Επιχειρησιακή Έρευνα
Ενότητα 2
Βελτιστοποίηση με ΗΥ - AMPL

Αθανασία Μάνου

Διαπανεπιστημιακό Διατμηματικό
Μεταπτυχιακό Πρόγραμμα Σπουδών
Μαθηματικά της Αγοράς και της Παραγωγής

8 Οκτωβρίου 2021

Η γλώσσα AMPL

- Η γλώσσα AMPL (A Mathematical Programming Language) έχει σχεδιαστεί ειδικά για την επίλυση προβλημάτων βελτιστοποίησης.
- Επιτρέπει την εισαγωγή στον Η/Υ ενός προβλήματος βελτιστοποίησης σε μορφή πολύ κοντινή στη συνήθη μαθηματική μορφή.
- Συνεργάζεται με πολλούς αλγόριθμους επίλυσης.
- Συνεργάζεται με πολλούς τρόπους εισαγωγής των δεδομένων.
- Παρέχει πολλούς τρόπους διαχείρισης των αποτελεσμάτων.

Πηγές για την AMPL

- Ο ιστότοπος της AMPL βρίσκεται στο

  `http://www.ampl.com`
- Για να κατεβάσουμε δοκιμαστική έκδοση της AMPL επιλέγουμε
  - `TRY AMPL`
  - `DOWNLOAD A FREE DEMO`
  - `AMPL IDE download for ...`

    (ανάλογα με το λειτουργικό).
- Για το πλήρες εγχειρίδιο της AMPL πηγαίνουμε στο
  `https://ampl.com/resources/the-ampl-book/`

“Μικρό” παράδειγμα προγραμματισμού παραγωγής

- Ελαιοτριβείο παράγει δυο τύπους ελαιολάδου, τον “κλασικό” (εξευγενισμένο) και τον “παρθένο”.
- Το ελαιοτριβείο έχει διαθέσιμες 40 ώρες παραγωγής ανά εβδομάδα.
- 1 λίτρο κλασικού πωλείται 10 ευρώ.
  1 λίτρο παρθένου πωλείται 15 ευρώ.
- Το ελαιοτριβείο μπορεί να παράγει σε κάθε δεδομένη στιγμή μόνο έναν τύπο ελαιολάδου.
- Σε 1 ώρα μπορεί να παράγει 40 λίτρα κλασικού ή 30 λίτρα παρθένου.
- Το εμπορικό τμήμα του ελαιοτριβείου εκτιμά ότι μπορεί να πουλήσει το πολύ 1000 λίτρα κλασικού και 860 λίτρα παρθένου κάθε εβδομάδα.
- Να μεγιστοποιηθεί ο συνολικός τζίρος ανά εβδομάδα.

Μοντελοποίηση ως π.γ.π.

- Oil_c, Oil_v : Οι ποσότητες κλασικού και παρθένου ελαιολάδου που θα παραχθούν σε μια εβδομάδα.
- Έχουμε το π.γ.π.

$$\begin{array}{ll} \max & 10Oil_c + 15Oil_v \\ \text{υπό} & \frac{1}{40}Oil_c + \frac{1}{30}Oil_v \leq 40 \\ & 0 \leq Oil_c \leq 1000 \\ & 0 \leq Oil_v \leq 860. \end{array}$$

Μοντελοποίηση στην AMPL

- Το π.γ.π.

$$\begin{array}{ll} \max & 10Oil_c + 15Oil_v \\ \text{υπό} & \frac{1}{40}Oil_c + \frac{1}{30}Oil_v \leq 40 \\ & 0 \leq Oil_c \leq 1000 \\ & 0 \leq Oil_v \leq 860. \end{array}$$

γράφεται σε έναν text editor ως

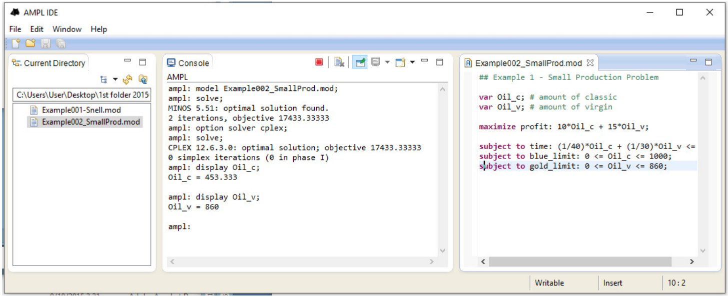
```
## Example 1 - Small Production Problem
var Oil_c; # amount of classic
var Oil_v; # amount of virgin
maximize profit: 10*Oil_c + 15*Oil_v;
subject to time: (1/40)*Oil_c + (1/30)*Oil_v <= 40;
subject to classic_limit: 0 <= Oil_c <= 1000;
subject to virgin_limit: 0 <= Oil_v <= 860;
```

Βασική σύνταξη στην AMPL

```
## Example 1 - Small Production Problem
var Oil_c; # amount of classic
var Oil_v; # amount of virgin
maximize profit: 10*Oil_c + 15*Oil_v;
subject to time: (1/40)*Oil_c + (1/30)*Oil_v <= 40;
subject to blue_limit: 0 <= Oil_c <= 1000;
subject to gold_limit: 0 <= Oil_v <= 860;
```

- `#` → Αρχή σχολίου.
- `var` → Δήλωση μεταβλητής.
- `;` → Τέλος γραμμής εντολής.
- `maximize` ή `minimize` + όνομα + `:` → Αντικ. συναρτ.
- `subject to` + όνομα + `:` → Περιορισμός.
- Case sensitive ονόματα. Τα ονόματα είναι μοναδικά.

Επίλυση στην AMPL IDE



Εκτέλεση, διαχείριση αρχείων στην AMPL

- Ένα αρχείο που περιέχει ένα πρόβλημα-μοντέλο πρέπει να σώζεται με την επέκταση `.mod`.
  Π.χ. το προηγούμενο θα μπορούσε να σωθεί σε κάποια θέση ως `z:\myFiles\ex01.mod`.
- Αν το αρχείο γραφεί σε text editor και έχει άλλη επέκταση (π.χ. `.txt`) πρέπει να αλλάξει η επέκταση.
- Για να εκτελεστεί ένα αρχείο πρέπει να τρέξουμε την εφαρμογή AMPL και να εμφανιστεί η αναμονή `ampl:`.
- Επιλέγουμε αλγόριθμο επίλυσης (solver).
  Π.χ. `option solver cplex;`.
- Φορτώνουμε το πρόβλημα-μοντέλο.
  Π.χ. `model z:\myFiles\ex_01.mod;`.
- Επιλύουμε με την εντολή `solve;`.

Διορθώσεις και εμφάνιση λύσης

- Αν υπάρξει λάθος στο μοντέλο κατά το τρέξιμο του αλγορίθμου επίλυσης τότε
  - Διορθώνουμε το `.mod` αρχείο στον text editor.
  - Δίνουμε την εντολή `reset;` στο AMPL.
  - Ξαναφορτώνουμε το μοντέλο.
    Π.χ. `model z:\myFiles\ex_01.mod;`.
- Η εντολή `solve;` θα μας δώσει κάποιες πληροφορίες και τη βέλτιστη τιμή της αντικειμενικής. Π.χ.
  ```
  CPLEX 6.5.3: optimal solution; objective 17433.33333
  2 simplex iterations (0 in phase I)
  ```
- Οι τιμές των μεταβλητών στη βέλτιστη λύση εμφανίζονται με την εντολή `display` + όνομα + `;`.
  Π.χ. `display Oil_c;`
  και `display Oil_v;`.

Αποθήκευση εξόδου σε αρχείο

- Για να αποθηκεύσουμε την έξοδο σε ένα αρχείο έχουμε δυο δυνατότητες, να δημιουργήσουμε-αντικαταστήσουμε (over-write) ένα αρχείο ή να προσθέσουμε περιεχόμενο σε ένα ήδη υπάρχον αρχείο.
- Π.χ. με την εντολή

  ```
  display Oil_c > z:\myFiles\ex_01.out;
  ```

  θα δημιουργηθεί αρχείο με το όνομα `ex_01.out` που θα αποθηκευτεί στη θέση `z:\myFiles`. Αν υπάρχει αρχείο με τέτοιο όνομα θα αντικατασταθεί.
- Με την εντολή

  ```
  display Oil_c >> z:\myFiles\ex_01.out;
  ```

  θα προστεθεί στο αρχείο με το όνομα `ex_01.out` η μεταβλητή Oil_c με την τιμή της.

Γενίκευση παραδ. προγραμματισμού παραγωγής

- Ελαιοτριβείο παράγει n τύπους ελαιολάδου.
- Το ελαιοτριβείο έχει διαθέσιμες t ώρες παραγωγής ανά εβδομάδα.
- 1 λίτρο τύπου i πωλείται p_i ευρώ.
- Το ελαιοτριβείο μπορεί να παράγει σε κάθε δεδομένη στιγμή μόνο έναν τύπο ελαιολάδου.
- Σε 1 ώρα μπορεί να παράγει r_i λίτρα τύπου i.
- Το εμπορικό τμήμα του ελαιοτριβείου εκτιμά ότι μπορεί να πουλήσει το πολύ m_i λίτρα τύπου i κάθε εβδομάδα.
- Να μεγιστοποιηθεί ο συνολικός τζίρος ανά εβδομάδα.

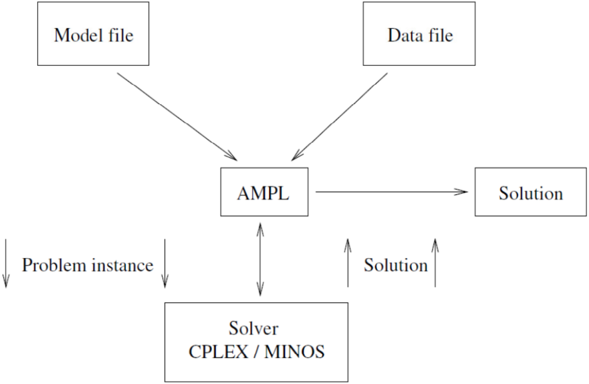
Μοντελοποίηση ως π.γ.π.

- x_i: Η ποσότητα ελαιολάδου τύπου i που θα παράγεται σε μια εβδομάδα.
- Έχουμε το π.γ.π.

$$
\begin{array}{ll}
\max & \sum_{i=1}^{n} p_i x_i \\
\text{υπό} & \sum_{i=1}^{n} (1/r_i) x_i \leq t \\
 & 0 \leq x_i \leq m_i, \ \ i = 1, 2, \ldots, n.
\end{array}
$$

- Για $n = 2$, $t = 40$, $p_1 = 10$, $p_2 = 15$, $m_1 = 1000$ και $m_2 = 860$ παίρνουμε το μικρό πρόβλημα παραγωγής.
- Το μικρό πρόβλημα παραγωγής είναι μια ειδική περίπτωση μοντέλου και όχι ένα μοντέλο.
- Το AMPL ενθαρρύνει το διαχωρισμό μοντέλου από τα δεδομένα και τον αλγόριθμο επίλυσης.

AMPL: Μοντέλο, δεδομένα, λύση



- Διάγραμμα αλληλεπίδρασης στοιχείων του AMPL

- Το π.γ.π.

$$\begin{array}{ll} \max & \sum_{i=1}^{n} p_i x_i \\ \text{υπό} & \sum_{i=1}^{n} (1/r_i) x_i \leq t \\ & 0 \leq x_i \leq m_i,\ i = 1, 2, \ldots, n. \end{array}$$

γράφεται ως

```
## Example 2 - Production Model
param n;
param t;
param p{i in 1..n};
param r{i in 1..n};
param m{i in 1..n};
var x{i in 1..n};
maximize profit: sum{i in 1..n} p[i]*x[i];
subject to time: sum{i in 1..n} (1/r[i])*x[i] <= t;
subject to capacity{i in 1..n}: 0 <= x[i] <= m[i];
```

Χρησιμοποιώντας δείκτες στην AMPL

```
## Example 2 - Production Model
param n;
param t;
param p{i in 1..n};
param r{i in 1..n};
param m{i in 1..n};
var x{i in 1..n};
maximize profit: sum{i in 1..n} p[i]*x[i];
subject to time: sum{i in 1..n} (1/r[i])*x[i] <= t;
subject to capacity{i in 1..n}: 0 <= x[i] <= m[i];
```

- `param` $\rightarrow$ Δήλωση παραμέτρου.
- `{i in 1..n}` $\rightarrow$ Δήλωση δείκτη και συνόλου διαδοχικών ακεραίων στους οποίου κινείται.
- Ένας δείκτης μπορεί να αφορά παραμέτρους, μεταβλητές, αθροίσεις ή περιορισμούς.

Δεδομένα στην AMPL I

```
## Example 2 - Data for the Production Model
param n:= 2;
param t:= 40;
param p:= 1 10 2 15;
param r:= 1 40 2 30;
param m:= 1 1000 2 860;
```

- param + όνομα + := → Δήλωση τιμών παραμέτρου.
- Αν μια παράμετρος είναι διανυσματική, τότε γράφουμε το δείκτη κάθε συνιστώσας και μετά την τιμή της.
- Το AMPL αγνοεί κενά και αλλαγές γραμμών (carriage returns).

Δεδομένα στην AMPL II

```
## Example 2 - Data for the Production Model
param n:= 2;
param t:= 40;
param p:= 1 10 2 15;
param r:= 1 40 2 30;
param m:= 1 1000 2 860;
```

$\Updownarrow$

```
## Example 2 - Data for the Production Model
param n:= 2;
param t:= 40;
param p:= 1 10
          2 15;
param r:= 1 40
          2 30;
param m:= 1 1000
          2  860;
```

Δεδομένα στην AMPL III

```
## Example 2 - Data for the Production Model
param n:= 2;
param t:= 40;
param p:= 1 10
          2 15;
param r:= 1 40
          2 30;
param m:= 1 1000
          2  860;
```

$\Updownarrow$

```
## Example 2 - Data for the Production Model
param n:= 2;
param t:= 40;
param:       p  r   m:=
          1 10 40 1000
          2 15 30  860;
```

AMPL Μοντέλο, δεδομένα, επίλυση

- Τυπική ακολουθία εντολών για την εισαγωγή μοντέλου, δεδομένων, την επίλυση και την εμφάνιση της λύσης:

```
reset;
model z:\myFiles\ex_02.mod;
data z:\myFiles\ex_02.dat;
solve;
display x;
```

Χρήση συνόλων στην AMPL I - Μοντέλο

```
## Example 2 - Production Model
param n;
param t;
param p{i in 1..n};
param r{i in 1..n};
param m{i in 1..n};
var x{i in 1..n};
maximize profit: sum{i in 1..n} p[i]*x[i];
subject to time: sum{i in 1..n} (1/r[i])*x[i] <= t;
subject to capacity{i in 1..n}: 0 <= x[i] <= m[i];
```

- Στην κωδικοποίηση του μοντέλου με δείκτες δεν υπάρχει σύνδεση των δεικτών των μεταβλητών με αυτό που μοντελοποιούν.
  Π.χ. το x[1] δεν παραπέμπει σε κάποιο συγκεκριμένο τύπο ελαιολάδου.

Χρήση συνόλων στην AMPL II - Μοντέλο

- Αυτό μπορεί να θεραπευτεί ορίζοντας και χρησιμοποιώντας σύνολα.
- Αντί για `param n;` ορίζουμε `set P;`.
- Αντί για `in 1..n` βάζουμε `in P`.

```
## Example 2 - Production Model - Modelling with sets
set P;
param t;
param p{i in P};
param r{i in P};
param m{i in P};
var x{i in P};
maximize profit: sum{i in P} p[i]*x[i];
subject to time: sum{i in P} (1/r[i])*x[i] <= t;
subject to capacity{i in P}: 0 <= x[i] <= m[i];
```

Χρήση συνόλων στην AMPL III - Δεδομένα

- Το αρχείο των δεδομένων πρέπει να προσαρμοστεί.
- Γράφουμε τα στοιχεία του συνόλου.
- Οι παράμετροι του συστήματος αναφέρονται με βάση τα στοιχεία του συνόλου.

```
## Example 2 - Data for the Production Model with sets
set P:= classic virgin;
param t:= 40;
param p:= classic 10
          virgin  15;
param r:= classic 40
          virgin  30;
param m:= classic 1000
          virgin   860;
```

Χρήση συνόλων στην AMPL IV - Δεδομένα

```
## Example 2 - Data for the Production Model with sets
set P:= classic virgin;
param t:= 40;
param p:= classic 10
          virgin  15;
param r:= classic 40
          virgin  30;
param m:= classic 1000
          virgin   860;
```

$\Updownarrow$

```
## Example 2 - Data for the Production Model with sets
set P:= classic virgin;
param t:= 40;
param:              p  r   m:=
          classic 10 40 1000
          virgin  15 30  860;
```

Μεταβλητές και παράμετροι με 2 δείκτες

- Σε προβλήματα μαθηματικού προγραμματισμού εμφανίζονται μεταβλητές και παράμετροι με 2 ή περισσότερους δείκτες.
- Στην AMPL, στο μοντέλο οι μεταβλητές και οι παράμετροι παριστάνονται όπως και πριν με 2 ή περισσότερους δείκτες που χωρίζονται με κόμμα.
- Στην AMPL, στα δεδομένα, οι τιμές των παραμέτρων παριστάνονται σε πινακική μορφή.

Παράδειγμα προβλήματος μεταφοράς

- Μια εταιρεία έχει 3 αποθήκες και 4 καταστήματα.
- Μια συγκεκριμένη εβδομάδα έχει να αποστείλει ένα είδος εμπορεύματος από τις αποθήκες στα καταστήματα.
- Για κάθε ζεύγος αποθήκης-καταστήματος υπάρχει διαφορετικό κόστος μεταφοράς ανά μονάδα προϊόντος που δίνεται.
- Για κάθε αποθήκη δίνεται το απόθεμά της για το συγκεκριμένο εμπόρευμα.
- Για κάθε κατάστημα δίνεται η ζήτησή του για το συγκεκριμένο εμπόρευμα.
- Το ζητούμενο είναι να ελαχιστοποιηθεί το συνολικό κόστος μεταφοράς από τις αποθήκες στα καταστήματα.

Δεδομένα προβλήματος μεταφοράς

- Τα κόστη μεταφοράς ανά μονάδα προϊόντος φαίνονται στον πίνακα:

	Καταστ. 1	Καταστ. 2	Καταστ. 3	Καταστ. 4
Αποθ. 1	1	2	1	3
Αποθ. 2	3	5	1	4
Αποθ. 3	2	2	2	2

- Τα αποθέματα στις αποθήκες είναι

	Αποθ. 1	Αποθ. 2	Αποθ. 3
Απόθεμα	250	800	760

- Οι ζητήσεις στα καταστήματα είναι

	Καταστ. 1	Καταστ. 2	Καταστ. 3	Καταστ. 4
Ζήτηση	300	320	800	390

Μοντελοποίηση στην AMPL

```
## Example 3 - Transportation Model

param warehouse; # number of warehouses
param shop; # number of shops

param cost{i in 1..warehouse, j in 1..shop};
#transportation cost from warehouse i to shop j
param supply{i in 1..warehouse};
#supply at warehouse i
param demand{i in 1..shop};
#demand at shop j

var amount{i in 1..warehouse, j in 1..shop};
```

Μοντελοποίηση στην AMPL (συνέχεια)

```
minimize Cost:
sum{i in 1..warehouse, j in 1..shop}
  cost[i,j]*amount[i,j];

subject to Supply {i in 1..warehouse}:
sum{j in 1..shop} amount[i,j] = supply[i];

subject to Demand {j in 1..shop}:
sum{i in 1..warehouse} amount[i,j] = demand[j];

subject to positive{i in 1..warehouse, j in 1..shop}:
amount[i,j]>=0;
```

Μοντελοποίηση στην AMPL (παρατηρήσεις)

- Κάθε παράμετρος και μεταβλητή έχουν περιγραφικά ονόματα.
- Οι παράμετροι και οι μεταβλητές με διπλούς δείκτες παριστάνονται όπως πριν, με τους δείκτες να χωρίζονται με κόμμα.
- Οι περιορισμοί και η αντικειμενική συνάρτηση ονομάζονται με τα ίδια ονόματα με κάποιες μεταβλητές, αλλά με κεφαλαίο το πρώτο γράμμα.

Δεδομένα στην AMPL

```
## Example 3 - Transportation Model Data
param warehouse:= 3;
param shop:= 4;
param cost: 1 2 3 4 :=
          1 1 2 1 3
          2 3 5 1 4
          3 2 2 2 2;
param supply:= 1 250 2 800 3 760;
param demand:= 1 300 2 320 3 800 4 390;
```

- Τα δεδομένα αυτά πρέπει να σωθούν σε ένα αρχείο με επέκταση `.dat`.
- Παρατηρείστε πώς εισάγονται οι τιμές στην παράμετρο `cost`:
  Μετά το όνομα της παραμέτρου μπαίνει `:` και
  μετά το όνομα του τελευταίου δείκτη μπαίνει `:=`

Μοντελοποίηση με σύνολα στην AMPL

```
## Example 3 - Transportation Model using sets

set Warehouses;
set Shops;

param cost{i in Warehouses, j in Shops};
#transportation cost from warehouse i to shop j
param supply{i in Warehouses};
#supply at warehouse i
param demand{j in Shops};
#demand at shop j

var amount{i in Warehouses, j in Shops};
```

Μοντελοποίηση με σύνολα στην AMPL (συνέχεια)

```
minimize Cost:
sum{i in Warehouses, j in Shops}
  cost[i,j]*amount[i,j];

subject to Supply {i in Warehouses}:
sum{j in Shops} amount[i,j] = supply[i];

subject to Demand {j in Shops}:
sum{i in Warehouses} amount[i,j] = demand[j];

subject to positive{i in Warehouses, j in Shops}:
amount[i,j]>=0;
```

Δεδομένα με σύνολα στην AMPL

```
## Example 3 - Transportation Model Data with sets
set Warehouses:= Oakland San_Jose Albany;
set Shops:= Home_Depot K_mart Wal_mart Ace;
param cost: Home_Depot K_mart Wal_mart Ace:=
Oakland          1        2        1       3
San_Jose         3        5        1       4
Albany           2        2        2       2;
param supply:= Oakland      250
              San_Jose     800
              Albany       760;
param demand:= Home_Depot   300
              K_mart       320
              Wal_mart     800
              Ace          390;
```

Περιορισμοί ακεραιότητας στην AMPL

- Για να δηλώσουμε ότι μια μεταβλητή περιορίζεται να είναι ακέραιη, γράφουμε `integer` στην εντολή εισαγωγής της, μετά το όνομά της και πριν το ;
- Π.χ.: Αν θέλουμε η μεταβλητή `x` να είναι ακέραια, την εισάγουμε γράφοντας:
  `var x integer;`
- Για να δηλώσουμε ότι μια μεταβλητή περιορίζεται να είναι δυαδική (0-1), γράφουμε `binary` στην εντολή εισαγωγής της, μετά το όνομά της και πριν το ;
- Π.χ.: Αν θέλουμε η μεταβλητή `y` να είναι δυαδική, την εισάγουμε γράφοντας:
  `var y binary;`
- Πρέπει ο αλγόριθμος επίλυσης που επιλέγουμε (solver) να υποστηρίζει επίλυση με ακέραιες μεταβλητές.

Παραγωγή με ακέραιες ποσότητες

- Στο αρχικό παράδειγμα το ελαιόλαδο πωλείται σε συσκευασίες του λίτρου.
- Δεν έχει νόημα να παραχθεί μη-ακέραιος αριθμός συσκευασιών.
- Το π.γ.π. τώρα γίνεται:

$$
\begin{array}{ll}
\max & 10Oil_c + 15Oil_v \\
\text{υπό} & \frac{1}{40}Oil_c + \frac{1}{30}Oil_v \leq 40 \\
& 0 \leq Oil_c \leq 1000 \\
& 0 \leq Oil_v \leq 860 \\
& Oil_c, Oil_v \text{ ακέραιες.}
\end{array}
$$

Μοντέλο στην AMPL

```
## Example 1 - Small Production Problem
## with integer variables
var Oil_c integer; # amount of classic
var Oil_v integer; # amount of virgin
maximize profit: 10*Oil_c + 15*Oil_v;
subject to time: (1/40)*Oil_c + (1/30)*Oil_v <= 40;
subject to classic_limit: 0 <= Oil_c <= 1000;
subject to virgin_limit: 0 <= Oil_v <= 860;
```

Πρόβλημα επιλογής εγκαταστάσεων

- Πρόβλημα επιλογής εγκαταστάσεων = Πρόβλημα μεταφοράς + Κόστη εγκατάστασης για τους σταθμούς αφετηρίας (αποθήκες) που θα αποφασιστεί να χρησιμοποιηθούν.
- Τα κόστη μεταφοράς ανά μονάδα προϊόντος φαίνονται στον πίνακα:

	Καταστ. 1	Καταστ. 2	Καταστ. 3	Καταστ. 4
Αποθ. 1	1	2	1	3
Αποθ. 2	3	5	1	4
Αποθ. 3	2	2	2	2

- Τα αποθέματα στις αποθήκες είναι

	Αποθ. 1	Αποθ. 2	Αποθ. 3
Απόθεμα	550	1100	1060

Πρόβλημα επιλογής εγκαταστάσεων (συνέχεια)

- Οι ζητήσεις στα καταστήματα είναι

	Καταστ. 1	Καταστ. 2	Καταστ. 3	Καταστ. 4
Ζήτηση	300	320	800	390

- Τα κόστη εγκατάστασης (εκκίνησης) είναι

	Αποθ. 1	Αποθ. 2	Αποθ. 3
Κόστος Εγκατάστ.	500	500	500

Μοντελοποίηση

- Για τη μοντελοποίηση χρησιμοποιούμε 0-1 μεταβλητές $open_i$, $i = 1, 2, 3$:

$$open_i = \begin{cases} 1, & \text{αν η αποθήκη } i \text{ χρησιμοποιηθεί,} \\ 0, & \text{διαφορετικά.} \end{cases}$$

- $open_i = 0 \Rightarrow amount_{ij} = 0$ για κάθε j.
  Αν η αποθήκη i δεν χρησιμοποιηθεί τότε δεν θα μεταφερθούν ποσότητες από αυτήν προς κανένα κατάστημα.
- Συνεπώς εισάγουμε τους περιορισμούς:

$$0 \leq amount_{ij} \leq supply_i \times open_i.$$

Μοντέλο στην AMPL

```
## Example 4 - Mixed-IP model file for the
## warehouse location problem

set Warehouses;
set Shops;

param cost{i in Warehouses, j in Shops};
#transportation cost from warehouse i to shop j
param supply{i in Warehouses};
#supply capacity at warehouse i
param demand{j in Shops};
#demand at shop j
param fixed_charge{i in Warehouses};
#cost of opening warehouse j
```

Μοντέλο στην AMPL (συνέχεια)

```
var amount{i in Warehouses, j in Shops};
var open{i in Warehouses} binary;
# = 1 if warehouse i is opened, 0 otherwise

minimize Cost:
sum{i in Warehouses, j in Shops}
cost[i,j]*amount[i,j]
+ sum{i in Warehouses} fixed_charge[i]*open[i];

subject to Supply {i in Warehouses}:
sum{j in Shops} amount[i,j] <= supply[i]*open[i];
subject to Demand {j in Shops}:
sum{i in Warehouses} amount[i,j] = demand[j];
subject to positive{i in Warehouses, j in Shops}:
amount[i,j]>=0;
```

Δεδομένα στην AMPL

```
## Example 4 - Data file for
## the warehouse location problem

set Warehouses:= Oakland San_Jose Albany;
set Shops:= Home_Depot K_mart Wal_mart Ace;

param cost: Home_Depot K_mart Wal_mart Ace:=
  Oakland        1          2        1     3
  San_Jose       3          5        1     4
  Albany         2          2        2     2;

param supply:=
  Oakland  550
  San_Jose 1100
  Albany   1060;
```

Δεδομένα στην AMPL (συνέχεια)

```
param demand:=
  Home_Depot 300
  K_mart     320
  Wal_mart   800
  Ace        390;

param fixed_charge:=
  Oakland  500
  San_Jose 500
  Albany   500;
```

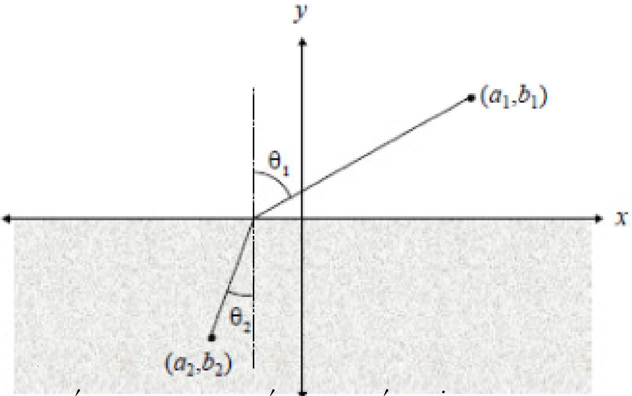
Μη-γραμμικός προγραμματισμός και AMPL

- Η AMPL μπορεί να λύνει και προβλήματα μη-γραμμικού προγραμματισμού.
- Θα πρέπει να επιλεγεί αλγόριθμος επίλυσης (solver) που υποστηρίζει τη λύση τέτοιων προβλημάτων.
  Π.χ. ο solver `cplex` δεν υποστηρίζει επίλυση μη-γραμμικών προβλημάτων προγραμματισμού, ενώ ο `minos` υποστηρίζει.
  Οπότε θα πρέπει να δώσουμε την εντολή:
  `option solver minos;`
- Η διαδικασία επίλυσης μπορεί να καταλήγει μόνο σε τοπικό ακρότατο ή σε κρίσιμο σημείο, οπότε τα αποτελέσματα πρέπει να χρησιμοποιούνται με προσοχή.

Παράδειγμα: Ο νόμος του Snell

- Έστω πηγή φωτός σε ένα μέσο.
- Έστω δέκτης σε άλλο μέσο.
- Τα δυο μέσα χωρίζονται από ένα επίπεδο.
- Αρχή του Fermat: Το φως που φθάνει στον δέκτη ακολουθεί το μονοπάτι ελάχιστου χρόνου για να φθάσει από την πηγή στο δέκτη.

Ο νόμος του Snell (συνέχεια)



- v_i η ταχύτητα του φωτός στο μέσο i.
- Χρόνος αν το φως περνάει από το σημείο $(0,x)$:

$$T(x) = \frac{\sqrt{(a_1 - x)^2 + b_1^2}}{v_1} + \frac{\sqrt{(x - a_2)^2 + b_2^2}}{v_2}.$$

Ο νόμος του Snell (συνέχεια)

- Χρόνος αν το φως περνάει από το σημείο $(0, x)$:

$$T(x) = \frac{\sqrt{(a_1 - x)^2 + b_1^2}}{v_1} + \frac{\sqrt{(x - a_2)^2 + b_2^2}}{v_2}.$$

- Παράγωγος:

$$\frac{dT}{dx}(x) = -\frac{1}{v_1}\frac{a_1 - x}{\sqrt{(a_1 - x)^2 + b_1^2}} + \frac{1}{v_2}\frac{x - a_2}{\sqrt{(x - a_2)^2 + b_2^2}}.$$

- $\frac{dT}{dx}(x) = 0 \Rightarrow$ Νόμος του Snell:

$$\frac{\sin\theta_1}{v_1} = \frac{\sin\theta_2}{v_2}.$$

```
## Example 5
## Snell's law of refraction obtained by minimizing
## the time for the light to get from point a to b.

param a{1..2};
param b{1..2};
param v1;
param v2;

var x;
minimize time:
sqrt((a[1]-x)^2 + b[1]^2)/v1
  + sqrt((x-a[2])^2 + b[2]^2)/v2;
```

Δεδομένα στην AMPL

```
data;

param a :=
1 1
2 -1 ;

param b :=
1 1
2 -1 ;

param v1 := 1;
param v2 := 0.8;
```

Επίλυση στην AMPL

```
option solver minos;

solve;

display x;

display ((a[1]-x)/sqrt((a[1]-x)^2 + b[1]^2))
/ ((x-a[2])/sqrt((x-a[2])^2 + b[2]^2));

quit;
```