CR TP5 Programmation par contraintes

Contraindre puis chercher

Julien LETOILE, Romain HUBERT

le 10/03/2021

Table des matières

Table des matières

- I. Réponses rédigées
  - Question 5.1
  - Question 5.2
  - Question 5.3
  - Question 5.4
  - Question 5.5
  - Question 5.6
- II. Annexes
  - Code source

I. Réponses rédigées

Question 5.1

```
/* Contraint avec les données de l'énoncé */
/* Vecteurs de valeurs (-, -, -) */
getData(Techniciens, Quantite, Benefices):-
    Techniciens = [](5, 7, 2, 6, 9, 3, 7, 5, 3),
    Quantite = [](140, 130, 60, 95, 70, 85, 100, 30, 45),
    Benefices = [](4, 5, 8, 5, 6, 4, 7, 10, 11).

/* Vecteur de variables (-, +) */
getVar(Fabriquer, Longueur):-
    dim(Fabriquer, [Longueur]),
    Fabriquer #:: 0..1.
```

Test:

> getData(T, Q, B).
>
> T = [](5, 7, 2, 6, 9, 3, 7, 5, 3)
> Q = [](140, 130, 60, 95, 70, 85, 100, 30, 45)
> B = [](4, 5, 8, 5, 6, 4, 7, 10, 11)
> Yes (0.00s cpu)

Question 5.2

```
/* Prédicat réalisant le produit de deux vecteurs (+, +, -) */
produitVecteur(Vect1, Vect2, Res):-
    dim(Vect1, [Longueur]),
    dim(Vect2, [Longueur]),
    dim(Res, [Longueur]),
    (
        for(I, 1, Longueur),
```

```
8           param(Vect1, Vect2, Res)
9       do
10          Res[I] #= Vect1[I] * Vect2[I]
11      ).
12
13  /* Prédicat réalisant le produit scalaire de deux vecteurs (+,
    +, -) */
14  produitScalaire(Vect1, Vect2, Res):-
15      produitVecteur(Vect1, Vect2, Vect3),
16      sommeVecteur(Vect3, Res).
17
18  /* Prédicat réalisant la somme de toutes les composantes d'un
    vecteur (+, -) */
19  sommeVecteur(Vect, Res):-
20      (
21          foreachelem(X, Vect),
22          fromto(0, AncienneValeur, NouvelleValeur, Res)
23      do
24          NouvelleValeur #= AncienneValeur + X
25      ).
```

Test:

```
[eclipse 12]: X = [](1, 2, 3), Y=[](4, 5, 6), produitVecteur(X, Y, Z).
X = [](1, 2, 3)
Y = [](4, 5, 6)
Z = [](4, 10, 18)
Yes (0.00s cpu)

[eclipse 41]: X = [](1, 2, 3), sommeVecteur(X, R).

X = [](1, 2, 3)
R = 6
Yes (0.00s cpu)

[eclipse 14]: X = [](1, 2, 3), Y=[](4, 5, 6), produitScalaire(X, Y, Z).
X = [](1, 2, 3)
Y = [](4, 5, 6)
Z = 32
Yes (0.00s cpu)
```

```
/* Indique le nombre total d'ouvriers nécessaire (+, +, -) */
totalOuvriers(Techniciens, Fabriquer, TotalOuvriers):-
    produitScalaire(Techniciens, Fabriquer, TotalOuvriers).

/* Indique le bénéfice total pour chaque sorte de téléphone (+, +, +, -) */
totalBenefice(Quantite, Benefices, Fabriquer, TotalBenefice):-
    produitVecteur(Quantite, Benefices, Temp),
    produitVecteur(Temp, Fabriquer, TotalBenefice).

/* Indique le profit total (+, -) */
totalProfit(TotalBenefice, TotalProfit):-
    sommeVecteur(TotalBenefice, TotalProfit).
```

Question 5.3

```
/* Equivalent d'un solve : pose les contraintes (?, ?, ?) */
poseContraintes(Fabriquer, NbTechniciensTotal, Profit):-
    getData(Techniciens, Quantite, Benefices),
    dim(Techniciens, [Longueur]),
    getVar(Fabriquer, Longueur),
    totalOuvriers(Techniciens, Fabriquer, NbTechniciensTotal),
    totalBenefice(Quantite, Benefices, Fabriquer, TotalBenefice),
    totalProfit(TotalBenefice, Profit),
    NbTechniciensTotal #=< 22.

getVarlist(Fabriquer, Liste):-
    term_variables(Fabriquer, Liste).
```

Test:

```
Fabriquer = [](0, 0, 0, 0, 0, 0, 0, 0, 0)
NbTechniciensTotal = 0
Profit = 0
Yes (0.02s cpu, solution 1, maybe more) ? ;

Fabriquer = [](0, 0, 0, 0, 0, 0, 0, 0, 1)
NbTechniciensTotal = 3
Profit = 495
Yes (0.02s cpu, solution 2, maybe more) ? ;

Fabriquer = [](0, 0, 0, 0, 0, 0, 0, 1, 0)
NbTechniciensTotal = 5
Profit = 300
Yes (0.02s cpu, solution 3, maybe more) ?

=> On ne trouve pas forcément le maximum
```

Question 5.4

```
1  /* Calcule la plus petite valeur X répondant aux contraintes à
   l'aide du Branch and Bound (-, -, -, -). Ici le labeling est sur
   [X, Y, Z, W] (2eme cas) */
2  minimum(X, Y, Z, W):-
3      [X, Y, Z, W] #::[0..10],
4      X #= Z+Y+2*W,
5      X #\= Z+Y+W,
6      minimize(labeling([X, Y, Z, W]), X).
```

Test:

```
labeling sur le X
Found a solution with cost 1
Found no solution with cost 0.0 .. 0.0
```

```
X = 1
Y = Y{[0, 1]}
Z = Z{[0, 1]}
W = 0
Min = Min

Delayed goals:
      - Z{[0, 1]} - Y{[0, 1]} #= -1
            - Y{[0, 1]} - Z{[0, 1]} - 0 + 1 #\= 0
Yes (0.00s cpu)

Labeling sur [X, Y, Z, W]
Found a solution with cost 2
Found no solution with cost 0.0 .. 1.0

X = 2
Y = 0
Z = 0
W = 1
```

Dans le premier cas, nous avons des "Delayed goals", c'est à dire que Prolog ne les as pas encore calculés. Ainsi, le résultat final n'est pas totalement calculé, donc il n'est pas consistent.

Pour éviter cela, il faut mettre tous les paramêtres pour être sûr qu'ils soient calculés. Après, si nous savons que certains paramêtres découlent d'autres paramêtres (comme le W dans ce cas), on n'est pas obligé de les mettre dans le labeling.

Question 5.5

```
1  /* Application des contraintes le utiisant le branch and bound
   (?, ?, ?) */
2  solve(Fabriquer, NbTechniciensTotal, Profit):-
3      poseContraintes(Fabriquer, NbTechniciensTotal, Profit),
4      X #= - Profit,
5      minimize(labeling(Fabriquer), X).
```

Test:

```
solve(Fabriquer, NbTechniciensTotal, Profit).

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 -4420.0 .. -2666.0

Fabriquer = [](0, 1, 1, 0, 0, 1, 1, 0, 1)
NbTechniciensTotal = 22
Profit = 2665
Yes (0.02s cpu)
```

Ainsi, on peut réaliser un profit maximum de 2665€ si on lance les fabrications de S2, S3, S6, S7 et S9.

Question 5.6

```
/* Application des contraintes le utiisant le branch and bound,
en minimisant le nombre de techniciens tout en gardant un revenu
suérieur à 1000 (?, ?, ?) */
solve1000(Fabriquer, NbTechniciensTotal, Profit):-
    poseContraintes1000(Fabriquer, NbTechniciensTotal, Profit),
    minimize(labeling(Fabriquer), NbTechniciensTotal).

poseContraintes1000(Fabriquer, NbTechniciensTotal, Profit):-
    getData(Techniciens, Quantite, Benefices),
    dim(Techniciens, [Longueur]),
    getVar(Fabriquer, Longueur),
```

```
10      totalOuvriers(Techniciens, Fabriquer, NbTechniciensTotal),
11      totalBenefice(Quantite, Benefices, Fabriquer,
    TotalBenefice),
12      totalProfit(TotalBenefice, Profit),
13      NbTechniciensTotal #=< 22,
14      Profit #>= 1000.
```

Test:

```
solve1000(Fabriquer, NbTechniciensTotal, Profit).

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 0.0 .. 6.0

Fabriquer = [](1, 0, 1, 0, 0, 0, 0, 0, 0)
NbTechniciensTotal = 7
Profit = 1040
Yes (0.01s cpu)
```

Si on lance la fabrication de S1 et S3, seuls 7 techniciens travaillent mais on réalise un profit d'au moins 1000€.

II. Annexes

Code source

```
/*

faire produitVecteur
faire sommeVecteur
faire produitScalaire

(foreacharg ?) ou foreachelem
*/

:-lib(ic).
:-lib(ic_symbolic).
:-lib(branch_and_bound).

/* Question 5.5 */

solve(Fabriquer, NbTechniciensTotal, Profit):-
    poseContraintes(Fabriquer, NbTechniciensTotal, Profit),
    X #= - Profit,
    minimize(labeling(Fabriquer), X).

/*
[eclipse 7]: solve(Fabriquer, NbTechniciensTotal, Profit).
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 -4420.0 .. -2666.0

Fabriquer = [](0, 1, 1, 0, 0, 1, 1, 0, 1)
NbTechniciensTotal = 22
Profit = 2665
Yes (0.02s cpu)
*/

/* Question 5.6 */

solve1000(Fabriquer, NbTechniciensTotal, Profit):-
    poseContraintes1000(Fabriquer, NbTechniciensTotal, Profit),
    minimize(labeling(Fabriquer), NbTechniciensTotal).

poseContraintes1000(Fabriquer, NbTechniciensTotal, Profit):-
    getData(Techniciens, Quantite, Benefices),
    dim(Techniciens, [Longueur]),
    getVar(Fabriquer, Longueur),
    totalOuvriers(Techniciens, Fabriquer, NbTechniciensTotal),
    totalBenefice(Quantite, Benefices, Fabriquer,
TotalBenefice),
    totalProfit(TotalBenefice, Profit),
    NbTechniciensTotal #=< 22,
    Profit #>= 1000.

/*
solve1000(Fabriquer, NbTechniciensTotal, Profit).

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 0.0 .. 6.0

Fabriquer = [](1, 0, 1, 0, 0, 0, 0, 0, 0)
NbTechniciensTotal = 7
Profit = 1040
Yes (0.01s cpu)
*/
```

```
75
76  /* Question 5.3 */
77
78  poseContraintes(Fabriquer, NbTechniciensTotal, Profit):-
79      getData(Techniciens, Quantite, Benefices),
80      dim(Techniciens, [Longueur]),
81      getVar(Fabriquer, Longueur),
82      totalOuvriers(Techniciens, Fabriquer, NbTechniciensTotal),
83      totalBenefice(Quantite, Benefices, Fabriquer,
    TotalBenefice),
84      totalProfit(TotalBenefice, Profit),
85      NbTechniciensTotal #=< 22.
86
87  /*
88
89  5.3:
90
91  Fabriquer = [](0, 0, 0, 0, 0, 0, 0, 0, 0)
92  NbTechniciensTotal = 0
93  Profit = 0
94  Yes (0.02s cpu, solution 1, maybe more) ? ;
95
96  Fabriquer = [](0, 0, 0, 0, 0, 0, 0, 0, 1)
97  NbTechniciensTotal = 3
98  Profit = 495
99  Yes (0.02s cpu, solution 2, maybe more) ? ;
00
01  Fabriquer = [](0, 0, 0, 0, 0, 0, 0, 1, 0)
02  NbTechniciensTotal = 5
03  Profit = 300
04  Yes (0.02s cpu, solution 3, maybe more) ?
05
06  => On ne trouve pas forcément le maximum
07
08  */
09
10  getVarlist(Fabriquer, Liste):-
11      term_variables(Fabriquer, Liste).
12
13  /* Question 5.1 */
14
```

```
15  getData(Techniciens, Quantite, Benefices):-
16      Techniciens = [](5, 7, 2, 6, 9, 3, 7, 5, 3),
17      Quantite = [](140, 130, 60, 95, 70, 85, 100, 30, 45),
18      Benefices = [](4, 5, 8, 5, 6, 4, 7, 10, 11).
19
20
21  getVar(Fabriquer, Longueur):-
22      dim(Fabriquer, [Longueur]),
23      Fabriquer #:: 0..1.
24  /*
25  getData(T, Q, B).
26
27  T = [](5, 7, 2, 6, 9, 3, 7, 5, 3)
28  Q = [](140, 130, 60, 95, 70, 85, 100, 30, 45)
29  B = [](4, 5, 8, 5, 6, 4, 7, 10, 11)
30  Yes (0.00s cpu)
31  */
32
33  /* Question 5.2 */
34
35  produitVecteur(Vect1, Vect2, Res):-
36      dim(Vect1, [Longueur]),
37      dim(Vect2, [Longueur]),
38      dim(Res, [Longueur]),
39      (
40          for(I, 1, Longueur),
41          param(Vect1, Vect2, Res)
42      do
43          Res[I] #= Vect1[I] * Vect2[I]
44      ).
45
46
47  produitScalaire(Vect1, Vect2, Res):-
48      produitVecteur(Vect1, Vect2, Vect3),
49      sommeVecteur(Vect3, Res).
50
51  sommeVecteur(Vect, Res):-
52      (
53          foreachelem(X, Vect),
54          fromto(0, AncienneValeur, NouvelleValeur, Res)
55      do
```

```
        NouvelleValeur #= AncienneValeur + X
    ).



/*
[eclipse 12]: X = [](1, 2, 3), Y=[](4, 5, 6), produitVecteur(X,
Y, Z).
X = [](1, 2, 3)
Y = [](4, 5, 6)
Z = [](4, 10, 18)
Yes (0.00s cpu)

[eclipse 41]: X = [](1, 2, 3), sommeVecteur(X, R).

X = [](1, 2, 3)
R = 6
Yes (0.00s cpu)

[eclipse 14]: X = [](1, 2, 3), Y=[](4, 5, 6), produitScalaire(X,
Y, Z).
X = [](1, 2, 3)
Y = [](4, 5, 6)
Z = 32
Yes (0.00s cpu)
*/

totalOuvriers(Techniciens, Fabriquer, TotalOuvriers):-
    produitScalaire(Techniciens, Fabriquer, TotalOuvriers).

totalBenefice(Quantite, Benefices, Fabriquer, TotalBenefice):-
    produitVecteur(Quantite, Benefices, Temp),
    produitVecteur(Temp, Fabriquer, TotalBenefice).

% Multipliser Quantite x Benefices x Fabriquer

totalProfit(TotalBenefice, TotalProfit):-
    sommeVecteur(TotalBenefice, TotalProfit).



/* Question 5.4 */
```

```
95
96  minimum(X, Y, Z, W):-
97      [X, Y, Z, W] #::[0..10],
98      X #= Z+Y+2*W,
99      X #\= Z+Y+W,
00      minimize(labeling([X, Y, Z, W]), X).
01
02  /*
03  labeling sur le X
04  Found a solution with cost 1
05  Found no solution with cost 0.0 .. 0.0
06
07  X = 1
08  Y = Y{[0, 1]}
09  Z = Z{[0, 1]}
10  W = 0
11  Min = Min
12
13  Le résultat n'est pas totalement calculé, c'est insuffisant, non
    consistent
14  Delayed goals:
15          - Z{[0, 1]} - Y{[0, 1]} #= -1
16          - Y{[0, 1]} - Z{[0, 1]} - 0 + 1 #\= 0
17  Yes (0.00s cpu)
18
19  Labeling sur [X, Y, Z, W]
20  Found a solution with cost 2
21  Found no solution with cost 0.0 .. 1.0
22
23  X = 2
24  Y = 0
25  Z = 0
26  W = 1
27  */
28
```