}} = \sum_{i=1}^{s} x_i \cdot c_i$
- ermittle obere Schranke für O_t für Z mittels Greedy-Heuristik
  für $A' = A - A_{\mathbf{t}}$, $Z_H = Z_{\mathbf{t}}$ über j = s+1,..., n mit Z_{LP} als Resultat, d.h. $O_{\mathbf{t}} = Z_{LP}$
- Anmerkungen:
  - Obere Schranke mittels relaxiertem Problem ist hier interessant, um uninteressante Teilbäume zu erkennen und zu verwerfen
  - Untere Schranken entstehen implizit durch den jeweils besten Lösungskandidaten, dienen als Vergleichswert für O_t
  - Bei $O_{\mathbf{t}} < Z^+$ ist Teilbaum uninteressant.

Teilproblem-Untersuchung, $\mathbf{t}=(x_1,...,x_s)^T$ mit A_t und Z_t

3 Fälle zu unterscheiden

1. $A_t > A$: Lösungsmenge leer
   $\Rightarrow$ TP ausgelotet kann nur bei $x_s=1$ auftreten
2. $A_t = A$: volle Zuweisung erreicht
   $\Rightarrow$ TP ausgelotet kann nur bei $x_s=1$ auftreten

   falls $Z_t>Z^+$ bessere Lösung gefunden $\Rightarrow (Z^+,x^+)=(Z_{\mathbf{t}},\mathbf{t})$
3. $A_t < A$: Optimum potentiell in **t**

   s=n: Aufteilung nicht mehr möglich,

   falls $Z_{\mathbf{t}}>Z^+$ bessere Lösung gefunden $\Rightarrow (Z^+,x^+)=(Z_{\mathbf{t}},\mathbf{t})$

   s<n: potentiell aufteilen, bestimme obere Schranke $O_{\mathbf{t}}$
   - $O_{\mathbf{t}} \leq Z^+$ dann ist Optimum nicht in **t** $\Rightarrow$ TP ausgelotet
   - $O_{\mathbf{t}} > Z^+$ dann ist Optimum potentiell in **t**

     $\Rightarrow$ TP zerteilen in $(x_1,...,x_s,0)^T$ und $(x_1,...,x_s,1)^T$ „Rekursionsfall“

u.U. Backtracking-Maßnahmen (Speichereffizienz)

Freiheitsgrad: Wahl der Bearbeitungsreihenfolge von Teilproblemen, die durch Zerteilung entstehen

unterschiedliche Strategien sinnvoll / empfohlen / bekannt:

- nach maximaler oberer Schranke
  wg. resultierender Chance Z^+ schnellstmöglich zu erhöhen
  Hoffnung: mehr Zweige wg $O_t < Z^+$ ohne Berechnung zu erledigen
- DFS in Abwandlung LIFO wg resultierender Chance zerteiltes Problem bzgl Schrankenermittlung „in nur leichter Abwandlung“ des Vaters zu untersuchen
  => Schrankenermittlung schneller

Terminierung: wenn kein TP mehr zu untersuchen, liefert (Z^+,x^+) optimale Lösung (Z^*,x^*)

Rucksack-B&B in vielen Varianten / mit vielen Verfeinerungen ausgiebig untersucht (vgl. Literatur)

Scheduling / Maschinenbelegungs-Probleme

Terminologie Scheduling / Maschinenbelegungsplanung:

- Jobs (Aufgaben/Aufträge) sind von
- Maschinen (Ressourcen, Geräten, Personen) zu bearbeiten

Vielfältige Interpretationen / Einsatzbereiche

- Fertigungsvorgänge in Werkstätten / Fabriken
- Ab-/Anflugvorgänge auf Rollbahnen (u.ä. Abfertigungen)
- Berechnungen/Bearbeitungen auf Prozessoren
  (Rechen- und Kommunikationssysteme)
- Kundenwünsche in Service-Einrichtungen (Warenhaus, Call-Center,…)

Allgemeine Fragestellung

Wann soll welcher Job (bzw einer seiner Arbeitsvorgänge: Tasks) auf welcher Maschine / welchen Maschinen bearbeitet werden, wenn bestimmte Zielfunktion gegeben und zu optimieren ist ?

Beispiele für Zielfunktionen sind

- Zeitspanne bis Fertigstellung letzter Job (makespan)
- mittlere Wartezeit (Zeit ohne Bearbeitung) von Jobs
- Überschreitung des Fertigstellungstermins
- Gesamtkosten der Bearbeitung
- …

Viele spezielle Klassen von Problemen bzgl der Art von Jobs und Maschinen, z.B.

- alle Jobs sind initial bekannt vs. Jobs tauchen zur Laufzeit auf
- Jobs sind unterbrechbar oder nicht unterbrechbar
  (dann ggfs. mit oder ohne Verlust weiterzubearbeiten)
- Jobs sind gleichwertig / wichtig oder von unterschiedlichem Wert

Achtung

Nur leichte Variation des Problemtyps kann bewirken, dass statt Lösungsalgorithmen mit polynomieller Laufzeit nur exponentielle bekannt sind und umgekehrt.

Notationen

n Jobs JM = {1,...,n} und m Maschinen MM={1,...,m}

zunächst:

- alle Jobs bei Einplanungsvorgang bekannt
- jeder Job zu jedem Zeitpunkt auf maximal 1 Maschine bearbeitet
- jede Maschine zu jedem Zeitpunkt bearbeitet maximal 1 Job
- keine Unterbrechungen

Potentielle Charakteristika, Attribute von Jobs sind

- die Bearbeitungsdauer: processing time $p_{ij} \geq 0$
  benötigte Arbeitszeit für Job j auf Maschine i
- der Bereitstellungstermin: release time $r_j \geq 0$
  frühester Bearbeitungsbeginn für Job j
- der Fälligkeitstermin: due date $d_j \geq 0$
  spätestes (gewünschtes) Bearbeitungsende j
- das Gewicht: weight $w_j \geq 0$
  Wichtigkeit / Priorität von j, $w_j > w_k$ dann sei w_j wichtiger als w_k

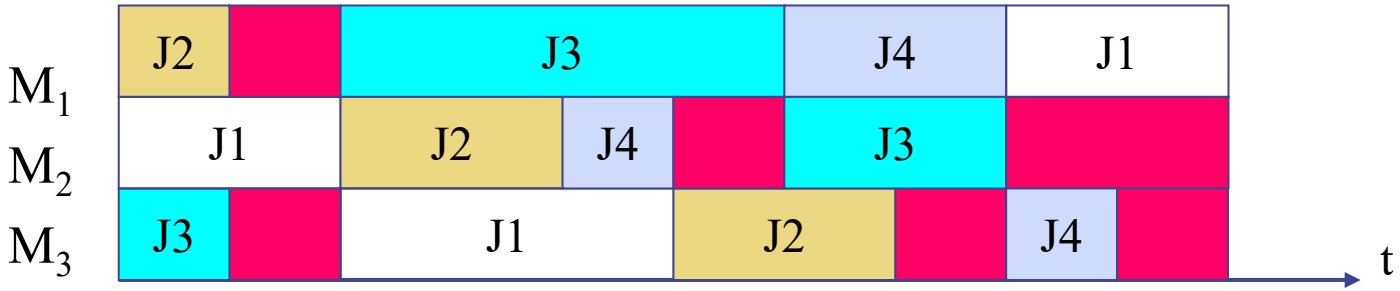
Größen seien ganzzahlig!

Nach Festlegung aller Zeitintervalle, in denen Maschinen Jobs zugewiesen wurden, existiert ein Schedule

(auch Bearbeitungsplan, Belegungsplan)

- für jeden Job: Folge von Zeitintervallen von Tasks auf Maschinen
- für jede Maschine: Job-(Task-) Reihenfolge

festgehalten z.B. in Balken- / Gantt-Diagrammen, hier n=4, m=3



Ein Schedule heißt

- zulässig, wenn alle Restriktionen des Scheduling-Problems erfüllt.
- optimal, wenn zulässig und Zielfunktion optimiert.

Aus (der Art der) Bearbeitung des Jobs j resultieren „Kosten“,

die (wie gewohnt) durch eine monoton wachsende (bzw. nicht fallende) Kostenfunktion $f_j : ?_+ \to ?$ beschrieben werden

d.h. $f_j(t)$ sind die anfallenden Kosten bei Beendigung von J_j zum Zeitpunkt t

Mit Schedule stehen Beurteilungsgrößen für Job j fest, z.B.

- Abschlusszeitpunkt: completion time C_j
- Durchlaufzeit, Verweilzeit: turnaround time, residence time
  $V_j=C_j-r_j$
- Verspätung: lateness $L_j=(C_j-d_j)^+$

Zielfunktionen mannigfaltiger Art, manche mit speziellen „Kennungen“

$$\max_{j=1}^{n} f_j(C_j) \ (:= f_{\max}) \ , \quad \sum_{j=1}^{n} f_j(C_j)$$ MinMax-, MinSum-Probleme

Mit spezieller Zielfunktion:

$$\max_{j=1}^{n} C_j \ (:= C_{\max}) \ , \quad \max_{j=1}^{n} L_j \ (:= L_{\max})$$

oder auch

$$\sum_{j=1}^{n} C_j \ , \quad \sum_{j=1}^{n} w_j \cdot C_j \ , \ \sum_{j=1}^{n} w_j \cdot U_j \text{ mit } U_j = \begin{cases} 1 & \text{falls } C_j > d_j \\ 0 & \text{sonst} \end{cases}$$

Kennzeichnung Scheduling Probleme

u.a. mit Tripel A|B|C wobei

- A die Maschinenkonfiguration beschreibt, z.B.
  - 1 = Einzelmaschine, P2 = 2 identische parallele Maschinen,
  - P = unbeschränkt viele parallele Maschinen
- B die Bearbeitungsspezifika angibt, z.B.
  - pmtn = preemptions, Jobunterbrechungen zugelassen
  - r_j = Bereitstellungstermine vorgegeben,
  - prec = precedence, Bearbeitungsreihenfolgen vorgegeben
  - tree = Bearbeitungsreihenfolge durch Baum/Wald vorgegeben
- C die Zielfunktion beschreibt, z.B. f_{max}, C_{max}, L_{max}

konkrete Beispiele

$1 \mid tree \mid \Sigma w_j C_j$; $1 \mid pmtn, r_j \mid f_{max}$; $P2 \mid\mid C_{max}$

Ein-Maschinen-Probleme

Restriktionen

Jobs werden nicht weiter in Tasks unterteilt

Jobs werden von derselben Maschine bearbeitet

=> Notation einfacher p_{ij} wird zu p_j

Satz 11.2
Jedes Ein-Maschinen-Problem ohne Bereitstellungstermine besitzt einen optimalen Schedule ohne Leerzeiten.
Dabei ist jeder optimale Schedule eines Problems ohne Unterbrechungen auch optimal für das entsprechende Problem mit zugelassenen Unterbrechungen.

Beweisskizze:

- f_j monoton steigend
- Zielfunktion ist monoton steigend, soll minimiert werden
- jede „Lücke“ im Schedule aufzufüllen,
- jede „Lücke“ in Bearbeitungsintervall aufzufüllen

Konkrete Probleme $1||C_{max}$: MinMax-Problem, Lösung trivial,
da für jede lückenlose Bearbeitung $C_{max} = \Sigma p_j$ konstant => optimal

Weitere konkrete Probleme

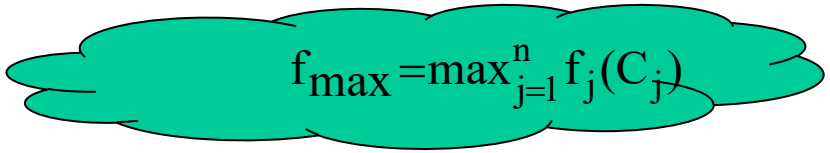
$1||L_{max}$: MinMax-Problem, mit Earliest Due Date Regel (EDD) gelöst, d.h. Jobs werden nach nichtfallenden Fälligkeitsterminen d_j bearbeitet

Aufwand: O(n log n) wegen des Sortierens

Warum ist EDD optimal für $1||L_{max}$?

- in allen Schedules tauchen alle Jobs auf,
- sei Π^* EDD Schedule, Π beliebiger anderer (nicht-EDD) Schedule => es existieren Jobs j, k mit $d_k \leq d_j$ welche
- in Π unmittelbar aufeinander folgen $\Pi = (..., j, k, ...)$
- in Π^* in umgekehrter Reihenfolge auftreten $\Pi^* = (..., k, ..., j, ...)$
- vertausche j und k in Π,
  wegen $d_k \leq d_j$ und $C_k \geq C_j$ in $\Pi \Rightarrow C_k - d_k \geq C_j - d_j \Rightarrow$
  durch die Vertauschung wird L_{max} nicht vergrößert
- Π^* kann aus Π mit endlich vielen Vertauschungen erstellt werden
  => $L_{max}(\Pi) \geq L_{max}(\Pi^*)$ und damit Π^* optimal

1| prec| f_{max}:



MinMax-Problem mit Lösung in $O(n^2)$

Beschreibung der Reihenfolge z.B. durch zyklenfreien gerichteten Graphen mit Jobs als Knoten, direkten Folgevorschriften als Kanten

Regel von Lawler:

„Unter allen noch nicht eingeplanten Jobs ohne Nachfolger, setze denjenigen Job an die letzte Stelle, der die geringsten Kosten in dieser Position verursacht."

Formal:

- sei $J \subseteq \{1,...,n\}$ Menge der noch nicht eingeplanten Jobs,
- $S \subseteq J$ Menge der Jobs aus J ohne Nachfolger, $p_J := \Sigma_{j \in J} p_j$
- dann wird Job $k \in S$ mit $f_k(p_J) = \min_{j \in S} f_j(p_J)$ nach Bearbeitung aller Jobs aus $J \backslash \{k\}$ ausgeführt.

Berechnung „rückwärts":

Starte mit JM, entferne jeweils letzen Job k

Regel liefert optimale Lösung, Beweisidee:

- Sei f^*_J Maximum der Jobabschlusskosten für optimale Reihenfolge.
- Bei Auswahl $k \in S$ gemäß Regel als letzter ausgewählt, ist minimaler Zielfunktionswert $\max(f_k(p_J), f^*_{J\setminus\{k\}})$.
- Aus $f_k(p_J) \leq f^*_J$ und $f^*_{J\setminus\{k\}} \leq f^*_J$ folgt Behauptung

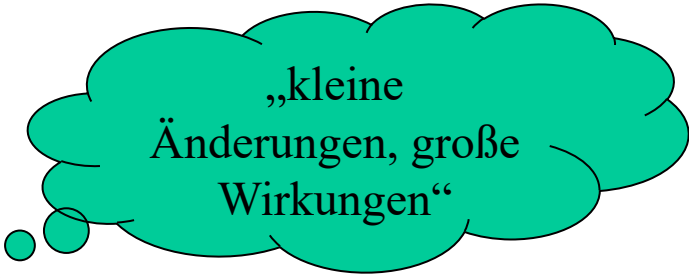
<u>Aufwand</u>:

- Annahme: Auswertung von f unabhängig von n
- n Mengen betrachtet,
- je Menge maximal n-mal $f_k(p_J)$ berechnen $\Rightarrow O(n^2)$

Aber

- $1|\, r_j|\, L_{max}$: MinMax-Problem, ist exponentiell
  Einführung von Bereitstellungsterminen r_j macht Problem „schwer“

Probleme mit polynomiellen Algorithmen sind:



- $1|\,|\, L_{max}$, also ohne Bereitstellungstermine
- $1|\, r_{j,}, p_{j,}=1|\, L_{max}$
  - Sonderfall: Bearbeitungsdauern alle gleich (= 1)
    Lösung mit erweiterter EDD-Regel in O(n log n)
    „Wähle in jedem Zeitpunkt t=0,1,... als nächsten einzuplanenden Job unter den verfügbaren Jobs j (also mit $r_j \leq t$) einen mit kleinstem Fälligkeitstermin d_j aus“
- $1|\, pmtn, prec, r_j \,|\, f_{max}$

  Einführen von Unterbrechungen macht Problem „einfacher“

  Algorithmus in $O(n^2)$, vgl Lawler ´82

$1||\Sigma C_j$: MinSum-Problem,

Shortest-Processing-Time-First (SPT) Regel:

Jobs werden nach nichtfallenden Bearbeitungsdauern bearbeitet

Regel liefert optimale Lösung, Beweisidee:

- betrachte Jobs j, k im SPT Schedule mit $p_j < p_k$,
- j starte unmittelbar vor k zum Zeitpunkt A
- $C = C_j + C_k = (A + p_j) + (A + p_j + p_k)$
- vertausche Reihenfolge der Jobs j und k,
- $C' = C_k' + C_j' = (A + p_k) + (A + p_k + p_j) > C$
  $\Rightarrow$ Abweichung von SPT vergrößert ggfs ΣC_j
  (bei $p_j = p_k$ passiert nichts)

Aufwand: O (n log n) wegen des Sortierens

$1||[\Sigma(C_j - r_j)/n]$: MinSum-Problem,

- Zielfunktion minimiert mittlere Durchlaufzeit,
- SPT ist wiederum optimal, da multiplikative, additive Konstanten in der Zielfunktion die Optimalitätsregeln nicht ändern

Mehrere Parallele Maschinen

Restriktionen

- Jobs werden nicht in Tasks unterteilt
- alle Jobs sind von demselben Typ von Maschine bearbeitbar
- m (zeitparallel arbeitende) Exemplare eines Maschinentyps
  - Unterschiede liegen ggfs in „Geschwindigkeiten“ s_{ij}, so dass die Ausführungszeit $s_{ij} \cdot p_{ij}$ maschinenspezifisch wird, Kennzeichnung: R
  - s_{ij}=c identische parallele Maschinen, Kennzeichung: P
    dort o.B.d.A s_{ij}=1 und $p_j = p_{ij}$ gesetzt
  - s_{ij}=s_i uniforme parallele Maschinen, Kennzeichnung: Q
    dort $p_j = s_i \, p_{ij}$, bei Unterbrechungen $p_j = \Sigma \, s_i \, p_{ij}$
- Jobs werden jeweils von maximal 1 Maschine bearbeitet, nicht von mehreren parallel

Große Bedeutung für das Scheduling paralleler Prozessoren, aber Problem oft exponentiell

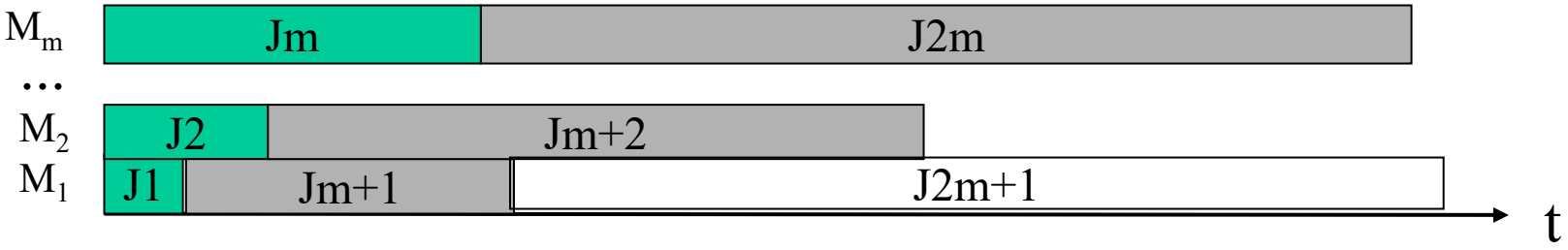
Betriebssystementwicklungen sind schon „daran“ gescheitert

- $P2 \mid\mid C_{max}$ bereits exponentiell $P2 \mid\mid \Sigma w_j C_j$: ebenfalls
- Bei Zulassung von Unterbrechungen existieren oft polynomielle exakte Lösungen

$P \mid\mid \Sigma C_j$, MinSum-Problem, beliebig (aber endlich: m) viele identische parallele Maschinen

- SPT Analogon (nach Conway): Jobs geordnet nach nicht-fallenden Bearbeitungszeiten $p_1 \leq ... \leq p_n$
- werden in dieser Reihenfolge für die Maschine eingeplant,
- die frühest möglich verfügbar ist
  (falls nicht eindeutig, kleinster M-Index)

Conway-Regel liefert Schedule



Conway-Regel ist optimal für $P||\ \Sigma C_j$

- Maschine i seien n_i Jobs zugewiesen, i=1,...,m, $\Sigma\ n_i=n$ mit aufeinanderfolgenden Bearbeitungen der Jobs $j_{i,1},...,j_{i,n_i}$
- auf M_i sind die Abschlusszeitpunkte $C_{i,1}=p_{i,1}$, $C_{i,2}=p_{i,1}+p_{i,2,}$ ... und $C_{i,n_i}=p_{i,1}+...+p_{i,n_i}$ und Gesamtsumme $\sum_{i=1}^{m} C_i = \sum_{i=1}^{m}\sum_{k=1}^{n_i}(n_i-k+1)p_{i,k}$
- Summe der Produkte aus Bearbeitungszeiten (jede 1mal) und Faktoren (ganzzahlig)
- minimiert für (nichtsteigende) Faktoren, Bearbeitungszeiten nicht fallend

Satz 11.3 (MinSum Probleme mit / ohne Unterbrechung)

Ist ein Schedule optimal für $P \mid\mid \Sigma w_j C_j$

dann ist er auch optimal für $P|pmtn| \Sigma w_j C_j$

Beweisidee:

Bei identischen Maschinen lässt sich die gewichtete Summe der Abschlusszeiten nicht durch Unterbrechung und Verschiebung verkleinern
(plausibel, weil Maschinen identisch sind)

$\Rightarrow$

$P|pmtn| \Sigma C_j$, MinSum-Problem, optimal mit Conway-Regel lösbar

Für 1| |ΣC_j und P | |ΣC_j (MinSum) konnten mittels SPT-Regel optimale Pläne erstellt werden

- Kann man das Vorgehen auf MinMax-Probleme übertragen?

Übertragung der Idee von SPT auf P| |C_{max},

- Largest Processing Time First (LPT),
  d.h. große zuerst => zum Schluss kleine gut verteilbar,

Ordne Jobs so, dass $p_1 \geq p_2 \geq \ldots \geq p_n$;
For i=1,..,m do $t_i = 0.0$;
For j=1,…,n do
    bestimme kleinstes $k \in \{1,\ldots,m\}$ mit $t_k = \min_{i=1,\ldots,m} t_i$;
    bearbeite Job j auf M_k im Intervall $[t_k, t_k+p_j]$;
    $t_k = t_k+p_j$;

Idee trägt leider nicht, exakte P||C_{max} Lösung exponentiell,

LPT aber gute Heuristik (max. rel. Fehler (1-1/m)/3, ohne Beweis)

Unterbrechungen vereinfachen das Problem!

P|pmtn ||C_{max}, MinMax-Problem, in O(n) exakt lösbar

Untere Schranken für C_{max}:

- $C_{max} \geq$ maximale Bearbeitungszeit eines Jobs
- $C_{max} \geq$ mittlere Belegungszeit der Maschinen

Also $$C_{\max} \geq C' = \max\left(\max_{j=1}^{n} p_j, \frac{1}{m}\sum_{j=1}^{n} p_j\right)$$

<u>McNaughton-Schedule</u>:

- verweise Jobs (in beliebiger Reihenfolge) lückenlos an Maschinen, d.h. eine Maschine nach der anderen, bis C´ erreicht (je Maschine),
- dann Unterbrechung und verweise Rest an die nächste Maschine

$\Rightarrow$ realisiert C´ mit maximal m-1 Unterbrechungen und

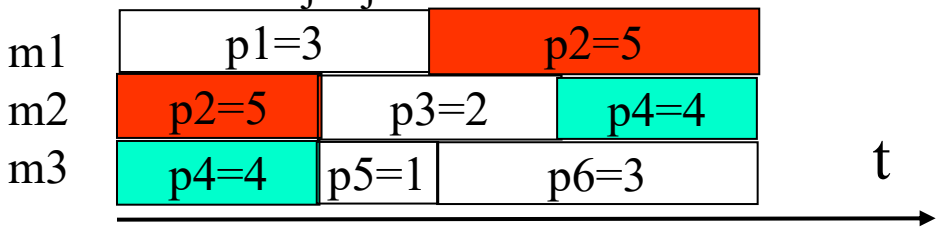
Leerzeiten der Gesamtlänge $$L \leq \max\left(0, m\left(\max_{i=1}^{n} p_i\right) - \sum_{i=1}^{n} p_i\right)$$

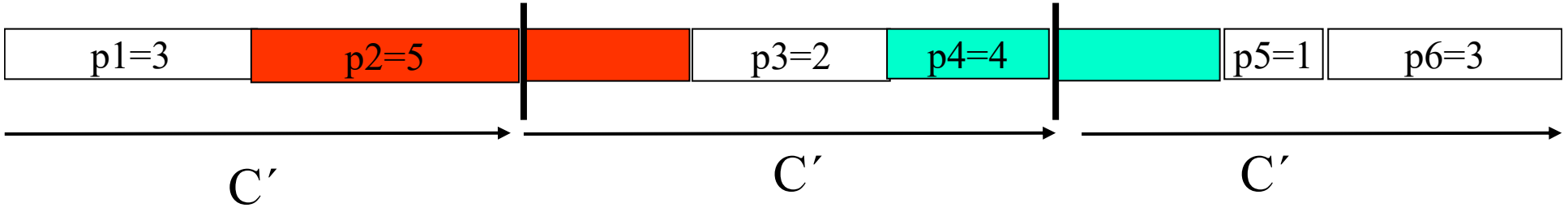
Beispiele für Mc Naughton bei m=3 Maschinen und 6 Jobs

Job j: 1 2 3 4 5 6 liefert $\Sigma_j p_j = 18$ und $\max_j p_j = 5$

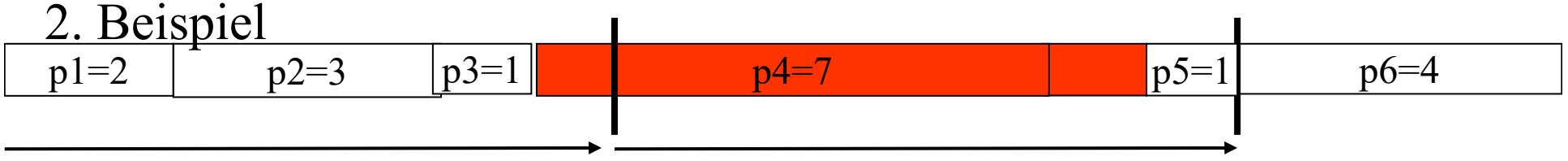
p_j: 3 5 2 4 1 3 $C' = (1/m) \Sigma_j p_j = 6$ für m=3

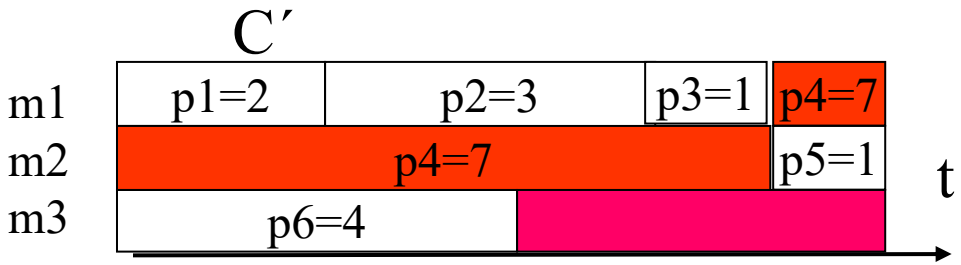
zugehöriger Plan





2. Beispiel





wg $\max_j p_j > (1/m) \Sigma_j p_j$

Leerzeiten L=3*7-18=3

Openshop-, Flowshop- und Jobshop-Probleme

Restriktionen

- $m > 1$ Maschinen $M_1,...,M_m$
- Jobs j, die aus Tasks /Arbeitsvorgängen O_{ij} bestehen, wobei die Zuordnung zu Maschinen M_i vorgeschrieben ist, so dass zu Task O_{ij} Bearbeitungsdauer p_{ij} bekannt ist
- wie bisher: zu jedem Zeitpunkt auf jeder Maschine nur 1 Job und jeder Job auf höchstens 1 Maschine bearbeitet
- Reihenfolgevorschriften bewirken Teilproblemklassen
- Openshop, Kennzeichen „O“
  Reihenfolge der Tasks von Jobs j nicht vorgeschrieben
- Flowshop, Kennzeichen „F“
  Reihenfolge der Tasks von Jobs j vorgeschrieben und für alle Jobs j identisch ($\rightarrow$ Maschinen entsprechend nummeriert)
- Jobshop, Kennzeichen „J“
  Reihenfolge der Tasks von Jobs j vorgeschrieben aber nicht notwendigerweise für alle Jobs j identisch

Übersicht I

Problem	Algorithmus	Rechenaufwand
$1 \mid \mid L_{max}$	EDD-Regel	O(n log n)
$1 \mid prec \mid f_{max}$	Lawler	$O(n^2)$
$1 \mid r_j, p_j=1 \mid L_{max}$	Erweiterte EDD-Regel	O(n log n)
$1 \mid \mid \Sigma C_j$	SPT-Regel	O(n log n)
$1 \mid \mid \Sigma w_j C_j$	Quotientenregel v. Smith	O(n log n)
$1 \mid tree \mid \Sigma w_j C_j$	Horn	O(n log n)
$1 \mid \mid \Sigma U_j$	Hodgson und Moore	O(n log n)

Anmerkungen:

- Übersicht aus Neumann/Morlock, Kap. 3.6
- aus dieser Auflistung von Problemen und Algorithmen
  haben wir nur eine Auswahl angesprochen,
  in Neumann/Morlock dazu mehr ...

z.B. Quotientenregel v. Smith ist Verallgemeinerung von SPT bzgl $q_j=p_j/w_j$
Verfahren von Horn baut auf Smith auf, wählt Kindknoten l mit minimalem q_l
als Nachfolger von Elternknoten k, betrachtet (k,l) als neuen Knoten im Tree.

Übersicht II

Problem	Algorithmus	Rechenaufwand
P | | ΣC_j, P |pmtn| ΣC_j	Conway	O(n log n)
Q | | ΣC_j,	Horowitz und Sahni	O(n log n)
Q | pmtn | ΣC_j	Gonzales	O(n log n+ m n)
P | | C_{max}	LPT-Regel (Heuristik)	O(n log m n)
P | pmtn | C_{max}	McNaughton	O(n)
P | tree, p_j=1 | C_{max}	Hu	O(n)
O2||C_{max}, O2|pmtn|C_{max}	Gonzales und Sahni	O(n)
F2||C_{max}, F2|pmtn|C_{max}	CDS (Heuristik)	O(m n log n)
J2||C_{max}	Jackson	O(n log n)
J||C_{max}	Giffler und Thomson (Heuristik)	O(m n^2)

Anmerkungen:

- Heuristik: Problem ist in NP („schwer“),
  polynomielle Laufzeiten nur mit approximativen Algorithmen erzielt

11.4 Gemischt ganzzahlige Probleme

Allgemeine Form:

$$\min_{\mathbf{x},\mathbf{y}} \left(\mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y}\right)$$

$$\mathbf{Ax} + \mathbf{By} \leq \mathbf{b}$$

$$\mathbf{x} \in \mathbb{R}_{\geq 0}, \mathbf{y} \in \mathbb{N}_0$$

Lineares Problem mit ganzzahligen und reellen Komponenten
Mixed Integer Linear Program (MILP)

- Kann durch entsprechende Modellierung genutzt werden um komplexe (auch nichtlineare) Probleme zu approximieren
- Lösung in der Regel NP schwer, aber
  - Gute Schranken können für viele Probleme in akzeptabler Zeit gewonnen werden
  - Mit heutigen Rechnern und Algorithmen können oft sogar globale Optima berechnet werden

11.4.1 Modellierung mit MILPs

Bisher bereits vorgestellt Modellierung von

- Variablen $x_i \geq u > 0$
- Variablen $x_i \leq 0$
- Unbeschränkten Variablen x_i

} auf ganzzahlige Variablen übertragbar

Mit ganzzahligen/binären Variablen darstellbar

- ✓ Logische Verknüpfungen
- ✓ Minimum, Maximum
- ✓ If … then …, Disjunktion
- ✓ k aus m Alternativen
- ✓ Eingeschränkte Produkte von Variablen
- ✓ Auswahl des Wertes einer Variablen
- ✓ Approximation nicht linearer Zielfunktionen und Nebenbedingungen

- Logische Verknüpfungen

  Variablen x, y, z $\in \mathbb{N}_0$ mit den Nebenbedingungen

  $x \leq 1, y \leq 1$ (binäre Variablen), dann

  - not x $\Rightarrow$ MILP Formulierung $z = 1 - x$
  - $z = x \wedge y = \min(x, y)$
  - $z = x \vee y = \max(x,y)$

- Minimum, Maximum

  Variablen $x_1, x_2 \in \mathbb{R}_{\geq 0}$ oder $\mathbb{N}_0$ mit $x_i \leq U_i$

  definiere $d_1, d_2 \in \mathbb{N}_0$, so dass $d_i = 1$, falls x_i Minimum/Maximum

  und $y \in \mathbb{R}_{\geq 0}$ oder $\mathbb{N}_0$ zur Aufnahme des Minimums/Maximums

  - $y = \min(x_1, x_2) \Rightarrow$ MILP Formulierung

    $y \leq x_1, y \leq x_2, y \geq x_1 - U_1(1-d_1), y \geq x_2 - U_2(1-d_2), d_1+d_2 = 1$
  - $y = \max(x_1, x_2) \Rightarrow$ MILP Formulierung

    $y \geq x_1, y \geq x_2, y \leq x_1 + U_1(1-d_1), y \leq x_2 + U_2(1-d_2), d_1+d_2 = 1$

Im Folgenden ist A_i von der Form $\Sigma_j a_{ij}x_j \leq b_i$

- If A_1 then A_2

  MILP Formulierung:

  definiere binäre Variable d und „große“ Konstanten M_1, M_2 und „kleines“ ε

  $\Sigma_j a_{1j}x_j \geq b_1 + \varepsilon - dM_1$ und $\Sigma_j a_{2j}x_j \leq b_2 + (1\text{-}d)M_2$

- Disjunktion: either A_1 or A_2

  MILP Formulierung:

  definiere binäre Variable d und „große“ Konstanten M_1, M_2 und

  $\Sigma_j a_{1j}x_j \leq b_1 + dM_1$ und $\Sigma_j a_{2j}x_j \leq b_2 + (1\text{-}d)M_2$

- mindestens k aus m Bedingungen A_i (i=1,..m)

  MILP Formulierung:

  definiere binäre Variablen d_i und „große“ Konstanten M_i (i=1,..m)

  $\Sigma_j a_{ij}x_j \leq b_i + d_iM_i$ und $\Sigma_i d_i \leq m - k$

- Eingeschränkte Produkte $y = dx$ (d binär, $x \in \mathbb{R}_{\geq 0}$)

  MILP Formulierung: Sei $x \leq U$

  $y \leq dU$, $x-y \leq (1-d)U$

  Erweiterung auf Produkte $y = d_1 \ldots d_M x$

  MILP Formulierung:

  $y \leq d_i U$ $(i=1,\ldots,M)$, $x-y \leq (M-\Sigma_i d_i)U$

- *Regel SOS 1 (special ordered set)*

  Auswahl des Wertes einer Variablen $d_i = 1$ (d_i binär) $\Leftrightarrow x \in [c_{i-1}, c_i)$ $(i=1,\ldots,K)$

  MILP Formulierung: $x < (1-d_i)M+c_i$, $x \geq d_i\, c_{i-1}$ $(i=1,\ldots K)$ und $\Sigma_i\, d_i = 1$

  Typische Anwendung: stückweise konstante Kosten

  Sei x die produzierte Menge, ax der Erlös

  ferner treten Kosten b_i auf, falls $x \in [c_{i-1}, c_i) \Rightarrow$ definiere Indikatoren d_i

  minimiere Zielfunktion $f(x) = -ax + \sum_{i=1}^{K} d_i b_i$

- *Regel SOS 2 (special ordered set 2)*

  Auswahl von y_{i-1}, y_i, so dass für $x \in [c_{i-1}, c_i]$ (i=1,…K):

  $x = y_{i-1}c_{i-1} + y_i c_i$ (Darstellung als Linearkombination der Intervallgrenzen

  MILP Formulierung: $y_i \in \mathbb{R}_{\geq 0}$ (i=1,...,K),

  $\Sigma_i y_i = 1$, $\Sigma_i y_i c_i - x = 0$ (garantiert Darstellung als Linearkombination)

  Zusätzliche Bedingung notwendig, um Grenzen eines Intervalls zu wählen:

  $y_{i-1} + y_i = 1$ für genau ein $i \in \{1,...,K\} \Rightarrow$

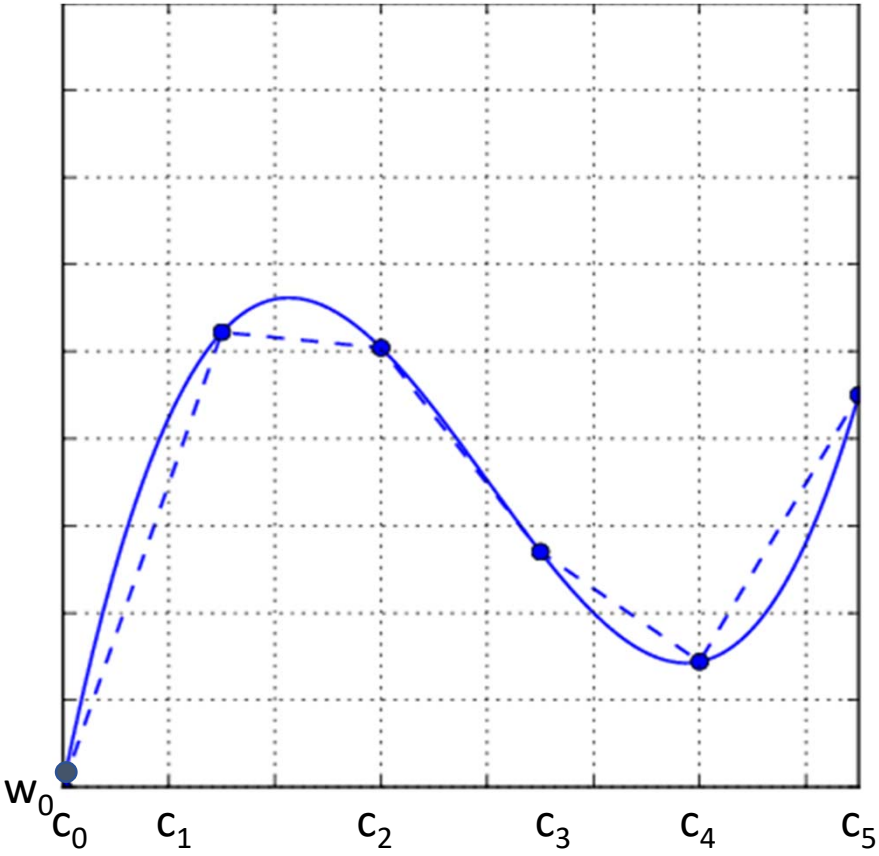
  binäre Variablen d_i (i=1,...K) und die Bedingungen

  $\Sigma_i d_i = 1$, $\Sigma_i d_i (y_i + y_{i-1}) = 1$

  nur von dem Intervall erfüllbar, in dem x liegt

Approximation nichtlinearer Funktionen

⇒ stückweise konstante Funktionen



MILP-Formulierung:

Füge Variablen y_i hinzu, so dass

$x = y_{i-1}\, c_{i-1} + y_i\, c_i$

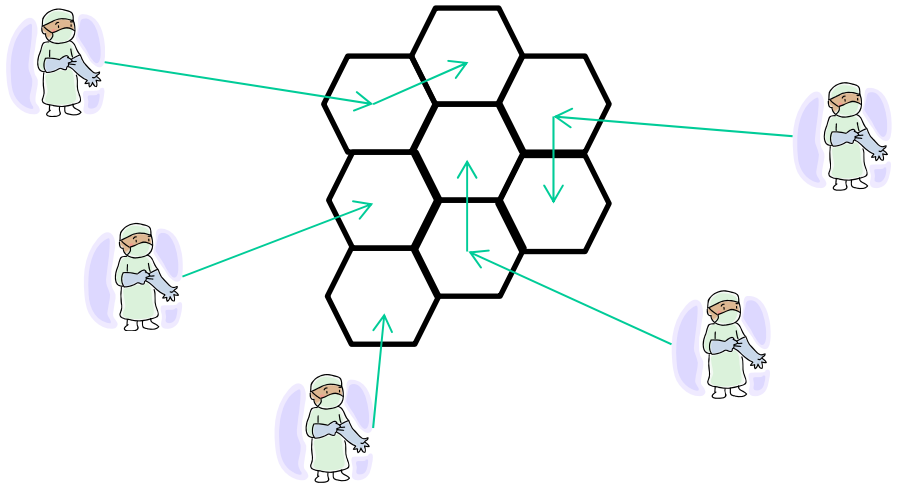
(siehe vorherige Folie)

Definiere Zielfunktion

$$\min_{d_1,\cdots,d_K,y_0,\cdots,y_K} \left(\sum_{i=1}^{K} d_i (y_i w_i + y_{i-1} w_{i-1}) \right)$$

Beispiel:

- Ausbruch einer Infektion in N Bezirken
- Es stehen M Untersuchungsteams zur Verfügung
- Team i kann Bezirk j in t_{ij} Zeiteinheiten untersuchen
- Ein Team kann 0, 1, oder 2 Bezirke untersuchen
- Wenn Team i die Bezirke j_1 und j_2 untersucht, so wird eine Fahrzeit s_{j1j2} zusätzlich benötigt
- Ziel: minimiere die Zeit, bis alle Bezirke untersucht wurden



Mathematisches Modell:

Entscheidungsvariablen $d_{ij} \in \mathbb{N}_0$ (i=1,…,M; j=1,…,N) binär (d.h. $d_{ij} \leq 1$)

Weitere Nebenbedingungen:

$$\underbrace{\sum_{j=1}^{N} d_{ij} \leq 2}_{\text{Pro Team max. 2 Bezirke}} \text{ und } \underbrace{\sum_{i=1}^{M} d_{ij} = 1}_{\text{Pro Bezirk genau 1 Team}}$$

Zielfunktion:

$$\underbrace{\min_{i \in \{1,\ldots,M\}}}_{\text{Kein Standard-Minimierungs-problem}} \left(\sum_{j=1}^{N} t_{ij} d_{ij} + \sum_{j_1=1}^{N} \sum_{j_2=1, j_1 \neq j_2}^{N} \underbrace{s_{j_1 j_2} d_{i j_1} d_{i j_2}}_{\text{Nicht linear}} \right)$$

Linearisiertes Modell:

Hilfsvariablen $w_{ij1j2} \in \mathbb{N}_0$ (i=1,…,M; j_1, j_2=1,…,N,$j_1 \neq j_2$) binär, $\theta \in \mathbb{R}_{\geq 0}$

Zusätzliche Nebenbedingungen:

$$0 \leq w_{ij_1j_2}, w_{ij_1j_2} \leq d_{ij_1}, w_{ij_1j_2} \leq d_{ij_2} \text{ und } w_{ij_1j_2} \geq d_{ij_1} + d_{ij_2} - 1$$

$w_{ij1j2} = d_{ij1}\, d_{ij2}$

$$\theta \geq \sum_{j=1}^{N} t_{ij} d_{ij} + \sum_{j_1=1}^{N} \sum_{j_2=1, j_1 \neq j_2}^{N} s_{j_1 j_2} w_{ij_1 j_2} \quad (i = 1, \ldots, M)$$

$\theta \geq$ max. Zeit, die ein Team benötigt

Neue Zielfunktion: $\min_{\theta} (\theta)$

11.4.2 Lösungsansätze für MILPs

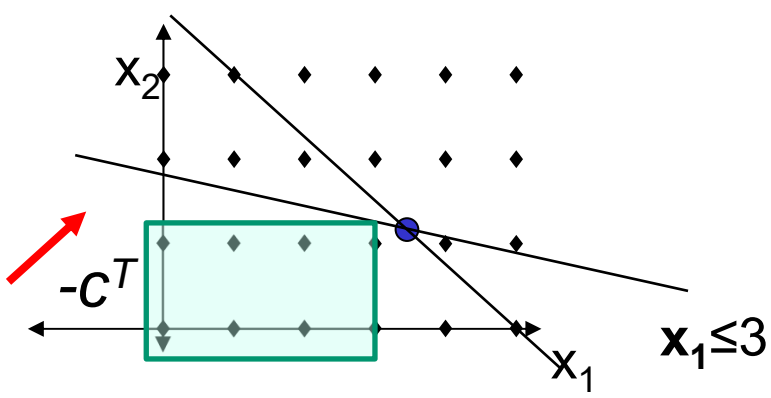
Erweiterung der bereits vorgestellten Ansätze

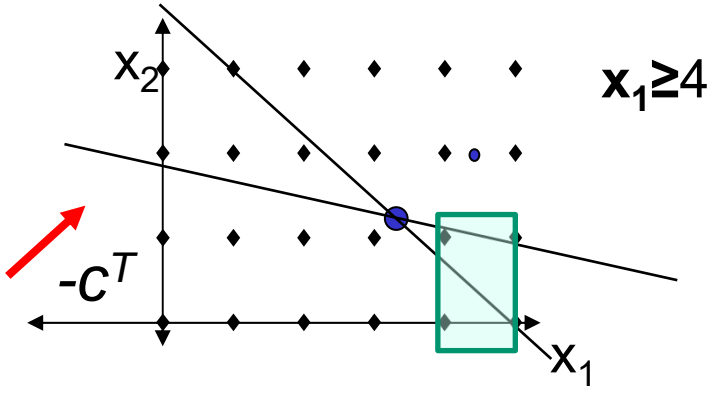
- LP-Lösung relaxierter Probleme (primal – dual)
- Branch-and-bound Ansätze
- Schnittebenen (Branch-and-cut)
- Heuristiken für die Auswahl von Schnittebenen, Branching-Punkten, ..
- A priori Problemreduktion oder –dekomposition

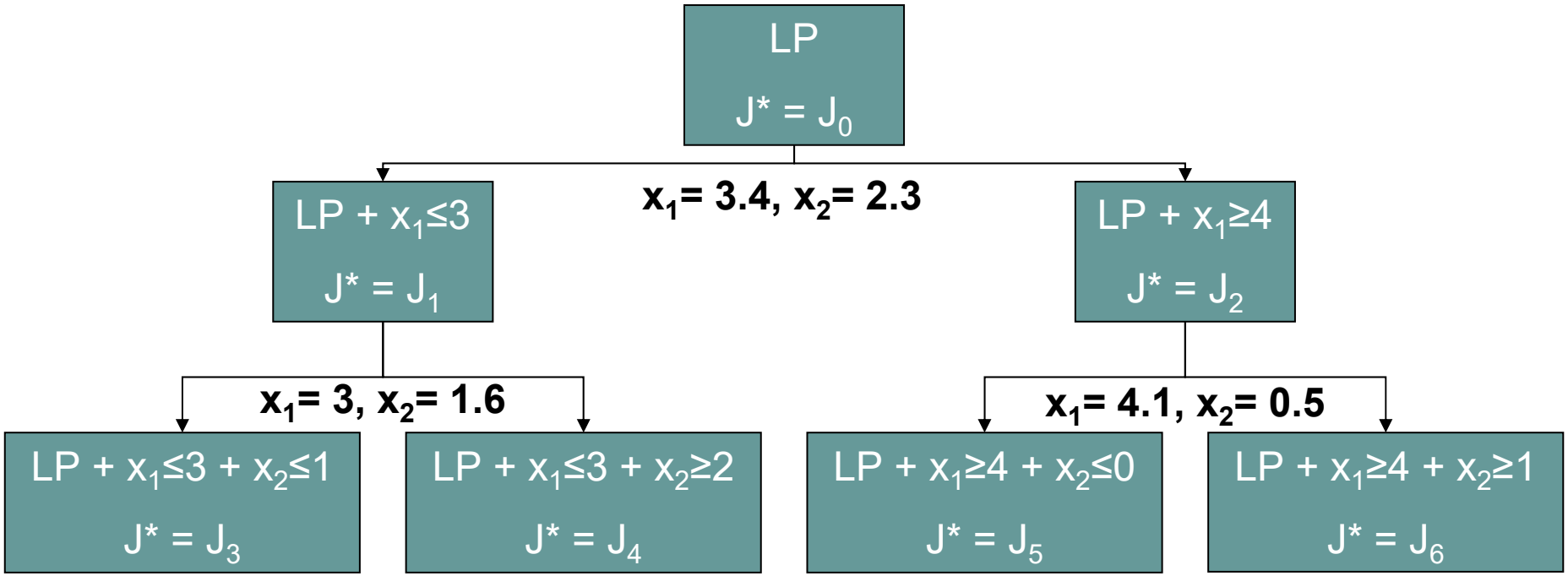
Grundlage der Optimierung

- ✓ Die größte LP Lösung des relaxierten Problems ist eine untere Schranke für das Minimum
- ✓ Die beste bekannte Lösung, die alle Nebenbedingungen erfüllt ist eine obere Schranke für das Minimum

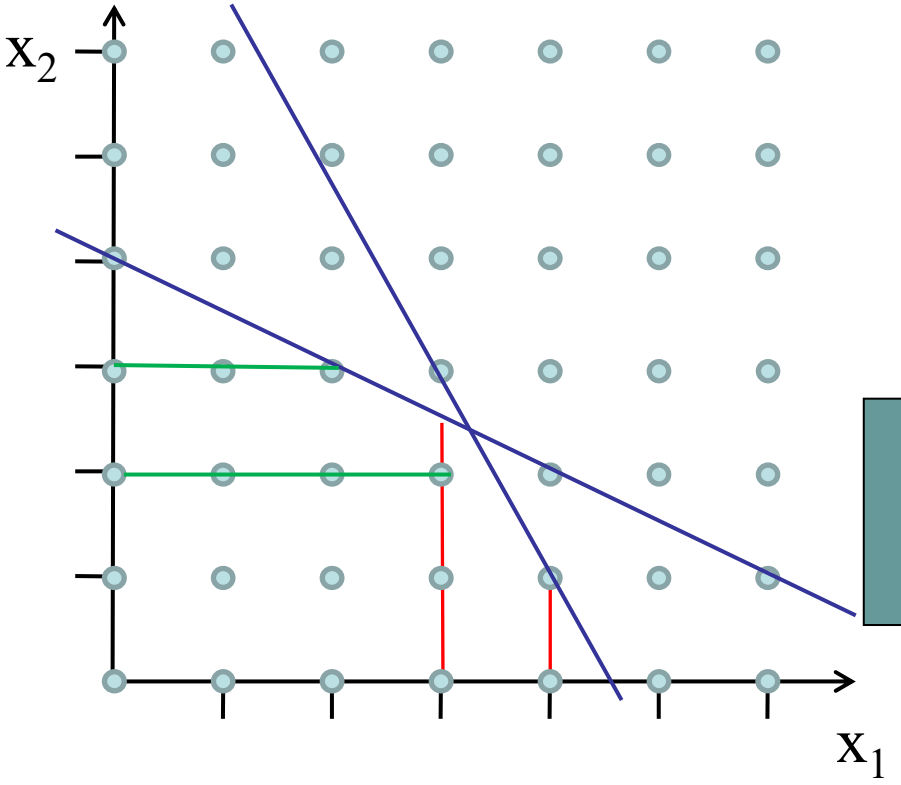
Grundlagen des Branch-and-bound Verfahrens

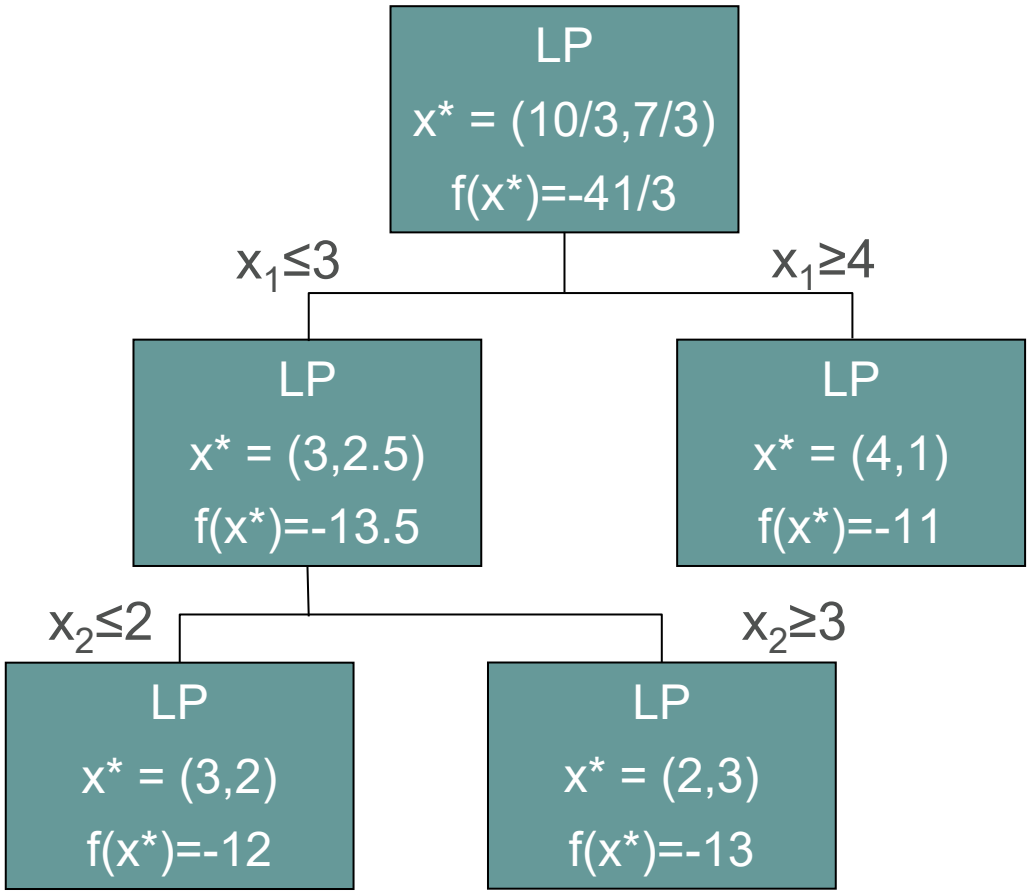






Branch & bound für das Beispiel von Folie 6





Lösungsschritte eines MILP-Lösers:

1. Reduktion der Variablen und Nebenbedingungszahl
2. Lösung des relaxierten linearen Programms
3. Vorverarbeitung des MILP-Problems
4. Hinzufügen von Schnittebenen
5. Heuristiken zum Finden zulässiger Lösungen
6. Branch-and-Bound Verfahren
   Aufteilung in Subprobleme durch Aufteilung der zulässigen Bereiche einzelner Variablen
   Analyse der Subprobleme durch
   - Lösung des relaxierten Problems
   - Weitere Aufteilung
   - Abbruch
     durch Unlösbarkeit oder zu große untere Schranke