Compte Rendu TP6

Prédicats importés du TP Précédent

```
vecProduct(V1, V2, VRes):-
    dim(V1,[D]),
    dim(V2,[D]),
    dim(VRes,[D]),
    (foreacharg(Elt1,V1), foreacharg(Elt2,V2), foreacharg(EltRes,VRes) do
        EltRes #= Elt1 * Elt2
    ).

vecSum(Vec,Sum):-(
    foreacharg(Elt,Vec), fromto(0,In,Out,Sum) do
        Out #= In + Elt
    ).

vecDotProduct(V1, V2, S):-
    vecProduct(V1, V2, TmpVec),
    vecSum(TmpVec, S).
```

Question 1

```
:-lib(ic).

poids([](24, 39, 85, 60, 165, 6, 32, 123, 7, 14)).

sieges(S):-
        dim(S, [10]),
    S #:: [ -8 .. -1, 1 .. 8].

    equilibre(S, P):-
    vecDotProduct(S, P, 0).

countPositif(V,Res):-
(
    foreacharg(Elt,V),
    fromto(0,In,Out,Res)
    do
    (Elt #> 0) => (Out #= In +1),
    (Elt #=< 0) => (Out #= In)
).

%On sait que s'il y en a 5 à droite il y en aura 5 à gauche
nbDroite(V):-
countPositif(V,C),
C #= 5.

parentsAuBout(S):-
        Lou is S[4],
        Tom is S[8],
```

```
        ic:max(S, Lou),
        ic:min(S, Tom).

enfantsAvecParents(S):-
        Lou is S[4],
        Tom is S[8],
        Dan is S[6],
        Max is S[9],
        (Dan #= Tom + 1 and Max #= Lou - 1)
        or
        (Dan #= Lou - 1 and Max #= Tom + 1).

constraints(S, P):-
        equilibre(S, P),
        parentsAuBout(S),
        enfantsAvecParents(S),
        nbDroite(S),
        alldifferent(S).

solve(Sieges):-
        poids(Poids),
        sieges(Sieges),
        constraints(Sieges, Poids),
        labeling(Sieges).
```

Tests

Nous ne voyions pas comment tester les prédicats séparément donc nous avons considéré la question 2 comme un test de la globalité du programme.

Question 2

Test

```
solve(S).

S = [](-6, -5, -1, 8, 5, 7, 3, -8, -7, 1)
Yes (0.01s cpu, solution 1, maybe more) ? ;

S = [](-6, -5, 2, 8, 4, -7, -2, -8, 7, 5)
Yes (0.01s cpu, solution 2, maybe more) ? ;

S = [](-6, -5, 2, 8, 4, 7, -2, -8, -7, 6)
Yes (0.01s cpu, solution 3, maybe more) ? ;

S = [](-6, -5, 5, 7, 3, 6, -1, -8, -7, 2)
Yes (0.02s cpu, solution 4, maybe more) ? ;

S = [](-6, -5, 6, 8, 2, 7, -1, -8, -7, 3)
Yes (0.03s cpu, solution 5, maybe more) ? ;

S = [](-6, -4, -2, 8, 5, -7, 4, -8, 7, 1)
Yes (0.04s cpu, solution 6, maybe more) ?
```

On obtient bien des solutions où :

- Les Valeurs 4 et 8 (les parents) sont le maximum et le minimum du tableau
- Les valeurs 6 et 9 sont directement adjacentes aux 4 et 8 (vers le milieu)
- La somme des moments est égale à 0
- Il y a 5 valeurs négatives et 5 valeurs positives

Question 3

La symmétrie la plus remarquable est due au fait qu'on équilibre le côté droit et le côté gauche, donc une fois qu'une solution est trouvée, on peut juste l'inverser. Pour éviter cela nous avons modifié le prédicat parentsAuBout pour décider à l'avance quel parent est à gauche et lequel est à droite.

Question 4

Prédicats de service

```
vecAbs(Vector, Abs):-
        dim(Vector, [Dim]),
        dim(Abs, [Dim]),
        (for(Ind, 1, Dim),
        param(Vector, Abs)
        do
                Elem is Vector[Ind],
                (Elem #> 0) => Abs[Ind] #= Elem,
                (Elem #=< 0) => Abs[Ind] #= -Elem
    ).

sommeMoments(S, P, Res):-
        vecAbs(S, SAbs),
        vecDotProduct(SAbs, P, Res).
```

Test

```
vecAbs([](1,0,-2),X).
X = [](1, 0, 2)
Yes (0.00s cpu)


sommeMoments([](0,1,-4),[](-2,-3,7),X).
X = 25
Yes (0.00s cpu)
```

Version basique

```
solve2(Sieges,SommeMoments):-
        poids(Poids),
        sieges(Sieges),
        constraints(Sieges, Poids),
        sommeMoments(Sieges, Poids, SommeMoments),
        ic:labeling(Sieges).
```

Test

```
minimize(solve2(S, SommeMoments), SommeMoments).
Found a solution with cost 2914
Found a solution with cost 2858
Found a solution with cost 2808
Found a solution with cost 2722
Found a solution with cost 2716
Found a solution with cost 2708
Found a solution with cost 2694
Found a solution with cost 2602
Found a solution with cost 2594
Found a solution with cost 2524
Found a solution with cost 2474
Found a solution with cost 2430
Found a solution with cost 2392
Found a solution with cost 2344
Found a solution with cost 2296
Found a solution with cost 2218
Found a solution with cost 2196
Found a solution with cost 2154
Found a solution with cost 2142
Found a solution with cost 2064
Found a solution with cost 1958
Found a solution with cost 1890
Found a solution with cost 1748
Found a solution with cost 1744
Found a solution with cost 1704
Found a solution with cost 1604
Found no solution with cost -1.0Inf .. 1603

S = [](3, -1, 2, 6, 1, -4, -3, -5, 5, -2)
SommeMoments = 1604
Yes (2.92s cpu)
```

Version 1

```
solve3(Sieges,SommeMoments):-
        poids(Poids),
        sieges(Sieges),
        constraints(Sieges, Poids),
        sommeMoments(Sieges, Poids, SommeMoments),
        search(Sieges, 0, occurence, indomain, complete, []).
```

Dans l'argument +Select, on utilise occurence, une façon classique d'optimiser en labelisant les variables les plus contraintes en priorité.

Test

```
minimize(solve3(S, SommeMoments), SommeMoments).
Found a solution with cost 1890
Found a solution with cost 1604
Found no solution with cost -1.0Inf .. 1603

S = [](3, -1, 2, 6, 1, -4, -3, -5, 5, -2)
SommeMoments = 1604
Yes (1.10s cpu)
```

On a quasiment triplé la vitesse d'exécution

Version 2

Commentaire

Pour minimiser les moments le plus vite, il vaut mieux placer les personnes sur les places les plus proche du centre. Pour cela on utilise via l'argument +Choice de search le mode indomain_split, pour tester les valeurs au centre du domaine en priorité

```
solve4(Sieges,SommeMoments):-
        poids(Poids),
        sieges(Sieges),
        constraints(Sieges, Poids),
        sommeMoments(Sieges, Poids, SommeMoments),
        search(Sieges, 0, input_order, indomain_split, complete, []).
```

Test

```
minimize(solve4(S, SommeMoments), SommeMoments).
Found a solution with cost 2914
Found a solution with cost 2858
Found a solution with cost 2808
Found a solution with cost 2722
Found a solution with cost 2716
Found a solution with cost 2708
Found a solution with cost 2694
Found a solution with cost 2602
Found a solution with cost 2594
Found a solution with cost 2524
Found a solution with cost 2474
Found a solution with cost 2430
```

```
Found a solution with cost 2392
Found a solution with cost 2344
Found a solution with cost 2296
Found a solution with cost 2218
Found a solution with cost 2196
Found a solution with cost 2154
Found a solution with cost 2142
Found a solution with cost 2064
Found a solution with cost 1958
Found a solution with cost 1890
Found a solution with cost 1748
Found a solution with cost 1744
Found a solution with cost 1704
Found a solution with cost 1604
Found no solution with cost -1.0Inf .. 1603

S = [](3, -1, 2, 6, 1, -4, -3, -5, 5, -2)
SommeMoments = 1604
Yes (1.85s cpu)
```

On va toujours plus vite que labeling mais pas plus vite que la Version 1

Version 3

On conserve indomain_split mais on reprend le mode occurence

```
solve5(Sieges,SommeMoments):-
        poids(Poids),
        sieges(Sieges),
        constraints(Sieges, Poids),
        sommeMoments(Sieges, Poids, SommeMoments),
        search(Sieges, 0, occurence, indomain_split, complete, []).
```

Test

```
minimize(solve5(S, SommeMoments), SommeMoments).
Found a solution with cost 1890
Found a solution with cost 1604
Found no solution with cost -1.0Inf .. 1603

S = [](3, -1, 2, 6, 1, -4, -3, -5, 5, -2)
SommeMoments = 1604
Yes (0.69s cpu)
```

Cette fois on est plus rapide que les deux versions précédentes

Version 4

Commentaire

Puisqu'on cherche à minimiser les moments, on sait que les poids les plus importants vont être les plus proche du centre. Comme on labelise déjà en essayant les valeurs au centre en premier, on va imposer que les poids les plus importants soient testé en premier. Cette priorité n'est appliquée qu'après la priorité occurence. Puisque le mode occurence quand il a la même priorité s'en remet à l'ordre dans l'input, on va utilise run nouveau prédicat pour changer l'ordre des personnes dans l'input.

```
ordrePoids(Sieges, Res):-
        dim(Sieges, [Dim]),
        dim(Res, [Dim]),
        Luc is Sieges[5],
        Res[1] #= Luc,
        Tom is Sieges[8],
        Res[2] #= Tom,
        Jim is Sieges[3],
        Res[3] #= Jim,
        Lou is Sieges[4],
        Res[4] #= Lou,
        Zoe is Sieges[2],
        Res[5] #= Zoe,
        Ted is Sieges[7],
        Res[6] #= Ted,
        Ron is Sieges[1],
        Res[7] #= Ron,
        Kim is Sieges[10],
        Res[8] #= Kim,
        Max is Sieges[9],
        Res[9] #= Max,
        Dan is Sieges[6],
        Res[10] #= Dan.

solve6(Sieges,SommeMoments):-
        poids(Poids),
        sieges(Sieges),
        constraints(Sieges, Poids),
        sommeMoments(Sieges, Poids, SommeMoments),
        ordrePoids(Sieges,Ordre),
        search(Ordre, 0, occurence, indomain_split, complete, []).
```

Test

```
minimize(solve6(S, SommeMoments), SommeMoments).
Found a solution with cost 2426
Found a solution with cost 2354
Found a solution with cost 2326
Found a solution with cost 2324
Found a solution with cost 2262
Found a solution with cost 2204
Found a solution with cost 2132
Found a solution with cost 2038
```

```
Found a solution with cost 2006
Found a solution with cost 1912
Found a solution with cost 1890
Found a solution with cost 1880
Found a solution with cost 1696
Found a solution with cost 1604
Found no solution with cost -1.0Inf .. 1603

S = [](3, -1, 2, 6, 1, -4, -3, -5, 5, -2)
SommeMoments = 1604
Yes (0.61s cpu)
```

Bonus

Remarque : pour réduire au plus tôt le domaine des places de Tom et Lou il est possible d'énoncer des contraintes redondantes

En effet pusiqu'on sait qu'ils seront au maximum de leur côté de la balançoire sur lequel il y a 4 autres personnes, on sait qu'ils ont au minimum la 5e place de leur côté

```
parentsAuBout2(S):-
        Lou is S[4],
        Tom is S[8],
        ic:max(S, Lou),
        ic:min(S, Tom),
        Lou #>4,
        Tom #<(-4).

constraints2(S, P):-
        equilibre(S, P),
        parentsAuBout2(S),
    enfantsAvecParents(S),
    nbDroite(S),
        alldifferent(S).

solve7(Sieges,SommeMoments):-
        poids(Poids),
        sieges(Sieges),
        constraints2(Sieges, Poids),
        sommeMoments(Sieges, Poids, SommeMoments),
        ordrePoids(Sieges,Ordre),
        search(Ordre, 0, occurence, indomain_split, complete, []).
```

Test

```
minimize(solve7(S, SommeMoments), SommeMoments).
Found a solution with cost 2914
Found a solution with cost 2864
Found a solution with cost 2744
Found a solution with cost 2722
```

```
Found a solution with cost 2694
Found a solution with cost 2588
Found a solution with cost 2572
Found a solution with cost 2432
Found a solution with cost 2410
Found a solution with cost 2384
Found a solution with cost 2322
Found a solution with cost 2246
Found a solution with cost 2222
Found a solution with cost 2202
Found a solution with cost 2124
Found a solution with cost 2110
Found a solution with cost 2078
Found a solution with cost 2026
Found a solution with cost 1940
Found a solution with cost 1908
Found a solution with cost 1894
Found a solution with cost 1880
Found a solution with cost 1844
Found a solution with cost 1828
Found a solution with cost 1760
Found a solution with cost 1712
Found a solution with cost 1672
Found a solution with cost 1668
Found a solution with cost 1666
Found a solution with cost 1604
Found no solution with cost -1.0Inf .. 1603

S = [](3, -1, 2, 6, 1, -4, -3, -5, 5, -2)
SommeMoments = 1604
Yes (1.06s cpu)
```

Malgré cela nous n'avons pas amélioré notre vitesse de recherche...