Kapitel 4

Mathematische Optimierungsmodelle

Einführung in AMPL (1)

Uwe H. Suhl
Lehrstuhl für Wirtschaftsinformatik
Freie Universität Berlin

Optimierungssysteme
Version 1.1 / SS 2008

Modellierungssysteme

- Sind Softwaresysteme zur Unterstützung der Implementierung von mathematischen Optimierungsmodellen; sie weisen mindestens folgende Systemkomponenten auf:
  - Einer algebraischen Modellierungssprache zur Beschreibung des Modells
  - Einem Editor zur Eingabe eines Modells in dieser Sprache (Modell und Daten getrennt)
  - Schnittstellen zu Optimierungssystemen
  - Je nach System kann das Optimierungssystem integriert sein
- Wir benutzen eine β-Version von MOPS Studio, das auf AMPL und MOPS basiert
- Eine Übersicht über bekannte Systeme findet man in folgender Tabelle

Name		URL	Solver	State
AIMMS	Advanced Integrated Multi-dimensional Modeling Software	www.aimms.com	open	commercial
AMPL	A Modeling Language for Mathematical Programming	www.ampl.com	open	commercial
GAMS	General Algebraic Modeling System	www.gams.com	open	commercial
LINGO	Lingo	www.lindo.com	fixed	commercial
LPL	(Linear|Logical|Literate) Programming Language	www.virtual-optima.com	open	commercial
MINOPT	Mixed Integer Non-linear Optimizer	titan.princeton.edu/MINOPT	open	mixed
MOSEL	Mosel	www.dashoptimization.com	fixed	commercial
MPL	Mathematical Programming Language	www.maximalsoftware.com	open	commercial
OMNI	Omni	www.haverly.com	open	commercial
OPL	Optimization Programming Language	www.ilog.com	fixed	commercial
GNU-MP	GNU Mathematical Programming Language	www.gnu.org/software/glpk	fixed	free
ZIMPL	Zuse Institute Mathematical Programming Language	www.zib.de/koch/zimpl	open	free

T. Koch, Rapid Mathematical Programming, TU Berlin, 2004, ZIB-Report 04-58, http://www.zib.de/Publications/abstracts/ZR-04-58/

Beispiel: Produktionsplanung

- In einer Stunde können 200 Tonnen bands oder 140 Tonnen coils produziert werden.
- Der Deckungsbeitrag beträgt 25 € für eine Tonne bands und 30 € für eine Tonne coils.
- Es können wöchentlich max. 6000 Tonnen bands und 4000 Tonnen coils verkauft werden.
- Es stehen 40 Stunden in der Woche zur Produktion zur Verfügung (einperiodisches Modell).
- Wir zeigen das mathematische Modell und die Umsetzung in AMPL
- Entscheidungsvariablen sind
  - xB: Menge der zu produzierenden bands
  - xC: Menge der zu produzierenden coils
  - Das Modell soll maximiert werden

LP-Modell

Maximiere 25 xB + 30 xC

$1/200\ xB + 1/140\ xC \leq 40$

$0 \leq xB \leq 6000$

$0 \leq xC \leq 4000$

LP-Modell in AMPL

```
var XB;
var XC;
maximize profit: 25 * XB + 30 * XC;
subject to Time: (1/200) * XB +
                 (1/140) * XC <= 40;
subject to B_limit: 0 <= XB <= 6000;
subject to C_limit: 0 <= XC <= 4000;
```

- In diesem sehr einfachen Beispiel sind die Daten (Parameter) explizit im AMPL-Programm definiert; später werden Modell und Daten in getrennten Dateien spezifiziert

Grundelemente von AMPL (1)

- Mengen: *set*
- Variablen: *var*
- Parameter: *param*
- Zielfunktion: *maximize / minimize*
- Restriktionen: *subject to*
- *Weist man den Parametern eines Modells feste Daten zu, so erhält man eine Modellinstanz*
- Einfache (nicht indizierte) Mengen in AMPL:

```
set PROD;                                  (Modelldatei)
set PROD := bands, coils, plate;           (Datendatei)
```

```
set numbers;                               (Modelldatei)
set numbers := 2.1, -1, 0.4E+2;            (Datendatei)
```

```
set YEARS;                                 (Modelldatei)
set YEARS := 1990 .. 2020 by 5;            (Datendatei)
```

```
set YEARS;
param start integer;
param end > start integer;                 (Modelldatei)
param interval > 0 integer;
param start := 1990;
param end := 2020;                         (Datendatei)
param interval := 5;
set YEARS := start .. end by interval;
```

Das Steel-Modell wird auf beliebig viele Produkte erweitert

- P Menge der Produkte
- a_j Anzahl Tonnen von Produkt j, die in einer Stunde produziert werden kann. $j \in P$
- b die zur Verfügung stehende Anzahl von Produktionsstunden
- c_j Deckungsbeitrag des Produktes j, $j \in P$
- u_j Obergrenze der Produktion von Produkt j, $j \in P$
- Entscheidungsvariablen: x_j Produktionsmenge von Produkt j in Tonnen, $j \in P$
- LP-Modell

$$\text{Maximiere } \sum_{j \in P} c_j x_j \text{ subject to } \sum_{j \in P} 1/a_j x_j \le b, \text{ wobei } 0 \le x_j \le u_j, j \in P$$

AMPL Modell (prod.mod)

```
set P;
param a {j in P};
param b;
param c {j in P};
param u {j in P};
var X {j in P};
maximize profit: sum {j in P} c[j] * X[j];
subject to Time: sum {j in P} (1/a[j]) * X[j] <= b;
subject to Limit {j in P}: 0 <= X[j] <= u[j];
```

AMPL Daten (prod.dat)

```
set P := bands coils;
param:   a    c     u :=
bands   200   25   6000
coils   140   30   4000 ;
param b := 40;
```

Dokumentation von AMPL Programmen

- Da AMPL eine (semantisch mächtige) Programmiersprache ist, gelten ähnliche Grundsätze für eine gute Dokumentation des Modells
- Wählen Sie aussagekräftige Namen für Parameter, Variablen und Restriktionen
- Dokumentieren Sie Ihr Programm mit Kommentaren
- Aus Kompatibilitätsgründen sollten keine Umlaute oder Sonderzeichen für Variablen und Kommentare verwendet werden!

Modell (steel.mod)

```
set PROD;                  # products
param rate {PROD} > 0;     # tons produced per hour
param avail >= 0;          # hours available in week
param profit {PROD};       # profit per ton
param market {PROD} >= 0; # limit on tons sold in week
var Make {p in PROD} >= 0, <= market[p]; # tons produced
maximize total_profit: sum {p in PROD} profit[p] * Make[p];
# Objective: total profits from all products
subject to Time: sum {p in PROD} (1/rate[p]) * Make[p] <= avail;
/* Constraint: total of hours used by all
   products may not exceed hours available */
```

Produktionsbeispiel mit zusätzlichem Produkt

- Um ein Produkt hinzuzufügen, braucht das AMPL Modell ***nicht*** geändert werden.
- Es reicht eine Erweiterung der Datendatei

steel2.dat

```
set PROD := bands coils plate;
param: rate profit market :=
bands   200   25    6000
coils   140   30    4000
plate   160   29    3500;
param avail := 40;
```

- Aufgaben
  - Installieren Sie MOPS Studio auf Ihrem Rechner
  - Ggf. müssen Sie den .Net Framework 2.0 herunterladen und installieren
  - Geben Sie das Steel-Beispiel ein und lösen Sie das LP
  - Erstellen Sie die Steel2.dat und lösen Sie das LP; wie lautet eine optimale Lösung?
  - Lesen Sie das Dokument AMPL und MOPS Studio

Parameter in AMPL

- Parameter können als *binary*, *integer* oder *symbolic* deklariert werden
- Skalare Parameter:
  - **param T;** (Modelldatei)
  - **param T := 12;** (Datendatei)
- Einfach indizierte Parameter
  - **set PROD;** (Modelldatei)
  - **param rate {PROD} > 0;**
  - **set PROD := bands coils plate;** (Datendatei)
  - **param rate := bands 200 coils 140 plate 160;**
- Mehrfach indizierte Parameter
  - **set ORIG;** (Modelldatei)
  - **set DEST;**
  - **param COST {ORIG, DEST} >= 0;**
  - **param cost: FRA DET LAN :=** (Datendatei)
    GARY 39 40 25
    CLEV 14 9 35
    PITT 34 50 12;

Variablen in AMPL

- Kontinuierliche Variablen mit oberen und unteren Schranken

```
set FOOD
param f_min {FOOD} >= 0;
param f_max{j in FOOD} >= f_min[j];
var Buy {j in FOOD} >= f_min[j], <= f_max{j};
```

- Fixierte Variablen

```
var minBuy {j in FOOD} = f_min{j};
```

- Binäre Variablen

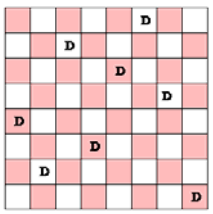
```
var shouldBuy binary;
```

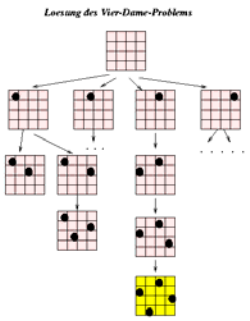
- Ganzzahlige Variablen

```
var intBuy {j in FOOD} integer >= f_min[j], <= f_max[j];
```

Eight Queens Problem

- Ziel: Acht „Damen“ sollen so auf ein Schachbrett gestellt werden, dass keine Dame von einer anderen Dame „geschlagen" werden kann
- Für dieses Problem gibt es 92 zulässige Lösungen
- Eine Lösung zeigt die nebenstehende Abbildung
- Das Problem kann für ein beliebiges n (statt n = 8) formuliert werden
- Die Lösungen können z.B. über „Backtracking“ Algorithmen gefunden werden
- Eine andere Möglichkeit eine zulässige Lösung zu finden ist das Problem als ein Integer Optimierungsproblem zu formulieren
- Wir definieren die 0-1-Variablen $x_{ij} = 1$, wenn eine Dame in Zeile i und Spalte j steht (i,j∈{1,..8}, 0 sonst
- Die Zielfunktion maximiert die Anzahl platzierter Damen
- Die Restriktionen erlauben in jeder Zeile und in jeder Spalte höchstens einen Eintrag $x_{ij} = 1$
- In *jeder* Haupt- und Nebendiagonale darf höchstens ein Eintrag $x_{ij} = 1$ sein





IP-Modell für Eight Queeens Problem

$x \in \{0,1\}^{64}$

Maximiere $\sum_{i=1}^{8}\sum_{j=1}^{8} x_{ij}$

$\sum_{i=1}^{8} x_{ij} \le 1, \quad j \in \{1,..,8\}$! in jeder Spalte höchstens ein Eintrag

$\sum_{j=1}^{8} x_{ij} \le 1, \quad i \in \{1,..,8\}$! in jeder Zeile höchstens ein Eintrag

$\sum_{j-i=k-7} x_{ij} \le 1, \quad k \in \{1,..,13\}$! max. Einträge in Hauptdiagonalen

$\sum_{j+i=k+2} x_{ij} \le 1, \quad k \in \{1,..,13\}$! max. Einträge in Nebendiagonalen

Indizierungen in AMPL

- {A} # über alle Elemente der Menge A
- {A, B} # über alle Paare aus A x B
- {i in A, j in B}, {i in A,B} # ebenfalls
- {i in A, C[i]} # über alle Paare aus A x C[i]
- {i in A, (j,k) in D} # über alle Tripel A x D, wobei D zweiwertig
- {i in A: p[i] > 0} # über alle Elemente von A für die p[i] > 0
- {i in A, j in C[i]: i <= j} # über alle Paare (i,j) aus A x C[i], für die i <= j
  # (A und C[i] dürfen nur numerische Elemente enthalten)
- {i in A, (i,j) in D: i <= j} # das Gleiche für eine zweiwertige Menge D
- Definition der Indexmengen für das 8-Queens-Problem
  - set Felder := {1..8, 1..8}; # 64 Felder des Schachbretts
  - set Zeile {1..8} within Felder; # Felder der Zeilen 1 bis 8
  - set Spalte{1..8} within Felder; # Felder der Spalten 1 bis 8
  - set HDiag {1..13} within Felder;# Felder der Hauptdiagonalen 1 bis 13
  - set NDiag {1..13} within Felder;# Felder der Nebendiagonalen 1 bis 13
  - let {i in 1..8} Zeile[i] := {(r,c) in Felder : r = i} # Zeile i
  - let {j in 1..8} Spalte[j] := {(r,c) in Felder : c = j}; # Spalte j
  - # NDiag von rechts oben nach links unten : r + c = const
  - let { k in 1..13} NDiag[k] := {(r,c) in Felder : r+c=k+2}; # NebDiag k
  - # HDiag von links oben nach rechts unten: c-r = const
  - let { k in 1..13} HDiag[k] := {(r,c) in Felder: c-r=k-7}; # HptDiag k

Eight Queeens Problem mit AMPL

```
set Felder := {1..8, 1..8};                                   # 64 Felder des Schachbretts
set Zeile {1..8} within Felder;                               # Felder der Zeilen 1 bis 8
set Spalte{1..8} within Felder;                               # Felder der Spalten 1 bis 8
set HDiag {1..13} within Felder;                              # Felder der Hauptdiagonalen 1 bis 13
set NDiag {1..13} within Felder;                              # Felder der Nebendiagonalen 1 bis 13
 let {i in 1..8} Zeile[i] :=  {(r,c) in Felder : r=i};        # Zeile i
let {j in 1..8} Spalte[j] :=  {(r,c) in Felder : c=j};        # Spalte j
# NDiag von rechts oben nach links unten  : r+c=const
let { k in 1..13} NDiag[k] := {(r,c) in Felder : c+r=k+2}; # NebDiag k
# HDiag von links oben nach rechts unten: c-r=const
let { k in 1..13} HDiag[k] := {(r,c) in Felder: c-r=k-7};   # HptDiag k
 param act;                                                    # aktuelle Loesung
let act:=0;
set Soln {0..act} within Felder;                              # Position der Damen in den Loesungen 0 bis act
let Soln[0] := {};                                            # Anfangsloesung ist leer
var x {Felder} binary;
 maximize Ziel: sum {(i,j) in Felder} x[i,j];
 subject to DiagN {k in 1..13}: sum{(r,s) in NDiag[k]} x[r,s] <= 1;
 subject to DiagH {k in 1..13}: sum{(r,s) in HDiag[k]} x[r,s] <= 1;
 subject to Row {i in 1..8}: sum{(r,s) in Zeile[i]} x[r,s] <= 1;
 subject to Col {i in 1..8}: sum{(r,s) in Spalte[i]} x[r,s] <= 1;
 subject to Old {k in 1..act}: sum{(r,s) in Soln[k]} x[r,s] <= 7.1 ; # Verbot alter Lösungen
 repeat
  {
 solve ;
       # solve_result_num = 200 ist der MOPSAMPL Returncode für infeasible
       if Ziel <= 7.5 or solve_result_num = 200 then break;
       let act := act + 1;
       let Soln[act] := {(i,j) in Felder : x[i,j] >= 0.9 };
 };
display Soln;
```