Kapitel 5

Mathematische Optimierungsmodelle

Ganzzahlige Optimierung - Grundlagen

Uwe H. Suhl
Lehrstuhl für Wirtschaftsinformatik
Freie Universität Berlin

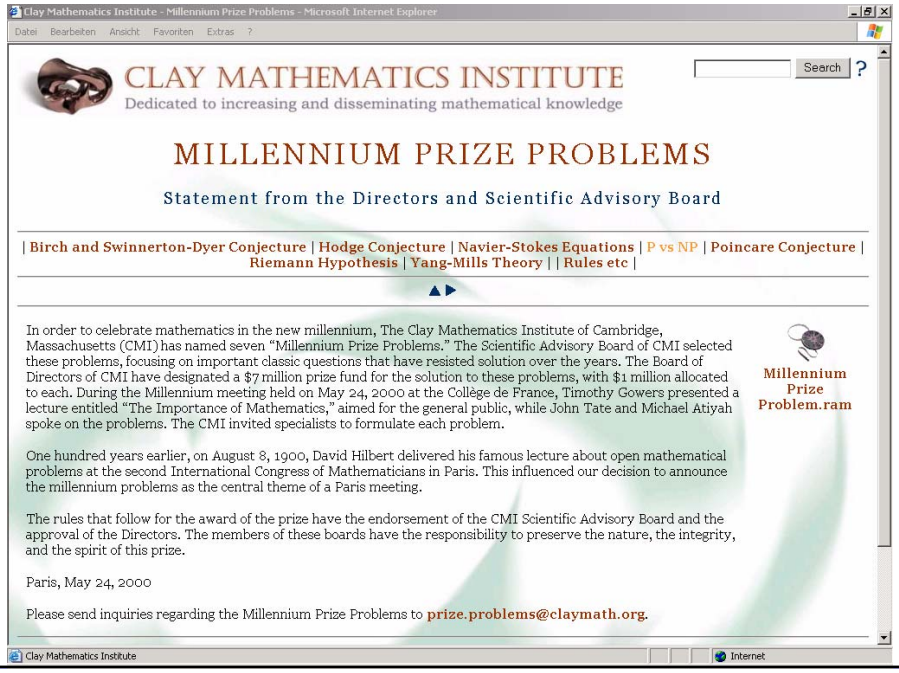
Optimierungssysteme
Version 1.1 / SS 2008

Gemischt-ganzzahlige Optimierung (IP)

- gemischt-ganzzahlige Modelle unterscheiden sich von LP-Modellen dadurch, daß einige Variablen nur ganzzahlige Werte annehmen dürfen
- der wichtigste Fall sind 0-1-Variablen; sie repräsentieren Entscheidungen über die Durchführung einer Aktivität
- viele *Modellierungstechniken* erlauben es, fast jedes deterministische Entscheidungsproblem in einem IP-Modell abzubilden
- obwohl IP-Modelle formal sehr ähnlich zu LP-Modellen sind, gehören sie (anders als LP) zur Klasse der NP-harten Probleme, für die keine effizienten Algorithmen bekannt sind
- alle Softwaresysteme zur Lösung von IP-Modellen basieren auf Branch-and-Bound-Algorithmen, bei denen an jedem Knoten des Baumes das zugehörige LP-Problem gelöst wird
- ein Teilproblem ist untersucht, wenn das LP ganzzahlig oder unzulässig ist oder dessen Zielfunktionswert schlechter als der Wert einer bereits gefundenen IP-Lösung ist
- die Algorithmen lösen also statt einem LP u.U. sehr viele LPs; im schlechtesten Fall, müssen 2^n LPs gelöst werden, n: Anzahl 0-1-Variablen
- *kritisch für die Lösbarkeit von IP-Modellen ist die Modellformulierung*

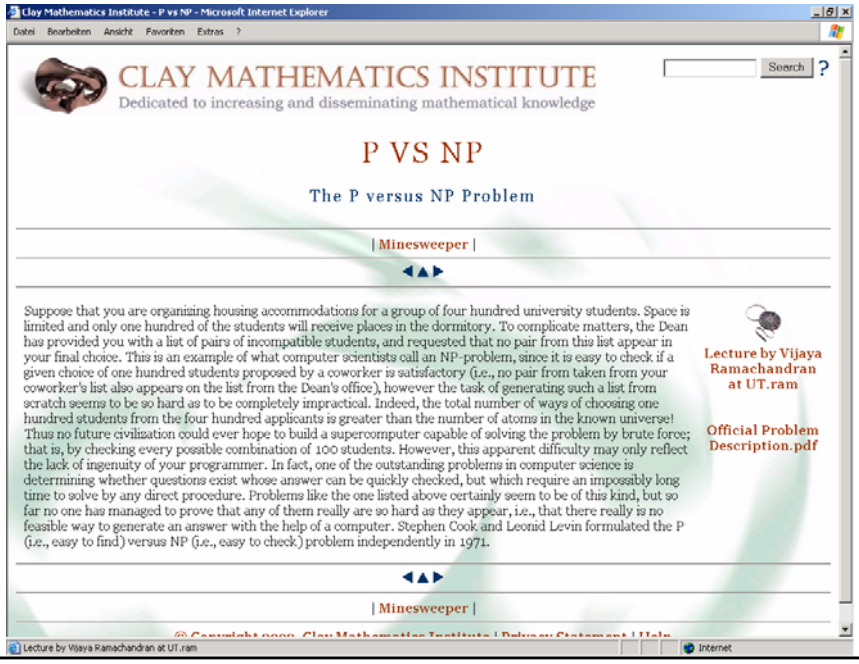
Millenium Prize Problems: US $ 7.000.000,-

- http://www.claymath.org/prizeproblems



P versus NP Problem

- **One Millenium Price Problem**



NP-Probleme: http://www.claymath.org/prizeproblems

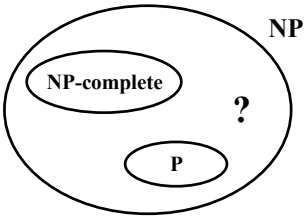
- Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. Two students have to share one room.
- To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appears in your final choice.
- This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair from taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical.
- Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking every possible combination of 100 students.
- However, this apparent difficulty may only reflect the lack of ingenuity of your programmer.
- In fact, one of the outstanding problems in computer science is determining whether questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure.
- Problems like the one listed above certainly seem to be of this kind, but so far no one has managed to prove that any of them really are so hard as they appear, i.e., that there really is no feasible way to generate an answer with the help of a computer.
- Stephen Cook and Leonid Levin formulated the P (i.e., easy to find) versus NP (i.e., easy to check) problem independently in 1971

NP-vollständige Probleme

- Oben wurde ein Beispiel vorgestellt, für das man allerdings einen gegebenen Lösungsvorschlag auf Zulässigkeit schnell testen kann
- Solche Modelle gehören zur Klasse NP (nichtdeterministisch polynomial)
- Die Menge NPC von NP-vollständigen (NP-complete) Modellen umfasst Modelle aus NP, die in polynomialer Zeit auf einander reduzierbar sind
- Bekanntestes Beispiel aus NPC: Satisfiability Problem (Cook 1970)
- Gilt P = NP oder P ≠ NP? Das ist eine US $ 1.000.000,- Preis-Frage!
- Aussagen angelehnt an L.A. Wolsey: Integer Programming (Literaturliste):
  - *Super-Pessimist:* Weil die meisten Probleme aus NP sind, ist ein Lösungsversuch sinnlos!
  - *Mathematiker (Optimist):* Möchte P=NP beweisen und dadurch reich u. berühmt werden
  - *Mathematiker (Pessimist):* Möchte P≠NP beweisen und dadurch reich u. berühmt werden
  - *Stochastiker:* Gibt es Algorithmen die *mit hoher Wahrscheinlichkeit* in *polynomialer Zeit* laufen und eine „*nahezu*“ optimale Lösung finden?
  - *Ingenieur:* Entwickelt einen *heuristischen Algorithmus*, der praktikable Ergebnisse liefert
  - *Mathematiker / Informatiker (Pragmatiker):* entwickelt *exakte Algorithmen* und Software, die prinzipiell jedes NP Problem lösen kann, viele in akzeptabler Zeit; wenn keine optimale Lösung gefunden wird, so gibt es aber eine *garantierte Qualitätsaussage*!
  - *Ihr (zukünftiger) Chef:* „Optimierungstheorie ist mir total egal. Geben Sie mir bis morgen 8 Uhr einen guten Produktionsplan für die nächste Woche!“
- Computers and Intractability: A Guide to the Theory of NP-Completeness, Garey/Johnson, Freeman

Beispiele

- **Leichte Probleme (aus P)**
  - Uncapacitated lot-sizing problem
  - Maximum weight tree problem
  - Shortest path problem
  - Max flow problem
  - Assignment problem
  - Sortieren



- **Schwere Probleme (kein polynomialer Algorithmus bekannt)**
  - 0/1 knapsack problem und seine Varianten (Geldwechselautomat)
    Max $c'x$, $w'x \leq b$, Min $c'x$, $w'x = b$
  - Set covering, partitioning, packing problem
    Min $c'x$, $A\,x \geq \underline{1}$, $A\,x = \underline{1}$, Max $c'x$, $A\,x \leq \underline{1}$, wobei A eine mxn 0-1-Matrix ist und $\underline{1} \in R^m$
  - Traveling salesman problem, Uncapazitated facility location problem
  - Steiner tree problem
- **Für einige Probleme gibt es *Heuristiken***
  - Heuristik: ein Verfahren / Algorithmus das manchmal aber nicht immer eine Lösung findet, für die man meistens keine *nichttriviale* Qualitätsaussage machen kann
  - Es gibt sehr unterschiedliche Klassen von Heuristiken
  - Die meisten Heuristiken arbeiten mit sequentiellen Entscheidungen die *nicht revidiert* werden: Beispiel Knapsack Problem, Greedy-Heuristik für Geldwechsel-Automat
- **Einige Probleme aus P bzw. NP werden wir später genauer besprechen**

Definition von (gemischt-) ganzzahligen Modellen

- Ein *ganzzahliges Programm* ist ein lineares Programm, bei dem *alle Variablen* nur ganzzahlige Werte annehmen dürfen
- Ein *gemischt-ganzzahliges Programm (MIP)* ist ein lineares Programm, bei dem *einige Variablen* nur ganzzahlige Werte annehmen dürfen; die übrigen Variablen dürfen kontinuierliche (reelle) Werte annehmen
- Ein *binäres Programm* oder *reines 0-1-Programm* ist ein lineares Programm, bei dem alle Variablen nur die Werte 0 und 1 annehmen dürfen
- Ein *gemischtes 0-1 Programm* ist ein lineares Programm, bei dem einige Variablen nur die Werte 0 und 1 annehmen dürfen; die übrigen Variablen dürfen kontinuierliche (reelle) Werte annehmen
- Wir sprechen im folgenden *für alle diese Modellklassen* von *Integer Programming Modellen*, kurz *IP-Modellen*
- Charakteristisch ist also bei *IP-Modellen* das Vorhandensein von *diskreten Variablen* mit einer Kombination aus 0-1 oder allgemeinen ganzz. Variablen
- die allgemeine Form (P) eines IP-Modells (externe Darstellung) lautet
  min $c'x$, $rl \leq A\,x \leq ru$, $l \leq x \leq u$, wobei $x, l, u, c \in R^n$, $rl, ru \in R^m$, A mxn Matrix; x_j muss ganzzahlig sein für $j \in JI \subseteq \{1,..,n\}$

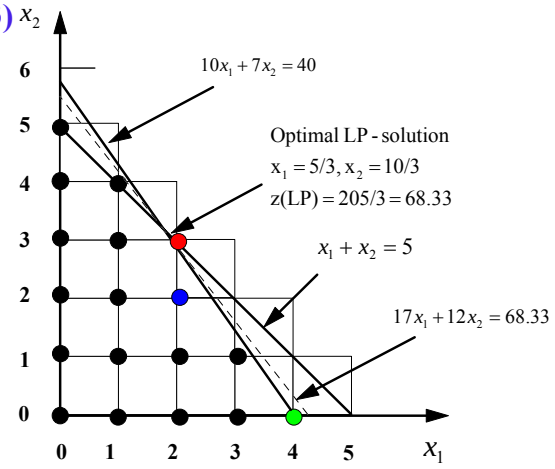
Beispiel eines Integer Optimierungsmodells

- maximiere $17x_1 + 12x_2$

  $10x_1 + 7x_2 \le 40,\ x_1 + x_2 \le 5$

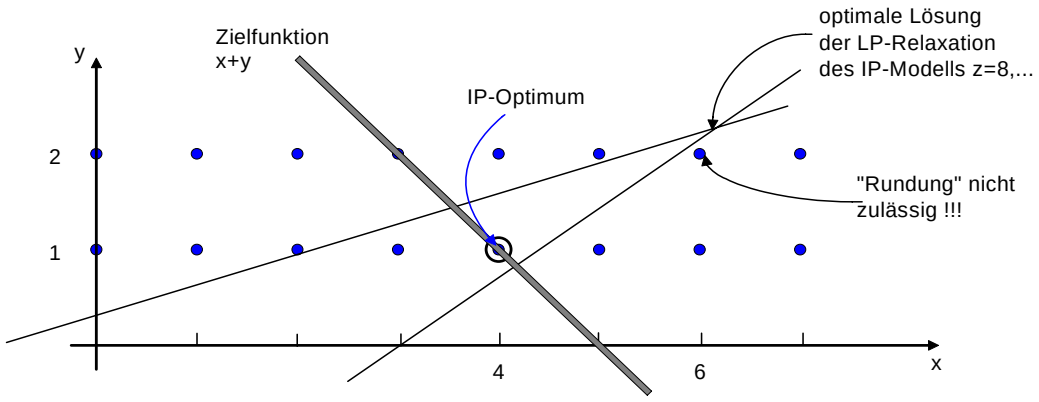
  $x_1, x_2 \ge 0$ und ganzzahlig

- **Die optimale LP-Lösung ist $x_1 = 1.66$, $x_2 = 3.33$ mit z(LP) = 68.33**
- **Die gerundete LP-Lösung (x_1=2, x_2=3) ergibt keine zulässige IP-Lösung!**
- **am nächsten zur LP-Lösung ist die IP-Lösung ($x_1 = 1$, $x_2 = 3$) bzw. ($x_1 = 2$, $x_2 = 2$); sie sind jedoch nicht optimal**
- **Eine optimale IP-Lösung ist durch x_1=4 und x_2=0 gegeben, wobei z(IP) = 68**



Schwierigkeiten beim Runden von LP-Lösungen

- **Der zulässige Bereich eines IP-Modells ist nicht konvex, nicht einmal zusammenhängend**
- **Daraus ergeben sich die besonderen Probleme beim Lösen solcher Modelle**
- **Auch wenn es eine optimale LP-Lösung der LP-Relaxierung gibt, so kann die Menge der zulässigen Lösungen des IP-Modells leer sein**
- **die Rundung einer LP-Lösung des IP-Modells ist i.A. weder zulässig noch optimal**



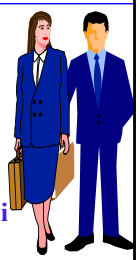
LP Relaxierung eines IP-Modells

- ***Hat eine zentrale Bedeutung für die Lösung und Modellierung von IP-Modellen***
- **Die *zugehörige LP-Relaxierung* (LP) zu einem IP-Modell (P) s.o. entsteht, wenn man die Ganzzahligkeits-Bedingungen im IP-Modell aufhebt, d.h. die Forderung x_j muss ganzzahlig sein für $j \in JI \subseteq \{1,..,n\}$ wird aus dem Modell entfernt**
- **Man erkennt unmittelbar folgende Aussagen**
  - **Für den Zielfunktionswert der LP-Relaxierung z(LP) gilt bei Minimierung bzw. Maximierung immer $z(LP) \leq z(IP)$ bzw. $z(LP) \geq z(IP)$, wobei z(IP) der Zielfunktionswert einer *zulässigen IP-Lösung* ist; insbesondere gilt dies füe eine optimale IP-Lösung**
  - **Der Zielfunktionswert der LP-Relaxierung ist also immer eine untere Schranke bei Minimierung bzw. eine obere Schranke bei Maximierung für den Zielfunktionswert einer IP-Lösung (warum?)**
  - **Wenn die LP-Relaxierung keine zulässige Lösung aufweist, dann hat auch das IP-Modell keine zulässige (IP-) Lösung (warum?)**
  - **Wenn die LP-Relaxierung eine optimale LP-Lösung aufweist, die die Ganzzahligkeits-Bedingungen erfüllt, dann ist diese Lösung auch eine optimale Lösung für das IP-Modell (warum?)**

Beispiel Personalbedarfsplanung

- **Bei einer Zeitarbeitsvermittlung ist die benötigte Anzahl von Mitarbeitern vom Wochentag abhängig:**

Mo	Di	Mi	Do	Fr	Sa	So
17	13	15	19	14	16	11



- **Betriebsvereinbarung:**
  - **jeder MA arbeitet 5 Tage hintereinander und hat danach 2 Tage frei**
  - **z.B.: wenn ein MA von Montag bis Freitag arbeitet, ist Samstag u. Sonntag frei**
- **es soll für jeden Tag die minimal benötigte Anzahl von Mitarbeitern bestimmt werden, um den Bedarf zu decken (für jede Woche)!**
- **es sei y_i die Anzahl der Mitarbeiter, die am Wochentag i ($1 \leq i \leq 7$) anfangen**
- **Modell:** $\min z = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7$, y_i ganzzahlig

$y_1 + y_4 + y_5 + y_6 + y_7 \geq 17$ (Mo), $y_1 + y_2 + y_5 + y_6 + y_7 \geq 13$ (Di)

$y_1 + y_2 + y_3 + y_6 + y_7 \geq 15$ (Mi), $y_1 + y_2 + y_3 + y_4 + y_7 \geq 19$ (Do)

$y_1 + y_2 + y_3 + y_4 + y_5 \geq 14$ (Fr), $y_2 + y_3 + y_4 + y_5 + y_6 \geq 16$ (Sa)

$y_3 + y_4 + y_5 + y_6 + y_7 \geq 11$ (So)

- **Optimale LP-Lösung: $z = 22\ 1/3$, $y_1 = 4/3$, $y_2 = 10/3$, $y_3 = 2$, $y_4 = 22/3$, $y_5 = 0$, $y_6 = 10/3$, $y_7 = 5$**
- **Aufrundung der LP-Lösung $\Rightarrow$ $y_1 = 2$, $y_2 = 4$, $y_3 = 2$, $y_4 = 8$, $y_5 = 0$, $y_6 = 4$, $y_7 = 5$, $z = 25$**
- **Diese Lösung ist jedoch nicht optimal !**
- **Eine optimale MIP-Lösung ist: $y_1 = 4$, $y_2 = 4$, $y_3 = 2$, $y_4 = 6$, $y_5 = 0$, $y_6 = 4$, $y_7 = 3$, $z = 23$; Alternativlösung: $y_1 = 7$, $y_2 = 2$, $y_3 = 2$, $y_4 = 8$, $y_5 = 0$, $y_6 = 4$, $y_7 = 0$; Hier werden zwei Mitarbeiter weniger eingestellt!**

Beispiel Spanplattenproduktion

- Hornitex produziert Spanplatten einer Größe in sechs verschiedenen Stärken
- Für jede produzierte Plattenart fallen Fixkosten in Höhe von € 1000 an
- für bestimmte Plattenarten sind minimale Losgrößen vorgegeben
- mit Ausnahme der Plattenart 5 und 6 können die anderen Plattenarten auch durch Plattenarten mit größerer Stärke ersetzt werden

Plattenart	1	2	3	4	5	6
Stärke (mm)	16	20	22	24	28	30
Bedarf (Stück)	6000	5000	4000	4000	3000	1000
Min. Losgröße	-	-	-	400	200	350
Variable Kosten [€/Stück]	1,6	2,0	2,2	2,4	2,8	3,0

- x_i ist die zu produzierende Menge von Plattenart i und y_i die entsprechende 0-1-Variable
- Die Produktionsmenge x_i wird als kontinuierlich definiert (approximiert!)

$$\min \sum_{i=1}^{4} 1000\, y_i + 1{,}6\, x_1 + 2{,}0\, x_2 + 2{,}2\, x_3 + 2{,}4\, x_4 + 2{,}8\, x_5 + 3{,}0\, x_6 + 2000$$

$$x_6 \geq 1000, x_5 \geq 3000$$

$$x_4 + x_5 + x_6 \geq 8000, x_3 + x_4 + x_5 + x_6 \geq 12000, x_2 + x_3 + x_4 + x_5 + x_6 \geq 17000$$

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 23000$$

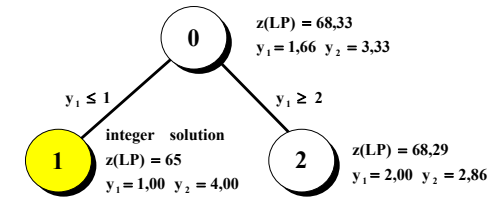
$$x_4 \geq 400 y_4, x_i \leq M y_i, i = 1,...,4; M = 23000 \text{ (dieser Wert kann verbessert werden!)}$$

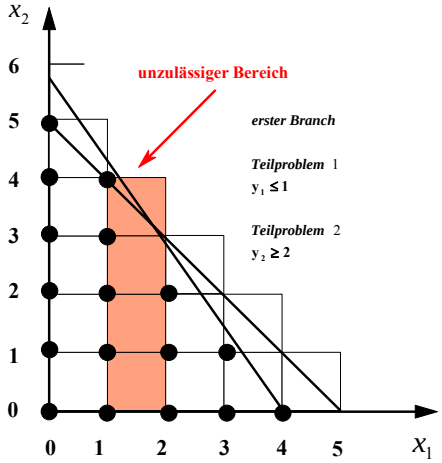
Lösungsverfahren für IP-Modelle

- Da IPs im allgemeinen aus *NP-vollständig* sind, gibt es also vermutlich kein effizientes Lösungsverfahren (polynomiale Laufzeit)
- Das einzige konkurrenzfähige Lösungsverfahren für allgemeine IP-Modelle ist bis heute *LP-basiertes Branch-and-Bound* mit folgenden Grundprinzipien
  - Es werden endlich viele *disjunkte Teilprobleme* durch Aufteilung der Lösungsmenge (Verzweigen, *branching*) gebildet, die systematisch partiell enumeriert werden
  - Eine optimale IP-Lösung ergibt sich aus der LP-Lösung eines der (vielen) Teilprobleme
  - *Entscheidend für die Effizienz des Verfahrens ist das Lösen der LP-Relaxierung eines Teilproblems*
  - Ein Teilproblem ist untersucht wenn (Eigenschaft der LP-Relaxierung)
    - Die LP-Relaxierung keine zulässige Lösung aufweist (infeasible)
    - Die LP-Relaxierung eine ganzzahlige Lösung aufweist (integer)
    - Der Zielfunktionswert z(LP) der LP-Relaxierung größer ist, als der Zielfunktionswert einer bereits gefundenen (gemischt-) ganzzahligen Lösung bzw. einer vorgegebenen Schranke
    - z(LP) ist (bei Minimierung) eine untere Schranke für den ZF-Wert einer zulässigen Lösung des (gemischt-) *ganzzahligen Teilproblems* (*bound*)
    - *Das Minimum der ZF-Werte aller LP-Teilprobleme ist eine untere Schranke für eine IP-Lös.*
  - Die *Schärfe einer LP-Relaxierung* wird u.a. durch den relativen Abstand der ZF-Werte zwischen einer optimalen IP-Lösung und des Anfangs-LP gemessen
  - Sie wird durch die Modellierung des Anwenders und durch eine Reihe von Techniken im IP-Preprocessing (super node processing) entscheidend beeinflusst
  - Wichtig können auch (problemabhängige) Heuristiken zur Bestimmung einer guten IP-Anfangslösung sein

Lösungsmethodik Branch & Bound am Beispiel

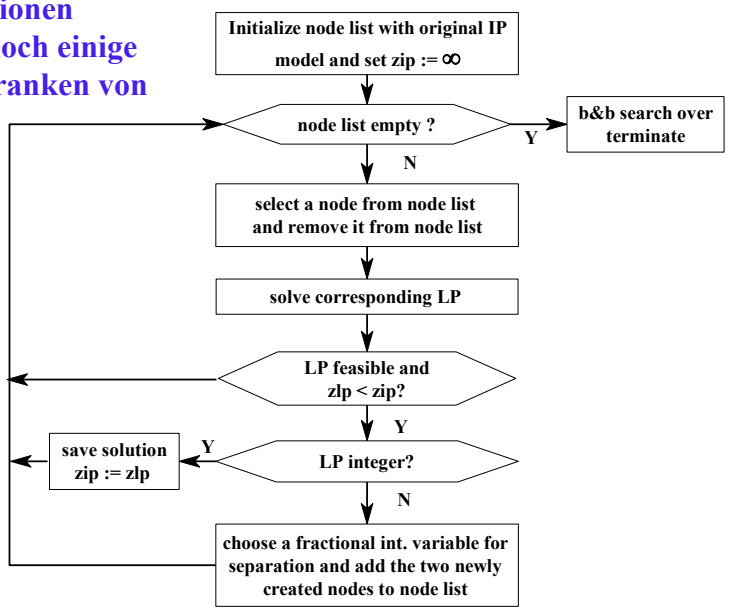
- an jedem Knoten wird ein LP gelöst
- gibt es in der LP-Lösung fraktionelle Integer-Variablen, so wird unter diesen eine Variable j als *Branching-Variable* gewählt *(Separierung)*
- ist $y_j = a$ (fraktionell), so werden zwei neue Teilprobleme definiert mit: 1. $lb_j \le y_j \le \lfloor a \rfloor$ und 2. $\lceil a \rceil \le y_j \le ub_j$
- ein Teilproblem ist untersucht, wenn es
  - unzulässig, ganzzahlig oder
  - einen schlechteren LP-ZF-Wert hat, als der ZF-Wert der besten gefundenen IP-Lösung
- eine ganzzahlige Lösung mit besserem ZF-Wert wird gespeichert
- Dann wird unter den aktiven Teilproblemen eines ausgewählt (Knotenauswahl-Strategie) und mit diesem Teilproblem fortgefahren
- das Verfahren ist beendet, wenn es keine aktiven Teilprobleme mehr gibt





Prinzip des Branch-and-Bound-Algorithmus (min)

- Node ist ein Teilproblem, das alle Modellrestriktionen beinhaltet, bei dem jedoch einige untere bzw. obere Schranken von Integer-Variablen eingeschränkt sind
- Separation beinhaltet die Auswahl einer Branching-Variablen die zur Erzeugung zweier neuer Teil-probleme verwendet wird

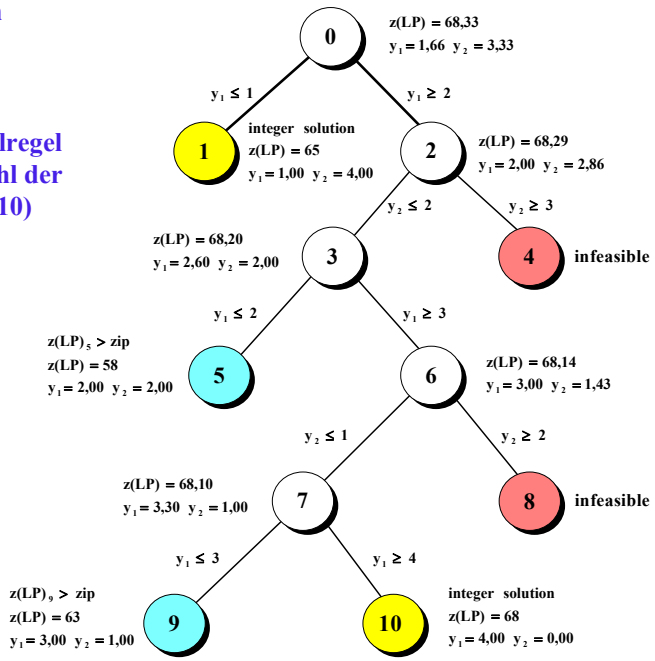


zip: function value of best integer solution zlp: function value of current LP relaxation

Lösung des IP-Beispiels mit Branch-and-Bound

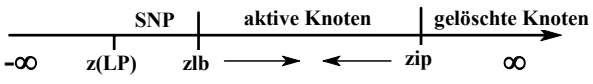
- die Knotennummer gibt an in welcher Reihenfolge die Teilprobleme ausgewählt wurden
- Mit einer anderen Auswahlregel ändert sich i.d.R. die Anzahl der entwickelten Knoten (hier 10)
- Teilprobleme die
  - unzulässig (Knoten 4, 9)
  - ganzzahlig (Knoten 1, 10)
  - LP-ZF-Wert > zip

  aufweisen (Knoten 5, 9)
  haben keine Nachfolger
- *MOPS löst dieses Problem ohne Branch-and-Bound bereits im IP-Preprocessing durch einen Gomory-Cut*



Eigenschaften des B&B-Verfahrens (Min)

- Zwei wesentliche Auswahlprozesse
  - Welches Teilproblem (Knoten) soll aus der Kandidatenliste gewählt werden ?
  - Welche fraktionelle Integer-Variable soll zur Separierung gewählt werden ?
  - Die verwendeten Auswahlregeln haben einen großen Einfluss auf das Leistungsverhalten (Anzahl entwickelter Knoten)
- Während des B&B-Verfahrens sind zwei Schranken von Bedeutung
  - zip ist der ZF-Wert der besten gefundenen IP-Lösung und bildet eine *obere Schranke*
  - Mit jeder besseren IP-Lösung wird zip reduziert
  - Ist der Zielfunktionswert der LP-Lösung eines Teilproblems ≥ zip, so ist das Teilproblem *untersucht* und wird aus der Liste der aktiven Knoten entfernt, d.h. gelöscht
  - zlb ist eine *untere Schranke* für den best möglichen ZF-Wert einer optimalen Integer Lösung
  - Zu Beginn des Verfahrens ist zlb = z(LP), der ZF-Wert der LP-Relaxierung des Modells; im 1. supernode-processing (snp) wird die LP-Relaxierung verschärft
  - Da jedes Teilproblem durch Einschränkung der Schranken von fraktionellen IP-Variablen definiert wird, wächst zlb monoton, jedoch i.d.R. nicht streng monoton
  - Der relative Abstand |zip-zlb|/(1+|zlb|) zwischen zlb und zip beinhaltet eine Schranke wie weit eine IP-Lösung maximal vom IP-Optimum entfernt ist; die Optimierung könnte vorzeitig beendet werden, wenn dieser Wert hinreichend klein ist, z.B. 0.05 (5%)

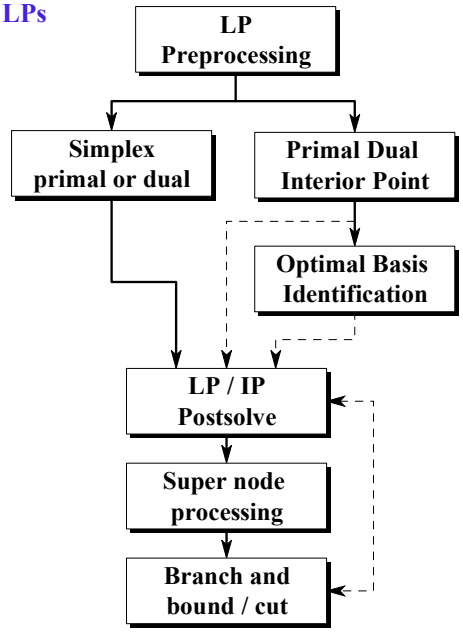


Möglichkeiten zur Begrenzung der B&B-Suche

- Alle Softwaresysteme zur Lösung von IP-Modellen bieten die Möglichkeit über den ***absoluten*** und ***relativen Gap*** die Suche zu begrenzen, wobei der Gap nur dann definiert ist, wenn zip endlich ist, d.h. wenn eine ganzzahlige Lösung gefunden wurde oder zip extern vorgegeben wird
  - Absoluter Gap: |zlb – zip|
  - Relativer Gap: |zlb – zip| / (1 + |zip|), wobei statt der 1 auch ein ε gewählt werden kann
  - In MOPS wird das Attribut xabgap (Default: 0) benutzt, d.h. ist xabgap > 0 so wird die B&B-Suche vorzeitig beendet wenn |zlb – zip| ≤ xabgap ist
  - das Attribut xglgap wird verwendet um einen relativen Gap zu definieren; ist |zlb–zip|/(1+|zip|)≤xglgap, so wird die B&B-Suche vorzeitig beendet
  - xglgap hat im Default den Wert 0.0001; wenn das Attribut xnogap = 0 ist, dann wird xglgap nach dem Supernode Processing nach der Formel min(xglgap,1/(1+|zlb|)) berechnet, um bei großen ZF-Werten nicht eine optimale IP-Lösung abzuschneiden
  - Ist einer der Werte xbagap oder xglgap > 0, so wird die Suche vorzeitig beendet und man hat ***bewiesen***, dass keine Integer-Lösung besser sein kann, als die vorgegebenen absoluten oder relativen Schranken (***garantierte Qualitätsaussage!***)
  - Weiterhin kann man über das Attribut xrimpr (Default: 0.0001) die relative Verbesserung einer neuen Integer-Lösung vorgeben; wenn xrimpr > 0 so wird dadurch auch der globale Gap entsprechend vergrößert.
  - Über die maximale Anzahl der Knoten, CP-Zeit , LP-Iterationen, … läßt sich die Suche ebenfalls limitieren und die Suche endet mit der Berechnung des globalen Gaps.

LP / IP-Optimization mit MOPS

- Normalfall der LP/IP-Optimierung:
  - LP-Preprocessing zur Entfernung trivialer Modellteile
  - IPM mit X-over oder Simplex zur Lösung des LPs
  - Postsolve
  - Supernode Processing zur Verschärfung der LP-Relaxierung des IP-Modells
    - Bestimmung aller Cliques und Speicherung
    - Logische Tests und Probing
    - Speicherung aller Implikationen
    - Bound und Koeffizienten Reduktion
    - Identifikation redundanter Restriktionen
    - Ableitung von Clique Cuts
    - Ableitung Implication Cuts
    - Ableitung von Cover Cuts
    - Ableitung von MIR Cuts
    - Ableitung von Gomory Cuts
  - MIP-Heuristik
  - Branch-and-Bound mit Simplex zur Reoptimierung der Teilprobleme



Lösbare Problemgrößen für IP-Modelle

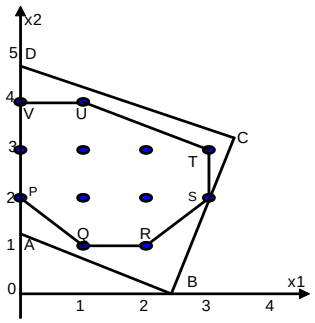
- **Für IP-Modelle lassen sich kaum allgemeine Aussagen über die Schwierigkeit in Abhängigkeit von der Problemgröße treffen**
- **Je mehr ganzzahlige Variablen desto schwieriger**
- **Binäre Variablen sind besser als allgemeine ganzzahlige Variablen**
- **Viele IP-Modelle aus der Praxis mit einigen hundert bis wenigen tausend Binärvariablen können bei guter Modellierung optimal gelöst werden**
- **Es gibt aber auch sehr kleine IPs mit weniger als hundert Variablen, die nicht gelöst werden können, und sehr große IPs / MIPs mit mehreren Millionen Binärvariablen, die optimal gelöst werden können.**
- **Die Schwierigkeit eines IP-Modells hängt stark von der Problemklasse und der Modellformulierung d.h. der Schärfe der LP-Relaxation ab**
- **Je größer der relative Abstand zwischen dem optimalen Zielfunktionswert der LP-Relaxation des Ausgangs-IPs und einer optimalen Integer-Lösung ist (*Duality-Gap*), desto schwieriger ist das IP-Modell zu lösen**
- **IP-Modelle wo dieser *Duality-Gap* unter 5% liegt lassen sich in der Regel relativ schnell, d.h. durch Untersuchen einiger tausend Teilprobleme (Knoten) lösen**

Zwei extreme Beispiele

- **Ein Rucksackproblem*) mit 6 allgemeinen Integer-Variablen:**
  Max 213 x1 − 1928 x2 − 11111 x3 − 2345 x4 + 9123 x5 − 12834 x6
  s.t. 12228 x1 + 36679x2 + 36682x3 +48908x4 + 61139x5 +73365x6 = 89716839
  x ≥ 0, ganzzahlig
- *) Aardal, K. and Lenstra, A.K. Hard equality constrained integer knapsacks. Preliminary version, in W.J. Cook and A.S. Schulz (eds.), Integer Programming and Combinatorial Optimization: 9th International IPCO Conference, Lecture Notes in Computer Science vol. 2337, Springer-Verlag, 2002, 350 - 366.
- Benötigt auch mit den besten Systemen Stunden Rechenzeiten
- Anderes Beispiel nort3 mit vielen tausend Variablen und Restriktionen und großem „duality gap“ wir im Supernode Processing optimal gelöst (⇒ Demo)
- Details des Supernode Processing:
  - Optimaler Zielfunktionswert ds LPs: 36.5
  - Inaktivierte Restriktionen: 3074
  - Neu generierte Cuts (Restriktionen): 3931
  - Neu eingefügte Matrixkoeffizienten: 3931
  - Modifizierte Matrixkoeffizienten: 764
  - Zielfunktionswert nach dem SNP: 63.0 (Lösung ist ganzzahlig und optimal)

Wann ist eine optimale LP-Lösung ganzzahlig?

- Bei *reinen IP-Modellen* könnte man theoretisch die konvexe Hülle aller zulässigen ganzzahligen Punkte bestimmen und über diese Restriktionen ein LP lösen, das dann ganzzahlig ist
- Leider ist die Bestimmung der konvexen Hülle auch NP-hart!
- Man kann aber bei vielen Modellen durch Ableitung von Schnittebenen das ursprüngliche Polytop verkleinern und das IP-Modell dann leichter lösen
- Beispiel: egout.mps



- Es gibt IP-Modelle bei denen die optimale LP-Lösung *immer* ganzzahlig ist
- Wir betrachten dazu *reine IP-Modelle*, bei denen alle Koeffizienten ganzzahlig sind und die Koeffizientenmatrix total unimodular ist:
  - Eine mxn Matrix ist total unimodular (TU), wenn jede Determinante einer quadratischen Teilmatrix von A den Wert 0, 1 oder -1 hat
- Theorem: Es sei A eine mxn Matrix mit Rang m. Folgende Aussagen sind äquivalent:
  - Jede LP-*Basislöusng* von A x = b, x ≥ 0 ist ganzzahlig, wenn b ganzzahlig ist
  - A ist TU
- Es gibt wichtige Modellklassen z.B. kostenminimale Netzwerklußprobleme die diese Eigenschaften haben und bei denen daher die LP-Lösung automatisch ganzzahlig ist

Einige wichtige IP-Modellklassen

- Set Packing, Set Partitioning und Set Covering Probleme
  - Es sei A eine mxn Matrix die nur aus 0 und 1 besteht, $c \in R^n_+$, $\underline{1} \in \{1\}^n$, $x \in \{0,1\}^n$
  - Set Partition Problem: Min c'x, A x = $\underline{1}$
  - Set Covering Problem: Min c'x, A x ≥ $\underline{1}$
  - Set Packing Problem: Max c'x, A x ≤ $\underline{1}$
- Einfache Knapssack Probleme
  - Es sei w, $c \in R^n_+$, $b \in R_+$ möglich ist auch w,$c \in N^n$, $b \in N$
  - Max c'x, s.t. w'x ≤b wobei $x \in \{0,1\}^n$ ist ein binäres Knapssack
  - Max c'x, s.t. w'x ≤b wobei x≥0, ganzzahlig ist ein ganzzahliges Knapssack
  - Min c'x, s.t. w'x = b wobei x≥0, ganzzahlig ist ein „Change Making Problem"
- Weitere Varianten
  - Multiple Choice Knapsack Probleme (auch Generalized Upper Bound) genannt: Hierbei werden die 0-1 Variablen in k disjunkte Teilmengen eingeteilt; aus jeder Teilmenge muss genau eine Variable gewählt werden; die übrigen Eigenschaften bleiben
  - Generalized Assignment Problem:

$$\text{Max} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_{ij}x_{ij} \le b_i, i=1,..,m; \quad \sum_{i=1}^{m} x_{ij} = 1, j=1,..,n$$

$$x_{ij} \in \{0,1\} j=1,..,m; j=1,..,n$$