

Rapport Travaux Pratiques : Programmation par Contraintes - TP 6 : **Sur une balançoire**

Nicolas Desfeux
Aurélien Texier

13 avril 2011

Dans ce TP, nous allons chercher à appliquer ce que nous avons appris à un problème de la vie réelle (aussi vraisemblable soit il !). Le problème modélise une balançoire à seize places (huit de chaque côté), qu'une famille de dix personnes veut utiliser. Il y a plusieurs règles à respecter pour cette balançoire :

- La balançoire doit être équilibré (calcul du moments des forces) ;
- La maman et le papa souhaite encadrer leurs enfants pour les surveiller ;
- Dan et Max doivent être assis chacun devant un parent.
- Il doit y avoir 5 personnes de chaque côté.

Table des matières

1	Trouver une solution au problème	2
2	Trouver la meilleure solution	5
3	Code Complet, avec l'ensemble des tests	9

1 Trouver une solution au problème

Question 6.1 Nous avons procédé comme pour les problèmes précédents pour résoudre celui ci. Vous retrouverez donc les étapes suivantes : définition des variables, définitions des domaines, pose des contraintes.

Listing 1 – "Solution 1"

```
1 %TP6
2
3 :- lib(ic).
4 :- lib(branch_and_bound).
5 :- lib(ic_global).
6
7 getData(Poids,NbPersonnes,NbPlaces) :-
8     NbPersonnes = 10,
9     NbPlaces = 16,
10    Poids = [(24,39,85,60,165,6,32,123,7,14)].
11
12 defineVars(Places,NbPlaces,NbPersonnes):-
13     dim(Places,[NbPersonnes]),
14     Borne #= NbPlaces/2,
15     ( for(I,1,NbPersonnes),param(Places,Borne)
16         do
17             Places[I] #:: -Borne..Borne,
18             Places[I] #\= 0
19     ).
20
21 getVarList(Places,NbPersonnes,L) :-
22     ( for(Ind,1,NbPersonnes),fromto([],In,Out,L),param(Places)
23         do
24             Var #= Places[Ind],
25             append([Var],In,Out)
26     ).
27
28 balancoireEquilibree(Poids,NbPersonnes,Places) :-
29     ( for(J,1,NbPersonnes),fromto(0,In,Out,SommeMoments),param(Places,
30         Poids)
31         do
32             Distance #= Places[J],
33             Poids1 #= Poids[J],
34             Out #= In + Distance * Poids1
35     ),
36     SommeMoments #= 0.
37
38 pasMemesPlaces(Places) :-
39     ic:alldifferent(Places).
40
41 cinqChaqueCote(Places,NbPersonnes) :-
42     NbPersGauche #= NbPersonnes/2,
43     ( for(K,1,NbPersonnes),fromto(0,In,Out,NbPersonnesAGauche),param(
44         Places)
45         do
46             Var #= Places[K],
```

```

45             (Var #< 0) => (Out #= In + 1),
46             (Var #> 0) => (Out #= In)
47         ),
48         NbPersGauche #= NbPersonnesAGauche.
49
50 parentsEncadrentEnfants(Places) :-
51     PlaceMere #= Places[4],
52     PlacePere #= Places[8],
53     ((PlaceMere #= ic:max(Places)) and (PlacePere #= ic:min(Places)))
54     or
55     ((PlacePere #= ic:max(Places)) and (PlaceMere #= ic:min(Places))).
56 /*
57     (maxlist(Places, PlaceMere) and minlist(Places, PlacePere))
58     or
59     (maxlist(Places, PlacePere) and minlist(Places, PlaceMere)).
60 */
61
62 danEtMaxDevantParents(Places) :-
63     PlaceDan #= Places[6],
64     PlaceMax #= Places[9],
65     PlaceMere #= Places[4],
66     PlacePere #= Places[8],
67     (((PlaceMere #<0) and (PlaceDan #<0)) and ((PlaceMere #= PlaceDan -
68         1) and (PlacePere #= PlaceMax + 1)))
69     or
70     (((PlaceMere #<0) and (PlaceMax #<0)) and ((PlaceMere #= PlaceMax -
71         1) and (PlacePere #= PlaceDan + 1)))
72     or
73     (((PlacePere #<0) and (PlaceDan #<0)) and ((PlacePere #= PlaceDan -
74         1) and (PlaceMere #= PlaceMax + 1)))
75     or
76     (((PlacePere #<0) and (PlaceMax #<0)) and ((PlacePere #= PlaceMax -
77         1) and (PlaceMere #= PlaceDan + 1))).
78
79 solve(Places) :-
80     getData(Poids, NbPers, NbPlac),
81     defineVars(Places, NbPlac, NbPers),
82     pasMemesPlaces(Places),
83     balancoireEquilibree(Poids, NbPers, Places),
84     cinqChaqueCote(Places, NbPers),
85     parentsEncadrentEnfants(Places),
86     danEtMaxDevantParents(Places),
87     getVarList(Places, NbPers, L),
88     labeling(L).
89
90 /* Tests
91 [eclipse 29]: solve(P).
92
93 P = [ ](-6, -5, -4, -8, 2, 5, 3, 6, -7, 1)
94 Yes (0.13 s cpu, solution 1, maybe more) ? ;
95
96 P = [ ](-6, -5, -1, 8, 5, 7, 3, -8, -7, 1)
97 Yes (0.15 s cpu, solution 2, maybe more) ? ;

```

```

94
95 P = [](-6, -5, 1, -8, -2, 6, 5, 7, -7, 4)
96 Yes (0.02s cpu, solution 3, maybe more) ?
97 */

```

Question 6.2 Voici une solution possible :

```

1 P = [](-6, -5, -4, -8, 2, 5, 3, 6, -7, 1)
2 Yes (0.13s cpu, solution 1, maybe more) ? ;

```

Eclipse a besoin de 0.02s pour trouver ce premier résultat. Il y en a bien sur d'autres.

Question 6.3 Il existe une symétrie entre les solutions du problème. Une solution qui place 5 personnes à droite possède une symétrie si l'on met ces 5 personnes à gauche. Il n'est donc pas forcément utile de les faire rechercher par Eclipse !

Pour éliminer cette symétrie, nous avons choisi de créer une nouvelle contrainte

```

1 % Impose que la mere soit a gauche sur la balançoire
2 elimineSymetrie(Places) :-
3     Places[4] #< 0.

```

Et nous avons juste eu besoin d'adapter le prédicat solve.

Listing 2 – "Solution 2"

```

1 solve2(Places) :-
2     getData(Poids, NbPers, NbPlac),
3     defineVars(Places, NbPlac, NbPers),
4     elimineSymetrie(Places), %Gain de temps monstrueux
5     pasMemesPlaces(Places),
6     cinqChaqueCote(Places, NbPers),
7     balançoireEquilibree(Poids, NbPers, Places),
8     parentsEncadrentEnfants(Places),
9     danEtMaxDevantParents(Places),
10    getVarList(Places, NbPers, L),
11    labeling(L).
12
13 /* Tests
14 [eclipse 31]: solve2(P).
15
16 P = [](-6, -5, -4, -8, 2, 5, 3, 6, -7, 1)
17 Yes (0.02s cpu, solution 1, maybe more) ? ;
18
19 P = [](-6, -5, 1, -8, -2, 6, 5, 7, -7, 4)
20 Yes (0.04s cpu, solution 2, maybe more) ? ;
21
22 P = [](-6, -5, 3, -8, -3, -7, 4, 7, 6, 5)
23 Yes (0.01s cpu, solution 3, maybe more) ?
24 */

```

Le gain de temps est assez impressionnant. Avec la solution 1, il faut 0.13s à Eclipse pour trouver une solution, alors que pour la solution 2, 0.02s suffisent !

2 Trouver la meilleure solution

Question 6.4 On cherche maintenant à trouver la meilleur solution, c'est à dire un solution qui minimise les normes des moments des forces pour la balançoire. Pour cela, nous avons effectuer deux modifications dans notre code. Nous avons ajouté le calcul du moment des forces sur chque coté de la balançoire. Nous avons aussi modifier notre prédicat solve, afin qu'il permettent de rechercher le minimum des moments de forces que l'on vient de calculer.

Listing 3 – "Solution 3"

```
1  balancoireEquilibree2 (Poids , NbPersonnes , Places , SNormeMoments) :-
2      ( for (J, 1, NbPersonnes) , fromto (0, In , Out , SommeMoments) , fromto (0 , In2 ,
          Out2 , SNormeMoments) , param ( Places , Poids )
3          do
4              Distance #= Places [J] ,
5              Poids1 #= Poids [J] ,
6              Out #= In + Distance * Poids1 ,
7              ( Places [J] #> 0 ) => ( Out2 #= In2 + Distance * Poids1 )
8              ,
9              ( Places [J] #=< 0 ) => ( Out2 #= In2 )
10         ) ,
11         SommeMoments #= 0.
12 solve3 ( Places ) :-
13     getData ( Poids , NbPers , NbPlac ) ,
14     defineVars ( Places , NbPlac , NbPers ) ,
15     elimineSymetrie ( Places ) , %Gain de temps monstrueux
16     pasMemesPlaces ( Places ) ,
17     cinqChaqueCote ( Places , NbPers ) ,
18     balancoireEquilibree2 ( Poids , NbPers , Places , SNormeMom ) ,
19     parentsEncadrentEnfants ( Places ) ,
20     danEtMaxDevantParents ( Places ) ,
21     getVarList ( Places , NbPers , L ) ,
22     minimize ( labeling ( L ) , SNormeMom ) .
23
24 /* Tests
25 [eclipse 25]: solve3(P).
26 Found a solution with cost 874
27 Found a solution with cost 872
28 Found a solution with cost 802
29 Found no solution with cost -1.0Inf .. 801
30
31 P = [ ](-3, 1, -2, -6, -1, 4, 3, 5, -5, 2)
32 Yes (1.66 s cpu)
33 */
```

Dans le prédicat balancoireEquilibree2, on effectue le calcul de la norme du moment des forces sur un coté de la balançoire. Dans notre problème, il suffit de calculer un seul coté, puisque l'on impose que la balançoire soit équilibrée !

Utilisation du prédicat search/6

Dans cette partie, nous avons utiliser le prédicat search qui va nous permettre de contrôler l'énumération de Eclipse (en influent sur l'ordre dans lequel on instancie les variables, et sur l'ordre dans lequel chaque valeur du domaine d'une variable est testée).

Version 1 « Pour réussir, essaie d'abord là où tu as le plus de chances d'échouer ». Ici, les premières variables testée sont celles qui interviennent dans le plus de contraintes.

Listing 4 – "Version 1"

```
1 solve4(Places) :-
2     getData(Poids, NbPers, NbPlac),
3     defineVars(Places, NbPlac, NbPers),
4     elimineSymetrie(Places),
5     pasMemesPlaces(Places),
6     cinqChaqueCote(Places, NbPers),
7     balancoireEquilibree2(Poids, NbPers, Places, SNormeMom),
8     parentsEncadrentEnfants(Places),
9     danEtMaxDevantParents(Places),
10    getVarList(Places, NbPers, L),
11    minimize(search(L, 0, occurrence, indomain_min, complete, []), SNormeMom).
12
13 /* Tests
14 [eclipse 9]: solve4(P).
15 Found a solution with cost 953
16 Found a solution with cost 852
17 Found a solution with cost 834
18 Found a solution with cost 802
19 Found no solution with cost -1.0Inf .. 801
20
21 P = [ ](-3, 1, -2, -6, -1, 4, 3, 5, -5, 2)
22 Yes (2.50s cpu)
23 */
```

Ici, il semble que most_constrained soit plus adapte que occurrence, vu la vitesse de reponse (2.13s au lieu de 2.50s)

Version 2 Ici on vise a essayer de mettre les places les plus au centre en premier, parce que c'est dans ce cas que l'on minimise le mieux le moment total.

Listing 5 – "Version 2"

```
1 quick_sort([], []).
2 quick_sort([H|T], Sorted):-
3     pivoting(H, T, L1, L2), quick_sort(L1, Sorted1), quick_sort(L2, Sorted2),
4     append(Sorted1, [H|Sorted2], Sorted).
5
6 pivoting(_H, [], [], []).
7 pivoting(H, [X|T], [X|L], G):-abs(X)=<abs(H), pivoting(H, T, L, G), !.
8 pivoting(H, [X|T], L, [X|G]):-abs(X)>abs(H), pivoting(H, T, L, G), !.
9
10 /* Tests
```

```

11  [eclipse 50]: quick_sort([1,2,3,4,-1,-2,-3,-4],L).
12
13  L = [-1, 1, -2, 2, -3, 3, -4, 4]
14  Yes (0.00s cpu)
15  */
16
17  choice(X) :-
18      get_domain_as_list(X,L),
19      quick_sort(L,L2),
20      member(X,L2).
21
22  solve5(Places) :-
23      getData(Poids,NbPers,NbPlac),
24      defineVars(Places,NbPlac,NbPers),
25      elimineSymetrie(Places),
26      pasMemesPlaces(Places),
27      cinqChaqueCote(Places,NbPers),
28      balancoireEquilibree2(Poids,NbPers,Places,SNormeMom),
29      parentsEncadrentEnfants(Places),
30      danEtMaxDevantParents(Places),
31      getVarList(Places,NbPers,L),
32      minimize(search(L,0,input_order,choice,complete,[]),SNormeMom).
33
34  /* Tests
35  [eclipse 32]: solve5(P).
36  Found a solution with cost 1216
37  Found a solution with cost 956
38  Found a solution with cost 947
39  Found a solution with cost 917
40  Found a solution with cost 852
41  Found a solution with cost 802
42  Found no solution with cost -1.0Inf .. 801
43
44  P = [](-3, 1, -2, -6, -1, 4, 3, 5, -5, 2)
45  Yes (1.80s cpu)
46  */

```

Version 3 Cette version est une combinaison des versions précédentes.

Listing 6 – "Version 3"

```

1  solve6(Places) :-
2      getData(Poids,NbPers,NbPlac),
3      defineVars(Places,NbPlac,NbPers),
4      elimineSymetrie(Places),
5      pasMemesPlaces(Places),
6      cinqChaqueCote(Places,NbPers),
7      balancoireEquilibree2(Poids,NbPers,Places,SNormeMom),
8      parentsEncadrentEnfants(Places),
9      danEtMaxDevantParents(Places),
10     getVarList(Places,NbPers,L),
11     minimize(search(L,0,occurrence,choice,complete,[]),SNormeMom).
12

```

```

13  /* Tests
14  [eclipse 52]: solve6(P).
15  Found a solution with cost 933
16  Found a solution with cost 898
17  Found a solution with cost 872
18  Found a solution with cost 852
19  Found a solution with cost 848
20  Found a solution with cost 834
21  Found a solution with cost 802
22  Found no solution with cost  $-1.0Inf$  .. 801
23
24  P = [](-3, 1, -2, -6, -1, 4, 3, 5, -5, 2)
25  Yes (2.98 s cpu)
26  */

```

Le résultat n'est pas satisfaisant ici.

Version 4 Cette version est basé sur les heuristiques de choix de l'ordre des variables.

Listing 7 – "Version 4"

3 Code Complet, avec l'ensemble des tests

Listing 8 – "TP6"

```
1  %TP6
2
3  :- lib(ic).
4  :- lib(branch_and_bound).
5  :- lib(ic_global).
6
7  getData(Poids ,NbPersonnes ,NbPlaces) :-
8      NbPersonnes = 10,
9      NbPlaces = 16,
10     Poids = [(24,39,85,60,165,6,32,123,7,14)].
11
12 defineVars(Places ,NbPlaces ,NbPersonnes):-
13     dim(Places ,[NbPersonnes]),
14     Borne #= NbPlaces/2,
15     ( for(I,1,NbPersonnes),param(Places ,Borne)
16         do
17             Places[I] #:: -Borne..Borne ,
18             Places[I] #\= 0
19     ).
20
21 getVarList(Places ,NbPersonnes ,L) :-
22     ( for(Ind,1,NbPersonnes),fromto([],In ,Out ,L),param(Places)
23         do
24             Var #= Places[Ind],
25             append([Var],In ,Out)
26     ).
27
28 balancoireEquilibree(Poids ,NbPersonnes ,Places) :-
29     ( for(J,1,NbPersonnes),fromto(0,In ,Out ,SommeMoments),param(Places ,
30         Poids)
31         do
32             Distance #= Places[J],
33             Poids1 #= Poids[J],
34             Out #= In + Distance * Poids1
35     ),
36     SommeMoments #= 0.
37
38 pasMemesPlaces(Places) :-
39     ic:alldifferent(Places).
40
41 cinqChaqueCote(Places ,NbPersonnes) :-
42     NbPersGauche #= NbPersonnes/2,
43     ( for(K,1,NbPersonnes),fromto(0,In ,Out ,NbPersonnesAGauche),param(
44         Places)
45         do
46             Var #= Places[K],
47             (Var #< 0) => (Out #= In + 1),
48             (Var #> 0) => (Out #= In)
49     ),
```

```

48     NbPersGauche #= NbPersonnesAGauche .
49
50 parentsEncadrentEnfants(Places) :-
51     PlaceMere #= Places[4],
52     PlacePere #= Places[8],
53     ((PlaceMere #= ic:max(Places)) and (PlacePere #= ic:min(Places)))
54     or
55     ((PlacePere #= ic:max(Places)) and (PlaceMere #= ic:min(Places))).
56     /*
57     (maxlist(Places,PlaceMere) and minlist(Places,PlacePere))
58     or
59     (maxlist(Places,PlacePere) and minlist(Places,PlaceMere)).
60     */
61
62 danEtMaxDevantParents(Places) :-
63     PlaceDan #= Places[6],
64     PlaceMax #= Places[9],
65     PlaceMere #= Places[4],
66     PlacePere #= Places[8],
67     (((PlaceMere #<0) and (PlaceDan #<0)) and ((PlaceMere #= PlaceDan -
68         1) and (PlacePere #= PlaceMax + 1)))
69     or
70     (((PlaceMere #<0) and (PlaceMax #<0)) and ((PlaceMere #= PlaceMax -
71         1) and (PlacePere #= PlaceDan + 1)))
72     or
73     (((PlacePere #<0) and (PlaceDan #<0)) and ((PlacePere #= PlaceDan -
74         1) and (PlaceMere #= PlaceMax + 1)))
75     or
76     (((PlacePere #<0) and (PlaceMax #<0)) and ((PlacePere #= PlaceMax -
77         1) and (PlaceMere #= PlaceDan + 1))).
78
79 solve(Places) :-
80     getData(Poids,NbPers,NbPlac),
81     defineVars(Places,NbPlac,NbPers),
82     pasMemesPlaces(Places),
83     balancoireEquilibree(Poids,NbPers,Places),
84     cinqChaqueCote(Places,NbPers),
85     parentsEncadrentEnfants(Places),
86     danEtMaxDevantParents(Places),
87     getVarList(Places,NbPers,L),
88     labeling(L).
89
90 /* Tests
91 [eclipse 29]: solve(P).
92
93 P = [ ](-6, -5, -4, -8, 2, 5, 3, 6, -7, 1)
94 Yes (0.13s cpu, solution 1, maybe more) ? ;
95
96 P = [ ](-6, -5, -1, 8, 5, 7, 3, -8, -7, 1)
97 Yes (0.15s cpu, solution 2, maybe more) ? ;
98
99 P = [ ](-6, -5, 1, -8, -2, 6, 5, 7, -7, 4)
100 Yes (0.02s cpu, solution 3, maybe more) ?

```

```

97  */
98
99  % Impose que la mere soit a gauche sur la balançoire
100 elimineSymetrie(Places) :-
101     Places[4] #< 0.
102
103 solve2(Places) :-
104     getData(Poids, NbPers, NbPlac),
105     defineVars(Places, NbPlac, NbPers),
106     elimineSymetrie(Places), %Gain de temps monstrueux
107     pasMemesPlaces(Places),
108     cinqChaqueCote(Places, NbPers),
109     balançoireEquilibree(Poids, NbPers, Places),
110     parentsEncadrentEnfants(Places),
111     danEtMaxDevantParents(Places),
112     getVarList(Places, NbPers, L),
113     labeling(L).
114
115 /* Tests
116 [eclipse 31]: solve2(P).
117
118 P = [ ](-6, -5, -4, -8, 2, 5, 3, 6, -7, 1)
119 Yes (0.02s cpu, solution 1, maybe more) ? ;
120
121 P = [ ](-6, -5, 1, -8, -2, 6, 5, 7, -7, 4)
122 Yes (0.04s cpu, solution 2, maybe more) ? ;
123
124 P = [ ](-6, -5, 3, -8, -3, -7, 4, 7, 6, 5)
125 Yes (0.01s cpu, solution 3, maybe more) ?
126 */
127
128 balançoireEquilibree2(Poids, NbPersonnes, Places, SNormeMoments) :-
129     ( for(J, 1, NbPersonnes), fromto(0, In, Out, SommeMoments), fromto(0, In2,
130         Out2, SNormeMoments), param(Places, Poids)
131         do
132             Distance #= Places[J],
133             Poids1 #= Poids[J],
134             Out #= In + Distance * Poids1,
135             (Places[J] #> 0) => (Out2 #= In2 + Distance * Poids1)
136             ,
137             (Places[J] #=< 0) => (Out2 #= In2)
138         ),
139     SommeMoments #= 0.
140
141 solve3(Places) :-
142     getData(Poids, NbPers, NbPlac),
143     defineVars(Places, NbPlac, NbPers),
144     elimineSymetrie(Places), %Gain de temps monstrueux
145     pasMemesPlaces(Places),
146     cinqChaqueCote(Places, NbPers),
147     balançoireEquilibree2(Poids, NbPers, Places, SNormeMom),
148     parentsEncadrentEnfants(Places),
149     danEtMaxDevantParents(Places),

```

```

148         getVarList(Places ,NbPers ,L) ,
149         minimize( labeling (L) ,SNormeMom) .
150
151     /* Tests
152     [eclipse 25]: solve3(P).
153     Found a solution with cost 874
154     Found a solution with cost 872
155     Found a solution with cost 802
156     Found no solution with cost -1.0Inf .. 801
157
158     P = [ ](-3, 1, -2, -6, -1, 4, 3, 5, -5, 2)
159     Yes (1.66 s cpu)
160     */
161
162     % Version 1 de search :
163
164     solve4(Places) :-
165         getData(Poids ,NbPers ,NbPlac) ,
166         defineVars(Places ,NbPlac ,NbPers) ,
167         elimineSymetrie(Places) ,
168         pasMemesPlaces(Places) ,
169         cinqChaqueCote(Places ,NbPers) ,
170         balancoireEquilibree2(Poids ,NbPers ,Places ,SNormeMom) ,
171         parentsEncadrentEnfants(Places) ,
172         danEtMaxDevantParents(Places) ,
173         getVarList(Places ,NbPers ,L) ,
174         minimize( search(L,0,occurrence ,indomain_min ,complete ,[ ]) ,SNormeMom) .
175
176     /* Tests
177     [eclipse 9]: solve4(P).
178     Found a solution with cost 953
179     Found a solution with cost 852
180     Found a solution with cost 834
181     Found a solution with cost 802
182     Found no solution with cost -1.0Inf .. 801
183
184     P = [ ](-3, 1, -2, -6, -1, 4, 3, 5, -5, 2)
185     Yes (2.50 s cpu)
186     */
187     % Ici , il semble que most_constrained soit plus adapte que occurrence , vu la
188         vitesse de reponse (2.13 s au lieu de 2.50 s)
189
190     % Version 2 de search :
191
192     /* Ici on vise a essayer de mettre les places les plus au centre en premier ,
193     parce que c'est dans ce cas que l'on minimise le mieux le moment total. */
194     quick_sort([ ] ,[ ]) .
195     quick_sort([H|T] ,Sorted):-
196         pivoting(H,T,L1,L2) ,quick_sort(L1 ,Sorted1) ,quick_sort(L2 ,Sorted2) ,
197         append(Sorted1 ,[H|Sorted2] ,Sorted) .
198
199     pivoting(_H,[ ] ,[ ] ,[ ]) .
200     pivoting(H,[X|T] ,[X|L] ,G):-abs(X)=<abs(H) ,pivoting(H,T,L,G) ,!.

```

```

200 pivoting (H,[X|T],L,[X|G]):-abs(X)>abs(H),pivoting (H,T,L,G),!.
201
202 /* Tests
203 [eclipse 50]: quick_sort([1,2,3,4,-1,-2,-3,-4],L).
204
205 L = [-1, 1, -2, 2, -3, 3, -4, 4]
206 Yes (0.00s cpu)
207 */
208
209 choice(X) :-
210     get_domain_as_list(X,L),
211     quick_sort(L,L2),
212     member(X,L2).
213
214 solve5(Places) :-
215     getData(Poids,NbPers,NbPlac),
216     defineVars(Places,NbPlac,NbPers),
217     elimineSymetrie(Places),
218     pasMemesPlaces(Places),
219     cinqChaqueCote(Places,NbPers),
220     balancoireEquilibree2(Poids,NbPers,Places,SNormeMom),
221     parentsEncadrentEnfants(Places),
222     danEtMaxDevantParents(Places),
223     getVarList(Places,NbPers,L),
224     minimize(search(L,0,input_order,choice,complete,[]),SNormeMom).
225
226 /* Tests
227 [eclipse 32]: solve5(P).
228 Found a solution with cost 1216
229 Found a solution with cost 956
230 Found a solution with cost 947
231 Found a solution with cost 917
232 Found a solution with cost 852
233 Found a solution with cost 802
234 Found no solution with cost -1.0Inf .. 801
235
236 P = [ ](-3, 1, -2, -6, -1, 4, 3, 5, -5, 2)
237 Yes (1.80s cpu)
238 */
239
240 % Version 3 de search :
241
242 solve6(Places) :-
243     getData(Poids,NbPers,NbPlac),
244     defineVars(Places,NbPlac,NbPers),
245     elimineSymetrie(Places),
246     pasMemesPlaces(Places),
247     cinqChaqueCote(Places,NbPers),
248     balancoireEquilibree2(Poids,NbPers,Places,SNormeMom),
249     parentsEncadrentEnfants(Places),
250     danEtMaxDevantParents(Places),
251     getVarList(Places,NbPers,L),
252     minimize(search(L,0,occurrence,choice,complete,[]),SNormeMom).

```

```

253
254 /* Tests
255 [eclipse 52]: solve6(P).
256 Found a solution with cost 933
257 Found a solution with cost 898
258 Found a solution with cost 872
259 Found a solution with cost 852
260 Found a solution with cost 848
261 Found a solution with cost 834
262 Found a solution with cost 802
263 Found no solution with cost -1.0Inf .. 801
264
265 P = [](-3, 1, -2, -6, -1, 4, 3, 5, -5, 2)
266 Yes (2.98s cpu)
267 */
268
269 % Le resultat n'est pas satisfaisant ici
270
271 % Version 4 :
272
273 solve7(Places) :-
274     getData(Poids ,NbPers ,NbPlac) ,
275     defineVars(Places ,NbPlac ,NbPers) ,
276     elimineSymetrie(Places) ,
277     pasMemesPlaces(Places) ,
278     cinqChaqueCote(Places ,NbPers) ,
279     balancoireEquilibree2(Poids ,NbPers ,Places ,SNormeMom) ,
280     parentsEncadrentEnfants(Places) ,
281     danEtMaxDevantParents(Places) ,
282     getVarList2(Places ,NbPers ,L) ,
283     minimize(search(L,0,input_order ,indomain_split ,complete ,[]) ,SNormeMom
        ) .

```