I.A. TD 1 Backtracking et Heuristiques

13 octobre 2010

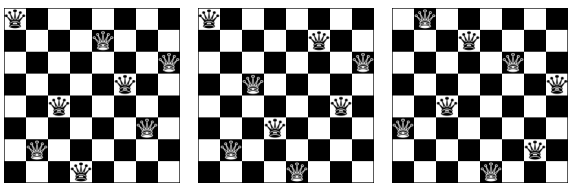
Table des matières

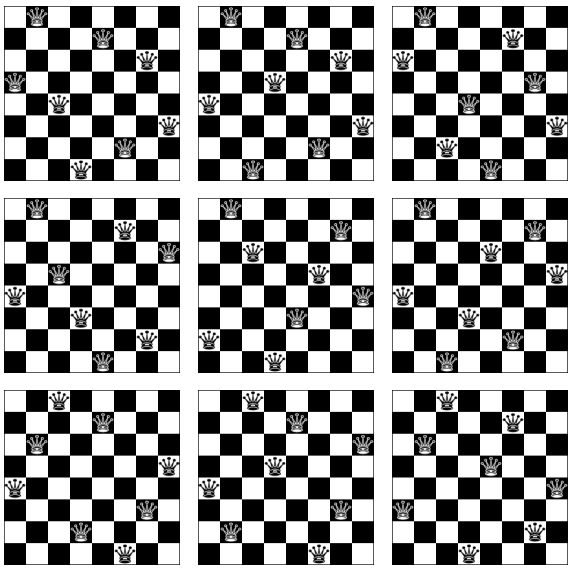
1 Le problème des 8 reines

Il s'agit de placer n reines sur un échiquier (n x n) sans qu'elles se menacent. Deux reines ne doivent pas se trouver sur la même ligne, sur la même colonne ou sur la même diagonale.

2 Solutions

Si on considère un échiquier 8 x 8, il y a 92 solutions, mais seulement 12 qui ne peuvent pas être obtenues les unes à partir des autres par symétrie ou rotation.





3 Algorithme

La méthode utilisée consiste à écrire un algorithme récursif capable de retourner en arrière, on parle souvent de backtracking. Pour cela il faut se souvenir des choix qui ont été faits précédemment. Les algorithmes à retour arrière procèdent pas essais/erreurs. Le schéma général consiste à décomposer le processus d'essais et erreurs en tâches partielles. Souvent ces tâches s'expriment naturellement de façon récursive et consistent en l'exploration d'un nombre fini de sous tâches. On peut voir tout ce processus comme un processus de recherche qui construit peu à peu et parcourt en l'élaguant un arbre des sous tâches. En général, celui-ci croît très vite, souvent de manière exponentielle, en fonction d'un paramètre donné. Fréquemment l'arbre de recherche doit être élagué à l'aide d'heuristique.

Pour écrire l'algorithme, on s'inspirera de ce qui a été présenté en cours.

L'objectif de ce TP est donc de mettre en œuvre une technique de recherche :
– utilisant le backtracking ;
– utilisant des heuristiques permettant d'augmenter la taille du problème traité.

3.1 Premier algorithme

Une méthode naïve consiste à placer une reine sur la première ligne, puis la deuxième ligne... la n-ième ligne et à choisir une colonne en utilisant le backtracking lorsque cela n'est plus possible. Votre premier programme devra tout d'abord trouver une solution et calculer le nombre de retour en arrière, puis ensuite calculer toutes les solutions et calculer le nombre total de retour arrière. Vous modifierez ensuite votre algorithme afin d'utiliser les symétries, et si vous avez le temps les rotations.

3.2 En utilisant les heuristiques

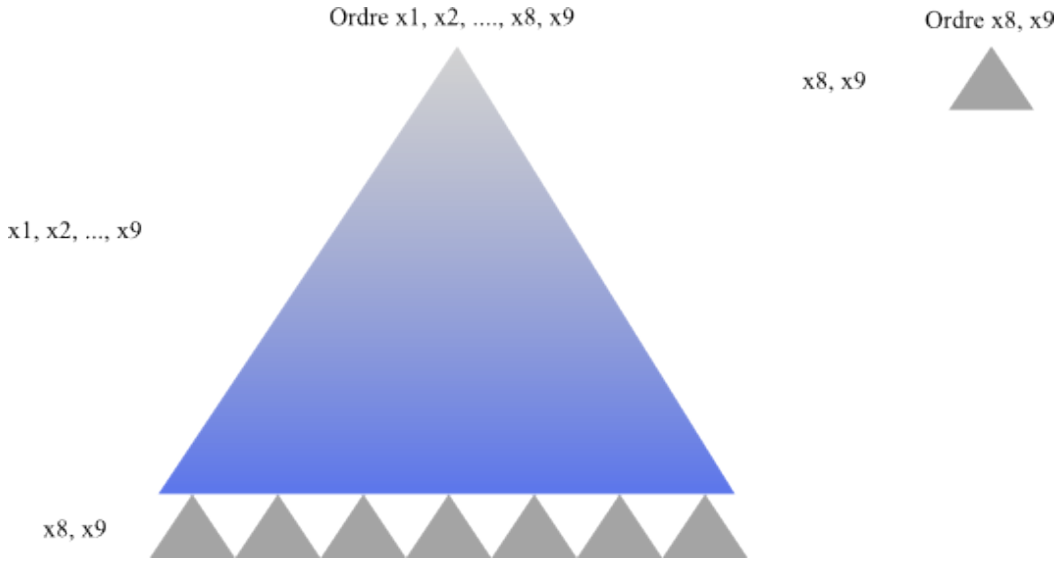
On ne cherche plus maintenant toutes les solutions, mais une solution et à faire croître n, par exemple avec des échiquiers de 100 x 100...

4 Les Heuristiques

Lors de la résolution du problème on développe un arbre et ainsi à chaque nœud il faut déterminer quelle variable il faut fixer et quelle valeur il faut lui donner. On parle d'ordre de sélection. Ces choix peuvent se faire soit de façon statique donc à l'avance soit de façon dynamique en cours de résolution. Nous avons utilisé la première démarche, dans le premier algorithme, en traitant le problème par ordre d'apparition des variables de la plus petite valeur à la plus grande.

4.1 Ordre de sélection

L'ordre de sélection intervient à la fois sur le choix des variables et sur le choix des valeurs. Ainsi l'ordre de sélection des variables influence la topologie de l'arbre de recherche et de ce fait sa taille.



L'ordre de sélection des valeurs quant à lui est souvent moins important s'il est nécessaire de parcourir tout l'arbre de recherche. En effet, il n'affecte que l'ordre dans lequel les branches sont explorées et donc dans lequel les solutions sont trouvées, cependant si on cherche une seule solution voire la meilleure solution cet ordre prend toute sont importance.

Deux grands principes régissent l'ordre de sélection :

1. Des variables : «Pour réussir, on essaye en premier les cas où on est le plus susceptible d'échouer». C'est le *first fail principle*.
2. Des valeurs : «On essaye d'abord de réussir». Démarche opposée au first-fail.

4.1.1 Sélection des variables : *first fail principle*.

Cette approche cherche à déterminer le plus rapidement possible les sous arbres qui ne conduisent à aucune solution. On s'occupe donc de la partie la plus difficile en priorité puisque celle-ci ne le deviendra pas moins lors de la résolution du problème. Plusieurs heuristiques peuvent suivre cette démarche :

– Plus petit domaine d'abord : on suppose que moins on a de valeurs, moins on a de chance de trouver une valeur pour la variable ;
– Plus petit domaine d'abord et plus grand nombre de contraintes ceci dans l'éventualité d'une égalité ;
– Plus petit domaine d'abord et plus de valeurs communes avec ses pairs ceci dans l'éventualité d'une égalité. On suppose que les variables doivent prendre des valeurs différentes ;
– Plus petite valeur maximum dans le domaine d'abord : on utilise en particulier cette technique quand les domaines représentent des instants dans le temps et que l'on procède de façon chronologique en allant au plus urgent ;
– Moindre regret : ceci ne suit pas réellement le principe du first fail mais un impératif d'optimisation. On favorise la plus grande différence de coût entre les deux meilleures valeurs dans le domaine ;

4.1.2 Sélection des valeurs

Si l'heuristique est idéale elle nous donne directement une solution sans backtracking, c'est malheureusement très rarement le cas. Le plus souvent les heuristiques s'inspirent de la structure particulière du problème et on tente de maximiser une estimation des chances de réussite.

4.2 Application au problème des n reines

Vous devez appliquer les différentes heuristiques présentées et déterminer sur quel(s) principe(s) elles reposent. Vous mesurerez en faisant varier n : le nombre de backtracking pour obtenir une solution ainsi que le temps CPU consommé. Vous comparerez ensuite les différentes heuristiques.

4.2.1 Première heuristique

Cette première heuristique est celle exposée en cours, qui consiste à placer une reine par ligne de la première à la dernière ligne et de choisir la colonne qui laisse le plus grand nombre de cases libres pour les lignes du dessous.

4.2.2 Deuxième heuristique

On place toujours une reine par ligne de la première à la dernière ligne et on choisit la colonne qui occupe le plus la ligne suivante. De cette façon on cherche à diminuer au maximum le degré des nœuds de l'arbre de façon locale et on peut donc espérer que le nombre total de nœuds dans l'arbre sera plus faible. Cela nous donne pour 8 reines l'échiquier suivant où les chiffres en rouge représentent les reines posées. Ainsi la première colonne de la première ligne laisse 6 colonnes disponibles pour la deuxième ligne.

6	5	5	5	5	5	5	6
-	-	-	4	4	3	3	4
2	-	2	-	-	-	-	2
-	-	3	-	-	-	2	-
-	-	-	1	1	-	-	-
-	-	-	-	-	-	-	1
-	-	1	-	-	-	-	-
-	-	-	-	1	-	-	-

L'énumération s'est faite dans l'ordre naturel des lignes on peut modifier cet ordre en énumérant les variables des plus contraintes aux moins contraintes. De ce fait l'ordre sur les lignes se fait maintenant en fonction de celles ayant le moins de positions libres. Une dernière amélioration de cette heuristique consiste à placer une reine obligatoirement dans une colonne lorsqu'elle n'a plus qu'une position de libre. Vous pourrez également tester que si on combine le critère de choix sur les lignes et les colonnes (i.e. on s'intéresse aux variables les plus contraintes à la fois sur les lignes et le colonne) cela fonctionne mal.

4.2.3 Troisième heuristique

- Lorsqu'on place une reine dans le premier ou dernier tiers de l'échiquier il faut commencer par le milieu ;
- Lorsqu'on place une reine dans le tiers du centre, il faut commencer par les bords de manière alternée ;
- On considère les colonnes quand il reste moins de 4 positions de libre et moins que le minimum des lignes ;

5 Quelques pointeurs

- La page de François Morain.
- Un peu de lecture : «Le Tableau du Maître Flamand» d'Arturo Perez-Reverte, pour les amateurs d'échecs et de romans policiers.
- Vous pouvez vous inspirer de ce petit programme qui prend en entrée une solution pour le problème des 8 reines et génère une image de l'échiquier. Ce programme nécessite ImageMagick et les images suivantes : reine noire, reine blanche et échiquier.