Bon travail, votre code aurait toutefois été plus élégant en définissant des prédicats génériques de manipulation de vecteurs (produit scalaire, multiplication) et en utilisant ces prédicats pour poser les contraintes.
NB : l'exercice intermédiaire sur le branch & bound est à reprendre.

# Rapport Travaux Pratiques : Programmation par Contraintes - TP 5 : **Contraindre puis chercher**

Nicolas Desfeux
Aurélien Texier

4 avril 2011

Dans ce TP, nous allons chercher à résoudre un problème de gestion de production. Pour cela, nous allons utiliser la programmation par contraintes.

## 1 Le problème

Le problème que nous allons résoudre concerne une unité de production pour différentes gammes de téléphones mobiles.
L'objectif est de trouver quels fabrications il faut lancer pour obtenir un bénéfice maximum.
Voici les données qui nous sont fournies :
Pour chaque type gamme, nous avons :

- Le nombre de techniciens de la gamme,
- La quantité de téléphone que l'on peut produire par jour,
- Le bénéfice obtenu.

Bonne intro

## 2 Modéliser et contraindre

**Question 5.1** On définit ici différents prédicats qui créent l'ensembles des données et des variables du problème.

Listing 1 – "Définition des 3 vecteurs de valeurs et du vecteur de variables"

```
1 getData(NbTechniciens,NbSortes,Techniciens,Quantite,Benefice) :-
2         NbTechniciens = 22,
3         NbSortes = 9,
4         Techniciens = [](5,7,2,6,9,3,7,5,3),
5         Quantite = [](140,130,60,95,70,85,100,30,45),
6         Benefice = [](4,5,8,5,6,4,7,10,11).
7
8 defineVars(Fabriquer,NbSortes):-
```

```
9           dim(Fabriquer,[NbSortes]),
10          (for(I,1,NbSortes),param(Fabriquer)
11          do
12                  Fabriquer[I] #:: 0..1
13          ).
14
15  getVarList(Fabriquer,NbSortes,L) :-
16          (for(Ind,1,NbSortes),fromto([],In,Out,L),param(Fabriquer)
17                  do
18                          Var #= Fabriquer[Ind],
19                          append(In,[Var],Out)
20          ).
```

OK

OK

**Question 5.2** Le code suivant permet de définir les prédicats permettant d'exprimer :

- Le nombre toal d'ouvriers nécéssaire,
- Le vecteur de bénéfice total par sorte de téléphones,
- Le profit total.

Listing 2 – "Différents prédicats utiles à la résolution du problème"

```
1   nbOuvriersNecessaires(Fabriquer,Techniciens,NbSortes,NbOuvriersTotal) :-
2           (for(J,1,NbSortes),fromto(0,In,Out,NbOuvriersTotal),param(Fabriquer,
                Techniciens)
3                   do
4                           Var is Fabriquer[J],
5                           NbTech is Techniciens[J],
6                           Out #= In + NbTech * Var
7           ).
8
9   vectBeneficeTotal(Fabriquer,Benefice,Quantite,NbSortes,VectBenefTotal) :-
10          (for(K,1,NbSortes),fromto([],In,Out,VectBenefTotal),param(Fabriquer,
                Benefice,Quantite)
11                  do
12                          Var is Fabriquer[K],
13                          Benef is Benefice[K],
14                          Quan is Quantite[K],
15                          Elem #= Var * Benef * Quan,
16                          append(In,[Elem],Out)
17          ).
18
19  profitTotal(Fabriquer,Benefice,Quantite,NbSortes,ProfitTotal) :-
20          vectBeneficeTotal(Fabriquer,Benefice,Quantite,NbSortes,VectBenefTotal
                ),
21          ProfitTotal #= sum(VectBenefTotal).
```

OK

OK

**Question 5.3** On a défini le prédicat pose_contraintes comme une succesion de prédicats. Contrairement aux T.P. précédents, nous n'avons pas créé de prédicat solve, qui aurait appelé (entre autre), pose_contrainte.

Listing 3 – "Prédicat **pose_contraintes**"

```
1   pose_contraintes(Fabriquer,NbTechniciensTotal,Profit) :-
```

En fait c'est ce prédicat que vous auriez pu appelé 'solve'...

```
2           getData(NbTech,NbSor,Tech,Quan,Bene),
3           defineVars(Fabriquer,NbSor),
4           nbOuvriersNecessaires(Fabriquer,Tech,NbSor,NbTechniciensTotal),
5           NbTechniciensTotal #=< NbTech,
6           getVarList(Fabriquer,NbSor,L),
7           profitTotal(Fabriquer,Bene,Quan,NbSor,Profit),
8           labeling(L).
```

OK

# 3 Optimiser

## 3.1 Branch and bound dans ECLiPSe

**Question 5.4** Il faut toujours faire un labeling dans le but, car il faut que les variables aient ete instanciees.

plus que ça : il faut faire un labeling sur TOUTES les variables du problème pour que la recherche de l'optimal se fasse uniquement sur des solutions

Listing 4 – Prédicat **test** avec minimize

```
1  test(X,Y,Z,W) :-
2           [X,Y,Z,W] #:: [0..10],
3           X #= Z+Y+2*W,
4           X#\= Z+Y+W.
5
6  /* Test
7
8  [eclipse 53]: minimize(test(X,Y,Z,W),X).
9  bb_min: search did not instantiate cost variable
10 Aborting execution ...
11 Abort
12
13 Avec le labeling sur [X,Y,Z,W], cela fonctionne, Prolog trouve les reponses
14
15 [eclipse 57]: test(X,Y,Z,W),labeling([X,Y,Z,W]).
16
17 X = 2
18 Y = 0
19 Z = 0
20 W = 1
21 Yes (0.00s cpu, solution 1, maybe more) ? ;
22
23 X = 3
24 Y = 0
25 Z = 1
26 W = 1
27 Yes (0.00s cpu, solution 2, maybe more) ? ;
28 */
```

Il vous était demandé de tester avec labeling([X]) puis de comparer avec labeling([X,Y,Z,W])...

## 3.2 Application à notre problème

**Question 5.5** Pour trouver le profit maximum, on crée une variable que l'on cherche à minimaliser, et l'on pose comme contrainte qu'elle soit égale à l'opposé du profit. On obtient ainsi le profit maximum.

Listing 5 – "Prédicat **pose_contrainte** avec maximisation du profit"

```
/* Test

[eclipse 71]: P2#=P*(-1), minimize(pose_contraintes(F,N,P),P2).
Found a solution with cost 0
Found a solution with cost -495
Found a solution with cost -795
Found a solution with cost -1195
Found a solution with cost -1495
Found a solution with cost -1535
Found a solution with cost -1835
Found a solution with cost -1955
Found a solution with cost -1970
Found a solution with cost -2010
Found a solution with cost -2015
Found a solution with cost -2315
Found a solution with cost -2490
Found a solution with cost -2665
Found no solution with cost -1.0Inf .. -2666

P2 = -2665
P = 2665
F = [](0, 1, 1, 0, 0, 1, 1, 0, 1)
N = 22
Yes (0.00s cpu)
*/
```

Bien

Nous avons choisi d'appeler le minimize lors de l'appel de la fonction principale.

**Question 5.6** Pour cette question, il nous est demandé de changer la politique de l'entreprise. On choisit de créer un seuil pour le profit, et minimiser le nombre d'employés.

Listing 6 – "Prédicat **pose_contrainte2** avec seuil pour le profit et minimisation du nombre d'ouvriers"

```
pose_contraintes2(Fabriquer,NbTechniciensTotal,Profit) :-
        getData(NbTech,NbSor,Tech,Quan,Bene),
        defineVars(Fabriquer,NbSor),
        nbOuvriersNecessaires(Fabriquer,Tech,NbSor,NbTechniciensTotal),
        NbTechniciensTotal #=< NbTech,
        getVarList(Fabriquer,NbSor,L),
        profitTotal(Fabriquer,Bene,Quan,NbSor,Profit),
        Profit #>=1000,
        labeling(L).

/* Tests
[eclipse 74]: minimize(pose_contraintes2(F,N,P),N).
Found a solution with cost 10
Found a solution with cost 9
Found a solution with cost 8
Found a solution with cost 7
Found no solution with cost -1.0Inf .. 6
```

OK

```
19 F = [](1, 0, 1, 0, 0, 0, 0, 0, 0)
20 N = 7
21 P = 1040
22 Yes (0.00s cpu)
23 */
```

Pour obtenir ce résultat, nous avons juste rajouter une contrainte pour le seuil du profit, et une recherche du minimum pour le nombre d'ouvrier.

# 4 Code Complet, avec l'ensemble des tests

Listing 7 – "TP5"

```
1  %TP5
2
3  :- lib(ic).
4  :- lib(branch_and_bound).
5
6  getData(NbTechniciens,NbSortes,Techniciens,Quantite,Benefice) :-
7          NbTechniciens = 22,
8          NbSortes = 9,
9          Techniciens = [](5,7,2,6,9,3,7,5,3),
10         Quantite = [](140,130,60,95,70,85,100,30,45),
11         Benefice = [](4,5,8,5,6,4,7,10,11).
12
13 defineVars(Fabriquer,NbSortes):-
14         dim(Fabriquer,[NbSortes]),
15         (for(I,1,NbSortes),param(Fabriquer)
16         do
17                 Fabriquer[I] #:: 0..1
18         ).
19
20 getVarList(Fabriquer,NbSortes,L) :-
21         (for(Ind,1,NbSortes),fromto([],In,Out,L),param(Fabriquer)
22                 do
23                         Var #= Fabriquer[Ind],
24                         append(In,[Var],Out)
25         ).
26
27 nbOuvriersNecessaires(Fabriquer,Techniciens,NbSortes,NbOuvriersTotal) :-
28         (for(J,1,NbSortes),fromto(0,In,Out,NbOuvriersTotal),param(Fabriquer,
               Techniciens)
29                 do
30                         Var is Fabriquer[J],
31                         NbTech is Techniciens[J],
32                         Out #= In + NbTech * Var
33         ).
34
35 vectBeneficeTotal(Fabriquer,Benefice,Quantite,NbSortes,VectBenefTotal) :-
36         (for(K,1,NbSortes),fromto([],In,Out,VectBenefTotal),param(Fabriquer,
               Benefice,Quantite)
37                 do
38                         Var is Fabriquer[K],
39                         Benef is Benefice[K],
40                         Quan is Quantite[K],
41                         Elem #= Var * Benef * Quan,
42                         append(In,[Elem],Out)
43         ).
44
45 profitTotal(Fabriquer,Benefice,Quantite,NbSortes,ProfitTotal) :-
46         vectBeneficeTotal(Fabriquer,Benefice,Quantite,NbSortes,VectBenefTotal
               ),
```

```
47          ProfitTotal #= sum(VectBenefTotal).
48
49  pose_contraintes(Fabriquer,NbTechniciensTotal,Profit) :-
50          getData(NbTech,NbSor,Tech,Quan,Bene),
51          defineVars(Fabriquer,NbSor),
52          nbOuvriersNecessaires(Fabriquer,Tech,NbSor,NbTechniciensTotal),
53          NbTechniciensTotal #=< NbTech,
54          getVarList(Fabriquer,NbSor,L),
55          profitTotal(Fabriquer,Bene,Quan,NbSor,Profit),
56          labeling(L).
57
58  test(X,Y,Z,W) :-
59          [X,Y,Z,W] #:: [0..10],
60          X #= Z+Y+2*W,
61          X#\= Z+Y+W.
62
63  /* Question 5.4
64  Il faut toujours faire un labeling dans le but,
65  car il faut que les variables aient ete instanciees.
66
67  [eclipse 53]: minimize(test(X,Y,Z,W),X).
68  bb_min: search did not instantiate cost variable
69  Aborting execution ...
70  Abort
71
72  Avec le labeling sur [X,Y,Z,W], cela fonctionne, Prolog trouve les reponses
73
74  [eclipse 57]: test(X,Y,Z,W),labeling([X,Y,Z,W]).
75
76  X = 2
77  Y = 0
78  Z = 0
79  W = 1
80  Yes (0.00s cpu, solution 1, maybe more) ? ;
81
82  X = 3
83  Y = 0
84  Z = 1
85  W = 1
86  Yes (0.00s cpu, solution 2, maybe more) ? ;
87  */
88
89  /* Test
90
91  [eclipse 71]: P2#=P*(-1), minimize(pose_contraintes(F,N,P),P2).
92  Found a solution with cost 0
93  Found a solution with cost -495
94  Found a solution with cost -795
95  Found a solution with cost -1195
96  Found a solution with cost -1495
97  Found a solution with cost -1535
98  Found a solution with cost -1835
99  Found a solution with cost -1955
```

```
Found a solution with cost -1970
Found a solution with cost -2010
Found a solution with cost -2015
Found a solution with cost -2315
Found a solution with cost -2490
Found a solution with cost -2665
Found no solution with cost -1.0Inf .. -2666

P2 = -2665
P = 2665
F = [](0, 1, 1, 0, 0, 1, 1, 0, 1)
N = 22
Yes (0.00s cpu)
*/

% Question 5.6

pose_contraintes2(Fabriquer,NbTechniciensTotal,Profit) :-
        getData(NbTech,NbSor,Tech,Quan,Bene),
        defineVars(Fabriquer,NbSor),
        nbOuvriersNecessaires(Fabriquer,Tech,NbSor,NbTechniciensTotal),
        NbTechniciensTotal #=< NbTech,
        getVarList(Fabriquer,NbSor,L),
        profitTotal(Fabriquer,Bene,Quan,NbSor,Profit),
        Profit #>=1000,
        labeling(L).

/* Tests
[eclipse 74]: minimize(pose_contraintes2(F,N,P),N).
Found a solution with cost 10
Found a solution with cost 9
Found a solution with cost 8
Found a solution with cost 7
Found no solution with cost -1.0Inf .. 6

F = [](1, 0, 1, 0, 0, 0, 0, 0, 0)
N = 7
P = 1040
Yes (0.00s cpu)
*/
```