Π.Μ.Σ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ & ΝΕΦΟΣ

Μέθοδοι και Εργαλεία Τεχνητής Νοημοσύνης (SD0203)

Εργασία 1

Ονοματεπώνυμο : Χρυσοχοΐδης Αναστάσιος
Αριθμός Μητρώου : mai25067
email : mai250067@uom.edu.gr
Απρίλιος 2025

Άσκηση 1 : Client/Server Allocation

Για την πρώτη άσκηση έπρεπε να βρεθεί σε ποιον server θα πρέπει αν φιλοξενηθεί κάθε υπηρεσία με το ελάχιστο δυνατό κόστος. Σε περίπτωση που ένας server δεν φιλοξενήσει κάποια υπηρεσία θεωρείται μη ενεργός, οπότε δεν έχει και κόστος. Ως δεδομένα μας δίνονται η χωρητικότητα του κάθε server, οι απαιτήσεις κάθε υπηρεσίας σε πόρους, το κόστος λειτουργίας κάθε server, και το κόστος φιλοξενίας της κάθε υπηρεσίας σε κάθε έναν server.

Ως πρώτο περιορισμό έχουμε ότι σε έναν ενεργό server δεν δύναται να ξεπερνιέται η χωρητικότητά του από το σύνολο των απαιτούμενων πόρων των υπηρεσιών που φιλοξενεί. Οπότε πρέπει να φτιαχτεί ένας πίνακας όπου θα υπολογίζεται το άθροισμα των απαιτούμενων πόρων για κάθε δυνατή περίπτωση.

Αυτό χρησιμοποιείται στον δεύτερο περιορισμό, όπου το άθροισμα των απαιτούμενων πόρων πρέπει να είναι μικρότερο ή ίσο της χωρητικότητας του εκάστοτε server.

Επίσης, πρέπει να καθοριστεί ποιοι server είναι ενεργοί ή όχι και αυτό καθορίζεται από τον τρίτο περιορισμό, όπου έχοντας φτιάξει έναν πίνακα που μπορεί να παίρνει τις τιμές 0 και 1, ανάλογα με το αν χρησιμοποιείται κάποιος server (φαίνεται από το πίνακα με το άθροισμα των απαιτούμενων πόρων) με το 0 να δηλώνει μη ενεργό server και το 1 ενεργό server.

Ύστερα, υπολογίζεται το συνολικό δικτυακό κόστος που προκύπτει ως το άθροισμα των απαιτήσεων κάθε υπηρεσίας σε πόρους επί το κόστος φιλοξενίας κάθε υπηρεσίας σε συγκεκριμένο server. Δηλαδή :

δικτυακό κόστος = sum(απαιτήσεις υπηρεσίας σε πόρους * κόστος φιλοξενίας υπηρεσίας σε συγκεκριμένο server)

και το κόστος ανάπτυξης του κάθε server που προκύπτει από το άθροισμα των κοστών λειτουργίας τους μόνο στην περίπτωση που είναι ενεργοί. Τέλος, υπολογίζεται το συνολικό κόστος που προκύπτει ως εξής :

συνολικό κόστος = συνολικό δικτυακό κόστος + συνολικό κόστος λειτουργίας

Παρακάτω ακολουθεί ο κώδικας που γράφτηκε στο MiniZinc.

```
%% Number of Servers and Clients and related Sets
par int: Servers;
par int: Clients;
set of int:SERVERS = 1..Servers;
set of int:CLIENTS = 1..Clients;

%% Arrays stating Problem Data
array[SERVERS] of int: Capacity;
array[SERVERS] of int: DeploymentCost;
array[CLIENTS] of int: ClientDemands ;
array[CLIENTS,SERVERS] of int: NetworkingCost;

%% Add your code here
array[CLIENTS] of var SERVERS: client_at;
var int: total_cost;

array[SERVERS] of var int : serverLoad;
array[SERVERS] of var 0..1 : usedServers;

%Constraint 1:save in an array the total amount of
services load on each server
constraint forall(s in SERVERS) (serverLoad[s] =
sum(c in CLIENTS where client_at[c] = s)
(ClientDemands[c]));

%Constraint 2: a server's total load should be less
than the server's capacity
constraint forall(s in SERVERS)(serverLoad[s] <=
Capacity[s]);

%Constraint 3: if a server's load is greater than 0
then thr server is active (hence 1), else if the
server's laod is 0 then is inactive (hence 0)
constraint forall(s in SERVERS) (usedServers[s] = (if
serverLoad[s] > 0 then 1 else 0 endif));

var int : totalNetworkingCost = sum(c in CLIENTS)
(ClientDemands[c] * NetworkingCost[c, client_at[c]]);
var int : totalDeploymentCost = sum(s in SERVERS)
(usedServers[s] * DeploymentCost[s]);

total_cost = totalNetworkingCost +
totalDeploymentCost;
solve minimize total_cost;
%%% DO NOT DELETE THIS LINES
output [" client_at = \(client_at); \n total_cost=\(total_cost);\n"];
```

Τα αποτελέσματα που παράγονται με την εκτέλεση του προγράμματος, με τα δεδομένα που μας δίνονται έχουν ως εξής :

```
 client_at = [1, 4, 1, 1, 2, 4];
 total_cost=5796;
----------
 client_at = [3, 4, 1, 3, 1, 4];
 total_cost=5462;
----------
 client_at = [1, 4, 1, 1, 3, 4];
 total_cost=5406;
----------
 client_at = [3, 4, 3, 3, 3, 4];
 total_cost=5067;
----------
==========
Finished in 386msec.
```

Αποτελέσματα με αρχείο δεδομένων το server_set0.dzn

```
----------
 client_at = [3, 1, 4, 1, 3, 4, 3, 2, 3, 2];
 total_cost=11096;
----------
 client_at = [2, 2, 4, 1, 3, 3, 1, 3, 2, 4];
 total_cost=10610;
----------
 client_at = [2, 1, 4, 1, 3, 3, 2, 3, 2, 4];
 total_cost=10202;
----------
 client_at = [3, 1, 4, 1, 3, 2, 3, 2, 3, 4];
 total_cost=10056;
----------
==========
Finished in 413msec.
```

Αποτελέσματα με αρχείο δεδομένων το server_set1.dzn

```
----------
 client_at = [7, 3, 2, 7, 3, 2, 7, 7, 7, 3];
 total_cost=5718;
----------
 client_at = [7, 3, 7, 2, 3, 2, 7, 7, 3, 3];
 total_cost=5665;
----------
 client_at = [2, 3, 7, 7, 3, 2, 7, 7, 7, 3];
 total_cost=5616;
----------
 client_at = [9, 3, 7, 7, 3, 9, 7, 7, 7, 3];
 total_cost=5613;
----------
==========
Finished in 411msec.
```

Αποτελέσματα με αρχείο δεδομένων το server_set2.dzn

```
----------
 client_at = [1, 4, 1, 7, 1, 7, 3, 1, 7, 7, 7, 4, 1, 1, 3];
 total_cost=7258;
----------
 client_at = [1, 4, 1, 1, 1, 7, 3, 1, 7, 4, 7, 4, 1, 7, 3];
 total_cost=7246;
----------
 client_at = [1, 4, 1, 7, 1, 7, 3, 1, 1, 4, 7, 4, 1, 7, 3];
 total_cost=7185;
----------
 client_at = [1, 3, 1, 7, 1, 7, 3, 1, 1, 8, 7, 8, 1, 7, 3];
 total_cost=7134;
----------
==========
Finished in 458msec.
```

Αποτελέσματα με αρχείο δεδομένων το server_set3.dzn

```
----------
 client_at = [1, 6, 1, 5, 1, 5, 3, 1, 1, 6, 6, 6, 1, 5, 3, 6, 3, 3, 5, 1];
 total_cost=8728;
----------
 client_at = [1, 4, 1, 7, 1, 7, 3, 1, 1, 4, 7, 4, 1, 7, 3, 4, 3, 3, 4, 1];
 total_cost=8674;
----------
 client_at = [1, 6, 1, 7, 1, 7, 3, 1, 7, 6, 6, 6, 1, 7, 3, 6, 3, 3, 6, 1];
 total_cost=8655;
----------
 client_at = [1, 6, 1, 7, 1, 7, 3, 1, 1, 6, 7, 6, 1, 7, 3, 6, 3, 3, 6, 1];
 total_cost=8559;
----------
==========
Finished in 15s 593msec.
```

Αποτελέσματα με αρχείο δεδομένων το server_set4.dzn

Άσκηση 2 : FireTrucks

Για την δεύτερη άσκηση έπρεπε να βρεθούν οι χρόνοι έναρξης για την πλήρωση και μετέπειτα την επισκευή όλων των πυροσβεστικών οχημάτων χρησιμοποιώντας ως πόρους 3 κρουνούς και 1 συνεργείο για service αντίστοιχα, στον ελάχιστο δυνατό χρόνο. Για την επίλυση των προβλήματος έπρεπε να βρεθούν για κάθε όχημα οι παρακάτω πληροφορίες :

1. Διάρκεια πλήρωσης του κάθε οχήματος, όπου ο κάθε κρουνός γεμίζει με 50lt ανα χρονική στιγμή και το κάθε όχημα έχει διαφορετική χωρητικότητα δεδομένη από την εκφώνηση της άσκησης
2. Χρονική στιγμή περάτωσης πλήρωσης
3. Χρονική στιγμή περάτωσης service, όπου το κάθε όχημα έχει διαφορετικό χρόνο συντήρησης δεδομένος από την εκφώνηση της άσκησης

Δημιουργείται ένας πίνακας όπου υπολογίζεται η διάρκεια πλήρωσης για κάθε όχημα ως εξής :

διάρκεια = χωρητικότητα οχήματος / 50

Η διάρκεια μας βοηθάει να υπολογίσουμε την χρονική στιγμή περάτωσης πλήρωσης για κάθε όχημα όπου :

περάτωση πλήρωσης = έναρξη πλήρωσης + διάρκεια πλήρωσης

Η χρόνοι περάτωσης των service για κάθε όχημα υπολογίζεται ως εξής :

περάτωση service = έναρξη service + διάρκεια service

Πλέον, έχοντας τα απαραίτητα στοιχεία και τις σχέσεις μεταξύ τους, μπορούμε να τα βάλουμε σε περιορισμούς . Στην επίλυση χρησιμοποιήθηκαν οι παρακάτω περιορισμοί :

1. Οι χρονικές στιγμές έναρξης service πρέπει να είναι μεγαλύτερες από τις χρονικές στιγμές περάτωσης πλήρωσης για κάθε όχημα
2. Μόνο ένα όχημα μπορεί να βρίσκεται σε κατάσταση service (μιας και υπάρχει μόνο 1 συνεργείο). Χρησιμοποιήθηκε ο περιορισμός disjunctive, χρησιμοποιώντας τους χρόνους έναρξης και την διάρκεια service για κάθε όχημα.
3. Τα οχήματα που γεμίζουν ταυτόχρονα δεν μπορεί να είναι πάνω από 3 (μιας και έχουμε 3 κρουνούς). Χρησιμοποιήθηκε ο περιορισμός cumulative, χρησιμοποιώντας τους χρόνους έναρξης και διάρκειας της πλήρωσης και τους 3 κρουνούς
4. Ο συνολικός χρόνος είναι στην ουσία η μέγιστη τιμή του πίνακα που δείχνει τις χρονικές στιγμές περάτωσης service, μιας και το service είναι πάντα μετά την πλήρωση και γίνεται 1 την φορά.
5. Ο συνολική διάρκεια service είναι η μέγιστη τιμή του πίνακα με τις χρονικές στιγμές περάτωσης service μείον την ελάχιστη τιμή του πίνακα με τις χρονικές στιγμές έναρξης service.

Παρακάτω ακολουθεί ο κώδικας που γράφτηκε στο MiniZinc.

```
include "globals.mzn";

int: no_firetrucks;
set of int: FIRETRUCKS = 1..no_firetrucks;
array[FIRETRUCKS] of int: water_cap;
array[FIRETRUCKS] of int: service_time;

%% Decision Variables - You can change anything, BUT
the names of the variables.
array[FIRETRUCKS] of var 0..1000: start_fill;
array[FIRETRUCKS] of var 0..1000: start_service;
var 0..1000: makespan;
var 0..1000: total_service;

%%% Add your code here

var int: fire_hydrant = 3;
var int: fill_per_min = 50;
array[FIRETRUCKS] of var int: time_fill =
[water_cap[i] div fill_per_min | i in FIRETRUCKS];
array[FIRETRUCKS] of var int: end_fill =
[start_fill[i] + time_fill[i] | i in FIRETRUCKS];
array[FIRETRUCKS] of var int: end_service =
[start_service[i] + service_time[i] | i in
FIRETRUCKS];

%Constraint 1 : fill happens first and after that it
goes to service
%end_fill for each firetruck is the time
start_service can begin
constraint forall(f in FIRETRUCKS) (start_service[f]
>= end_fill[f]);

%Constraint 2 : only 1 firetruck can be for service
(1 service at a time)
constraint disjunctive([start_service[f] | f in
FIRETRUCKS], [service_time[f] | f in FIRETRUCKS]);

%Constraint 3 : no more than 3 firetrucks can fill at
the same time
constraint cumulative([start_fill[f] | f in
FIRETRUCKS], [time_fill[f] | f in FIRETRUCKS], [1 | f
in FIRETRUCKS],fire_hydrant);
```

```
%Constraint 4: total time for fill and service is the
maximum time of the end of service for the last
truck, since service happens after the fill
constraint makespan = max([end_service[f] | f in
FIRETRUCKS]);

%Constraint 5: total service time is from the moment
the first truck goes to service(min start time)
%until the last truck finishes(max start time +
service time)
constraint total_service = max(end_service) -
min(start_service);

solve minimize makespan;

output [" start_fill=\(start_fill);\n
start_service=\(start_service);\n"]
    ++ [" total_service = \(total_service);\n makespan=\(makespan);\n"]
```

Τα αποτελέσματα που παράγονται με την εκτέλεση του προγράμματος, με τα δεδομένα που μας δίνονται έχουν ως εξής :

```
Running FireTrucks.mzn, FireData1.dzn
 start_fill=[12, 0, 0, 18, 30, 0];
 start_service=[70, 18, 30, 85, 95, 50];
 total_service = 82;
 makespan=100;
----------
 start_fill=[28, 26, 0, 0, 0, 16];
 start_service=[81, 71, 31, 16, 26, 51];
 total_service = 80;
 makespan=96;
----------
 start_fill=[0, 30, 0, 24, 12, 0];
 start_service=[52, 82, 32, 67, 77, 12];
 total_service = 80;
 makespan=92;
----------
==========
Finished in 712msec.
```

Αποτελέσματα με αρχείο δεδομένων το FireData1.dzn

```
Running FireTrucks.mzn, FireData2.dzn
 start_fill=[8, 30, 0, 26, 0, 0, 36, 32];
 start_service=[65, 90, 30, 80, 50, 8, 105, 100];
 total_service = 102;
 makespan=110;
----------
 start_fill=[8, 30, 0, 30, 4, 0, 32, 0];
 start_service=[65, 90, 30, 80, 50, 9, 100, 4];
 total_service = 101;
 makespan=105;
----------
 start_fill=[10, 34, 0, 30, 8, 0, 4, 0];
 start_service=[69, 94, 34, 84, 54, 9, 29, 4];
 total_service = 100;
 makespan=104;
----------
==========
Finished in 725msec.
```

Αποτελέσματα με αρχείο δεδομένων το FireData2.dzn