Πανεπιστημιο Μακεδονιας

Τμημα Εφαρμοσμενης Πληροφορικης

Εργασία 3 για το μάθημα 'Σχεδιασμός και Χρονοπρογραμματισμός'

Κωνσταντίνος Πασβάντης
Email: aid23005@uom.edu.gr
Αριθμός Μητρώου: aid23005

26 Απριλίου 2023

1 SuperMarket

Στο συγκεκριμένο πρόβλημα έπρεπε να βρούμε την κατάλληλη επιλογή προσφορών από τις επιλογές που μας δίνεται, προκειμένου να μεγιστοποιήσουμε την ποιότητα που μας παρέχεται. Ο κώδικας ο οποίος χρησιμοποίησα είναι ο παρακάτω:

```
%All packages must be different
include "all_different.mzn";
constraint all_different(basket);

%Two new arrays including the cost and and the quality of each package that we are buying
array[BUYS] of var int: cost_of_packages;
array[BUYS] of var int: quality_of_packages;

%Giving each new array the appropriate value
constraint forall(t in BUYS)
( cost_of_packages[t] = package_price[basket[t]] /\quality_of_packages[t] = package_quality[basket[t]]);

%Calculating quality and price by adding the values of the arrays
constraint quality = sum(t in BUYS)(quality_of_packages[t]);
constraint price = sum(t in BUYS)(cost_of_packages[t]);

% We create an array that holds the offer we are picking, and then we must have different categories in
each offer.
array[BUYS] of var set of int : basket_offer;
constraint forall (t in BUYS)
(basket_offer[t] = package[basket[t]]);
constraint forall (i,j in 1..4 where i!=j)( (basket_offer[i] intersect basket_offer[j]) = {} );

%Price of packages must be lower than max money
constraint price < max_money ;

%Solving the problem
solve maximize quality;
```

Τα σχόλια του κώδικα φαίνονται παραπάνω. Στην αρχή συμπεριλαβα το πακέτο all_different, καθώς θέλουμε το καλάθι να μην περιλαμβάνει ίδιες προσφορές. Στην συνέχεια δημιούργησα δύο πίνακες οι οποίοι θα έχουν τα κόστη και την ποιότητα του κάθε πακέτου που θα έχει μέσα το καλάθι μας, χρησιμοποιώντας τον κατάλληλο περιορισμό με τα σωστα ινδεχες των πακέτων που βρίσκεται στο καλάθι. Μετά, αθροίζωντας όλες τις τιμές από τον κάθε πίνακα που δημιούργησα υπολογίζω την τιμή των μεταβλητών qualityκαι price και λέω ότι η τιμή της μεταβλητής money θα πρέπει να είναι μικρότερη από το πόσα λεφτά διαθέτουμε. Τέλος, με την βοήθεια ενός νέου πίνακα που περιλαμβάνει τις προσφορές που θα αγοράσουμε, βάζω τον περιορισμό ότι σε κάθε προσφορά θα πρέπει να υπάρχουν και διαφορετικες κατηγορίες προιόντων, κάτι το οποίο επιτυγχάνεται με την βοήθεια της συνάρτησης ιντερσεςτ, καθώς οι προσφορές είναι set . Το αποτέλεσμα του παραπάνω κώδικα σε συνδυασμό με αυτόν που ήδη υπήρχε στο αρχείο που μας δώθηκε, είναι:

```
%%% Just for Debuging: {1,2,5} {3,6} {4,7} {}
basket = [6, 3, 2, 0];
quality = 6;
price = 60;
----------
%%% Just for Debuging: {3,7} {4,6} {1,2,5} {}
basket = [8, 7, 6, 0];
quality = 10;
price = 97;
----------
==========
```

Γενικά στον παραπάνω κώδικα δεν παρατήρησα κάποιο συγκεκριμένο bug.

2 Flow

Στο συγκεκριμένο πρόβλημα έπρεπε να βρούμε τον σωστό τρόπο να τοποθετήσουμε τις συσκεύες με τα δύο πορτ που μας δίνονται, προκειμένου να ελαχιστοποιήσουμε την απόσταση των καλωδίων που θα χρειαστούμε. Ο κώδικας ο οποίος χρησιμοποίησα είναι ο παρακάτω:

% We store wich port is first and wich second of each row from device_levels
array[DEVICES] of var 1..2: second_port;
array[DEVICES] of var 1..2: first_port;

%The above matrices must have different elements in the same index
constraint forall (t in DEVICES) (first_port[t]!=second_port[t]);

%FInally, we caclulate the total_length by adding the device level of the first device's first port, and the sum of absolute differences between the device_levels of the first port of a device, and the device level of the second port of its next device.
var float:total_length;
constraint total_length = device_levels[1,first_port[1]] + sum(t in DEVICES where t < ndevices)
(abs(device_levels[t+1,first_port[t+1]] -device_levels[t,second_port[t]]));

%We want to minimize the total_length
solve minimize total_length;

Τα σχόλια του κώδικα φαίνονται παραπάνω. Αρχικά, αποφάσισα να αποθηκεύσω με τον τρόπο με τον οποίο τοποθετούμε τις συσκευές με την βοήθεια δύο πινάκων. Ο ένας πίνακας κρατάει το ποιά είσοδος χρησιμοποείται σαν πρώτο port της κάθε συσκευής, και αντίστοιχα ο άλλος κρατάει το ποιά είσοδος χρησιμοποιείται σαν δεύτερο. Προφανώς οι πίνακες αυτοί θα έχουν τόσες θέσεις όσες είναι και ο αριθμός των συσκευών που θα έχουμε σε κάθε πρόβλημα. Επίσης, οι δύο πίνακες πρέπει να έχουν διαφορετικά στοιχεία μεταξύ τους στην κάθε θέση, καθώς για κάθε συσκευή θα πρέπει να οριστεί ένα μόνο port εισόδου και ένα port εξόδου. Αφού ορίσουμε αυτούς τους πίνακες, είμαστε σε θέση να υπολογίσουμε τον συνολικό μήκος των καλωδίων που θα χρειαστούμε. Στα σχόλια του κώδικα παρατηρούμε ότι στα αγγλικά μου ήταν πιο δύσκόλο να το περιγράψω, οπότε στα ελληνικά θα εξηγηθεί καλύτερα. Πρώτα από όλα αφού για το πρώτο port της πρώτης συσκευής το συνολικό μήκος θα είναι το πόσο απέχει από το έδαφος, αποφάσισα απλά να το χρησιμοποιήσω αυτούσιο. Στην συνέχεια, χρησιμοποιώ την εντολή forall για να κάνω μία επανάληψη για κάθε συσκευή, πλην της τελευταίας, αφού δεν θα συνδέεται κάπου. Ουσιαστικά για να βρω το πόση απόσταση απέχουν δύο port που συνδέονται, χρησιμοποιώ την απόλυτη τιμή της υψομετρικής διαφοράς μεταξύ του πρώτου port μίας συσκευής, από αυτή του δεύτερου porttης προηγούμενης της. Τέλος, αναφέρω ότι πρέπει να ελαχιστοποιήσουμε την συνολική απόσταση. Για αυτό το πρόβλημα μας δινόταν 3 αρχεία με πιθανές τιμές για τις παραμέτρους, εγώ θα χρησιμοποιήσω τα 2 για να δείξω τα αποτελέσματα του παραπάνω κώδικα, σε συνδυασμό με αυτόν που μας δινόταν. Για το αρχείο με τις παραμέτρους Dataflow1.dzn το πρόγραμμα κατάφερε μετά από δύο επαναλήψεις να βρει την ελάχιστη απόσταση η οποία ισούται με 40:

total_length =40;
———-
total_length =30;
———-
==========

Για το αρχείο με τις παραμέτρους Dataflow3.dzn το πρόγραμμα κατάφερε μετά από 5 επαναλήψεις να βρει την ελάχιστη απόσταση η οποία ισούται με 145.

total_length =235;
———-
total_length =205;
———-
total_length =195;

———-
total_length =165;
———-
total_length =145;
———-
==========
Δεν παρατήρησα κάποιο bug ούτε σε αυτόν τον κώδικα. Γενικά για να ελέξγω αν το πρόγραμμα βγάζει σωστά αποτελέσματα είχα βάλει στο output να εκτυπώνεται και τα αποτελέσματα των πινάκων που αποθηκεύουν το ποια port είναι πρώτα και ποιά δεύτερα, και στην συνέχεια έκανα και μόνος μου τους υπολογισμούς.

3 NuclearPlants

Στο συγκεκριμένο πρόβλημα έπρεπε να βρούμε τον ελάχιστο χρόνο που θα μπορούσαμε να τροφοδοτήσουμε με νερό τους σταθμούς παραγωγής που μας δινόταν, ικανοποιώντας ταυτόχρονα και τον περιορισμό που έχουν κάποιοι σταθμοι, όσον αφορά την ταυτόχρονη τροφοδότηση τους. Ο κώδικας ο οποίος χρησιμοποίησα είναι ο παρακάτω:

```
%Creating the necessary arrays for the problem
array[PLANTS] of var int: durations;
array[PLANTS] of var TIME: start;
array[PLANTS] of var TIME: end;

%The variable we want to minimize
var int: end_of_supply;

int: total_supply = 120;

%The duration array will be the result that comes from deviding an element of needs with the same element from supply,
% because, by the time we start providing water supply in a factory, we can't stop.
constraint forall(m in PLANTS)(durations[m] = needs[m]/supply[m]);

% Creating the 'end' array
constraint forall(m in PLANTS) (end[m] = start[m] + durations[m]);

%We must use the cumulative function, as long as there is a max cap of water flow we want to provide simultaneously
constraint cumulative(start,durations,supply,total_supply);

%We must use the disjunctive function because we cant provide with water, the factories that are presented in a row of the array not_sim.
%With this constraint we iterate through every row of not_sim, to denote that we can't have simultaneous flows, using array notation to pick the correct elements from every row.
constraint forall (m in PROBLEMS )(disjunctive([start[not_sim[m,j]] | j in 1..2],[durations[not_sim[m,j]] | j in 1..2]));

% Finally, we create the variable end_of_supply which is the max cxompletition time and we want to minimize it
constraint end_of_supply = max(end);
solve minimize end_of_supply;
```

Τα σχόλια του κώδικα φαίνονται παραπάνω. Αρχικά δημιουργούμε τους απαραίτητους πίνακες για τα προβλήματα χρονοπρογραμματισμού που υποδηλώνουν το πότε ξεκίνησε, το πότε τελείωσε και την διάρκεια της τροφοδοσίας του κάθε πυρηνικού σταθμού. Στην συνέχεια δημιουργούμε την μεταβλητή που θέλουμε να ελαχιστοποιήσουμε, και αναφέρουμε ότι η μέγιστη παροχή νερού που μπορεί να δοθεί είναι 120 κμ/λεπτό. Αφού δημιουργήσαμε τις μεταβλητές γράφουμε τους περιορισμούς.
Αρχικά, η διάρκεια παροχής του κάθε σταθμού θα είναι το ποσό χωρητικότητας του διά το μέγιστο πο-

σό παροχής που μπορεί να πάρει σε ένα λεπτό. Επίσης, όπως ξέρουμε οι πίνακες start, end και duration συνδέονται από την σχέση τέλος = αρχή + διάρκεια για κάθε σταθμό. Στην συνέχεια, λέμε ότι πρέπει να ικανοποιείται ο περιορισμός cumulative καθώς δεν μπορούμε να υπερβούμε τα 120 κμ/λεπτό, και ο περιορισμός disjunctive , ώστε να πούμε ότι οι σταθμοί που βρίσκονται στην ίδια γραμμή του πίνακα not_sim δεν γίνονται να λαμβάνουν ταυτόχρονα νερό. Για να το καταφέρουμε αυτό, κάνουμε μία επανάληψη για όλες τις γραμμές του και λέμε ότι δεν πρέπει να επικαλύπτονται οι χρόνοι παροχής των δύο σταθμών για κάθε γραμμή του πίνακα. Τέλος, αναφέρουμε ότι η μεταβλητή που ψάχνουμε να ελαχιστοποιήσουμε είναι το end_of_supply το οποίο όπως ξέρουμε ορίζεται ως το μέγιστο του χρόνου ολοκλήρωσης των εργασιών μας.
Για αυτό το πρόβλημα μας δινόταν 2 αρχεία με πιθανές τιμές για τις παραμέτρους.

Για το αρχείο με τις παραμέτρους plants1.dzn το πρόγραμμα κατάφερε μετά από δύο επαναλήψεις να βρει τον ελάχιστο χρόνο ολοκήρωσης, ο οποίος ισούται με 10:

starts = [3, 0, 0, 8, 5];
end_of_supply = 12;
———-
starts = [3, 0, 0, 3, 7];
end_of_supply = 10;
———-
==========

Για το αρχείο με τις παραμέτρους plants2.dzn το πρόγραμμα κατάφερε μετά από τέσσερις επαναλήψεις να βρει τον ελάχιστο χρόνο ολοκήρωσης, ο οποίος ισούται με 16 (αναγράφονται οι δύο τελευταίες):

starts = [7, 3, 12, 3, 0, 9, 12, 5, 0];
end_of_supply = 18;
———-
starts = [7, 3, 9, 3, 0, 13, 7, 12, 0];
end_of_supply = 16;
———-
==========
Δεν παρατήρησα κάποιο bug ούτε σε αυτόν τον κώδικα. Στην αρχή δεν μπορούσα να βγάλω αποτέλεσμα με την συνάρτηση disjunctive και χρησιμοποίησα τις εντολές που αναγράφονται στο πρώτο παράδειγμα των διαφανειών:
constraint forall(o in PROBLEMS) (end[not_sim[o,1]] <= start[not_sim[o,2]] \/end[not_sim[o,2]] <= start[not_sim[o,1]]);
, όμως στην συνέχεια κατάφερα να το βγάλω με τον περιορισμό disjunctive .