AMPL: Eine Kurzbeschreibung

AMPL (**A** **M**athematical **P**rogramming **L**anguage) ist eine algebraische Modellierungssprache für mathematische Optimierungsprobleme. AMPL bietet die Möglichkeit, Modelle für mathematische Optimierungsprobleme in abstrakter Form aufzustellen und diese dann durch Aufruf externer Solver zu lösen.

Einige Hinweise

Die Student Edition von AMPL Plus (Version 1.6) ist frei erhältlich und kann aus dem WWW unter

`http://www.ampl.com/DOWNLOADS/cplex71.html`

ohne Einschränkungen heruntergeladen werden.
Die auf dieser Seite downloadbare zip-Datei enthält auch (die Studentenversion für) den Solver CPLEX (AMPL braucht ja zusätzlich einen externen Solver (zB CPLEX für lineare und ganzzahlige lineare Programme und MINOS für lineare und nichtlineare Programme)). Die Studentenversion dieser Solver unterscheidet sich von der Vollversion dadurch, dass nur Probleme mit maximal 300 Variablen und maximal 300 Nebenbedingungen gelöst werden können.
Auf oben angeführter Website befinden sich auch Links zu einigen Kurzbeschreibungen von AMPL. Eine detaillierte Beschreibung von AMPL findet man im Buch

> *AMPL: A Modelling Language For Mathematical Programming*, R. Fourer, D. H. Gay, B. W. Kernighan, **1993**, Boyd & Fraser Publishing Company, Massachusetts.

Beispiele

Beispiel 1: Direkte Verwendung (unüblich)

Kommandofolge (interaktiv):

```
> ampl
ampl: var X1;
ampl: var X2;
ampl: minimize ZF: -4 * X1 + 3 * X2;
ampl: subject to NB: -2 * X1 - X2 = 5;
ampl: subject to X1_Limit: 0 <= X1 <= 10;
ampl: subject to X2_Limit: X2 <= 8;

ampl: solve;
CPLEX 7.1.0: optimal solution; objective -115
0 simplex iterations (0 in phase I)

ampl: display X1, X2;
X1 = 10
X2 = -25

ampl: quit;
>
```

Beispiel 1: Aufspaltung in 3 Dateien (üblich)

Man speichert üblicherweise in einer Datei das Modell und in einer zweiten die entsprechenden Daten.

Inhalt der Modell-Datei `direkt.mod`:

```
set INDEX;    # Menge der Indizes

param c {INDEX};    # Zielfunktionskoeffizienten
param a {INDEX};    # Restriktionskoeffizienten
param b;            # "rechte Seite"
param l {INDEX};    # untere Schranken
param u {INDEX};    # obere Schranken

var X {INDEX};    # Vektor der (gesuchten) Variablen
minimize ZF: sum {i in INDEX} c[i] * X[i];    # Zielfunktion
subject to NB: sum {i in INDEX} a[i] * X[i] = b;    # Restriktion
subject to Schranken {i in INDEX}: l[i] <= X[i] <= u[i];
# Schranken der Variablen
```

Inhalt der Daten-Datei `direkt.dat`:

```
set INDEX :=   1   2;

param:  c     a      l        u :=
 1     -4    -2      0        10
 2      3    -1   -Infinity   8;

param b := 5;
```

(Bei n Variablen schreibt man beispielsweise `set INDEX := 1..n;`)

Zusätzlich empfielt es sich, eine Batch-Datei `direkt.run` zu verfassen, um sich Schreibarbeit zu sparen:

```
reset;              # löscht das alte Modell
model direkt.mod;   # Laden des Modells
data direkt.dat;    # Laden der Daten
solve;
display X;
```

Um das Programm zu lösen und die Werte der Variablen anzuzeigen, ist nur noch `include direkt.run;` nach dem AMPL-Prompt einzugeben.

Beispiel 2: Produktionsplan

Ein (sehr vereinfachter) Sommersportartikel-Hersteller möchte einen Produktionsplan für die nächste Woche erstellen. Er kann Wasserbälle, Luftmatratzen und Schwimmflossen produzieren, die er um 25, 30 bzw. 29 Euro anbietet.
Für 1000 Wasserbälle, 500 Luftmatratzen und 750 Schwimmflossen hat er schon fixe Lieferverträge. Aus Erfahrung ist bekannt, dass maximal 6000, 4000 bzw. 3500 Stück Wasserbälle, Luftmatratzen bzw. Schwimmflossen abgesetzt werden können.

Weiters muss er folgende Produktionsbeschränkung berücksichtigen: Er hat insgesamt nur 35 Stunden auf der Färbe- und 40 auf der Zuschneidemaschine zur Verfügung, wobei der Maschinenoutput pro Stunde durch folgende Tabelle gegeben ist:

Produktionsrate:	*Färben*	*Zuschneiden*
Wasserbälle	*200*	*200*
Luftmatratzen	*200*	*140*
Schwimmflossen	*200*	*160.*

Der Hersteller will wissen, wieviel er von welchem Produkt fertigen soll, um den Profit unter den vorliegenden Beschränkungen zu maximieren, wobei er annimmt, dass jedes produzierte Stück auch verkauft wird.

Dazu erstellt man folgende Modell-Datei `sommer.mod`:

```
set PROD;      # Menge der Produkte
set MASCH;     # Menge der Maschinen

param rate {PROD, MASCH} > 0;     # Stück pro Stunde und Maschine
param stunden {MASCH} >= 0;       # verfügbare Maschinenstunden pro Woche

param profit {PROD};              # Profit pro Stück
param minbedarf {PROD} >= 0;      # mind. zu produzierende Stück pro Woche
param maxbedarf {PROD} >= 0;      # max. absetzbare Stück pro Woche

var Anzahl {p in PROD} >= minbedarf[p], <= maxbedarf[p];
# (gesuchte) Anzahl der zu produzierenden Stück

maximize gesamt_profit: sum {p in PROD} profit[p] * Anzahl[p];
# ZF: Gesamtprofit aller Produkte

subject to zeit {m in MASCH}: sum {p in PROD} (1/rate[p, m]) * Anzahl[p]
<= stunden[m];
# NB: Gesamtanzahl der Stunden, die für alle Produkte benötigt wird, darf verfügbare
# Stunden nicht überschreiten
```

Die Daten für das Produktionsmodell befinden sich in `sommer.dat`:

```
set PROD := Wasserbaelle  Luftmatratzen  Schwimmflossen;
set MASCH := Faerben  Zuschneiden;

param rate:     Faerben  Zuschneiden :=
Wasserbaelle    200      200
Luftmatratzen   200      140
Schwimmflossen  200      160;

param stunden := Faerben 35    Zuschneiden 40;

param:          profit  minbedarf  maxbedarf :=
Wasserbaelle    25      1000       6000
Luftmatratzen   30      500        4000
Schwimmflossen  29      750        3500;
```

Die Batch-Datei `sommer.run` enthält folgende Zeilen:

```
reset;              # löscht das alte Modell
model sommer.mod;   # liest das Modell ein
data sommer.dat;    # liest die Daten ein
solve;
display Anzahl;
```

Durch die Eingabe von `include sommer.run;` nach dem AMPL-Prompt erhält man die Optimallösung.

Beispiel 3: Einige Erweiterungen von Beispiel 2

Der Sommersportartikel-Hersteller möchte jetzt einen vierwöchigen Produktionsplan erstellen, der den Gesamtprofit maximiert, wobei folgende Zusatzinformationen berücksichtigt werden sollen:

Verfügbare Maschinenstunden:	*Färben*	*Zuschneiden*
Woche 1	*35*	*40*
Woche 2	*32*	*40*
Woche 3	*40*	*41*
Woche 4	*30*	*45*

Mindestbedarf:	*Wasserbälle*	*Luftmatratzen*	*Schwimmflossen*
Woche 1	*900*	*500*	*750*
Woche 2	*1000*	*450*	*700*
Woche 3	*950*	*550*	*650*
Woche 4	*1100*	*400*	*600*

Maximalbedarf:	*Wasserbälle*	*Luftmatratzen*	*Schwimmflossen*
Woche 1	*6000*	*4000*	*3400*
Woche 2	*6500*	*4200*	*3500*
Woche 3	*6200*	*4000*	*3300*
Woche 4	*6500*	*3900*	*3200.*

Die entsprechende erweiterte Modelldatei ist `sommer_ff.mod`:

```
set PROD;          # Menge der Produkte
set MASCH;         # Menge der Maschinen
param T > 0;       # Anzahl der Wochen

param rate {PROD, MASCH} > 0;         # Stück pro Stunde und Maschine
param stunden {MASCH, 1..T} >= 0;     # verfügbare Maschinenstunden je Woche

param profit {PROD};                  # Profit pro Stück
param minbedarf {PROD, 1..T} >= 0;    # mind. zu produzierende Stück je Woche
param maxbedarf {PROD, 1..T} >= 0;    # max. absetzbare Stück je Woche

var Anzahl {p in PROD, t in 1..T} >= minbedarf[p,t], <= maxbedarf[p,t];
# Anzahl der zu produzierenden Stück je Woche

maximize gesamt_profit: sum {p in PROD, t in 1..T} profit[p] * Anzahl[p,t];
# ZF: maximiere Gesamtprofit aller Produkte in allen Wochen
```

```
subject to zeit {m in MASCH, t in 1..T}:
        sum p in PROD (1/rate[p, m]) * Anzahl[p,t] <= stunden[m,t];
# NB: Gesamtanzahl der Stunden je Woche und Maschine für alle Produkte darf
# verfügbare Stunden je Woche und Maschine nicht überschreiten
```

Inhalt der zugehörigen Daten-Datei `sommer_ff.dat`:

```
set PROD := Wasserbaelle Luftmatratzen Schwimmflossen;
set MASCH := Faerben Zuschneiden;
param T := 4;

param rate :=
Wasserbaelle Faerben       200
Wasserbaelle Zuschneiden   200
Luftmatratzen Faerben      200
Luftmatratzen Zuschneiden  140
Schwimmflossen Faerben     200
Schwimmflossen Zuschneiden 160;

param stunden (tr):   Faerben   Zuschneiden :=
     1                35        40
     2                32        40
     3                40        41
     4                30        45;

param profit := Wasserbaelle 25
                Luftmatratzen 30
                Schwimmflossen 29;

param minbedarf (tr):   Wasserbaelle   Luftmatratzen   Schwimmflossen :=
     1                  900            500             750
     2                  1000           450             700
     3                  950            550             650
     4                  1100           400             600;

param maxbedarf (tr):   Wasserbaelle   Luftmatratzen   Schwimmflossen :=
     1                  6000           4000            3400
     2                  6500           4200            3500
     3                  6200           4000            3300
     4                  6500           3900            3200;
```

Die Batch-Datei `sommer_ff.run` sieht so aus:

```
reset;
model sommer_ff.mod;
data sommer_ff.dat;
solve;
display Anzahl;
```

Weitere Befehle

Änderung der Formatierung:

```
option display_1col 0;      # verändert die Ausgabe eines Vektors von Spalten- auf
                            # Zeilenschreibweise.
```

Ausgabe von Werten:

```
display zeit.dual, zeit.slack;    # Anzeige der Werte der dualen Variablen bzw.
# der Abstände der Werte der Variablen zur jeweils nächstgelegenen Schranke
display Anzahl.lb, Anzahl.ub, Anzahl.rc;    # Anzeige der durch die NB
# definierten unteren bzw. oberen Schranken sowie der reduzierten Kosten
```

Ändern von einzelnen Komponenten:

```
let profit["Luftmatratzen"] := 32;          # Änderung des Wertes eines Parameters

drop zeit["Zuschneiden", 2];                # Weglassen einer Bedingung
restore zeit["Zuschneiden", 2];             # Wiederaufnahme einer Bedingung

fix Anzahl["Luftmatratzen", 1] := 501;      # Fixierung einer Variable
unfix Anzahl["Luftmatratzen", 1] := 501;    # Aufhebung der Fixierung einer Variable
```

Explizite Wahl des Solvers:

```
option solver;           # Anzeige des derzeit verwendeten Solvers
option solver minos;     # Explizite Wahl des Solvers
```

Rücksetzen der Optionen:

```
reset option;
```

Definition einer ganzzahligen Variable:

```
var Anzahl {p in PROD, t in 1..T}
            >= minbedarf[p,t], <= maxbedarf[p,t], integer;
```

Ausgabe in eine Datei:

```
display Anzahl > sommer_d.out;    # Erzeugt eine neue Datei sommer_d.out und
# speichert darin die formatierte Ausgabe von Anzahl
display Anzahl.lb, Anzahl.ub >> sommer_d.out;
# Hinzufügen von Werten zur Datei sommer_d.out
close sommer_d.out;    # Schließen der Datei sommer_d.out

print Anzahl["Luftmatratzen", 1] > sommer_p.out;    # Erzeugt eine neue
# Datei sommer_p.out und speichert darin die angegebene Zahl
print Anzahl["Wasserbaelle", 1], Anzahl["Schwimmflossen", 1] >> sommer_p.out;
# Hinzufügen von Werten zur Datei sommer_p.out
close sommer_p.out;    # Schließen der Datei sommer_p.out
```