Rapport

UE PPC

Projet N-Queens

Projet 2017-2018

Étudiants : LESUR Antoine, VERON Yann

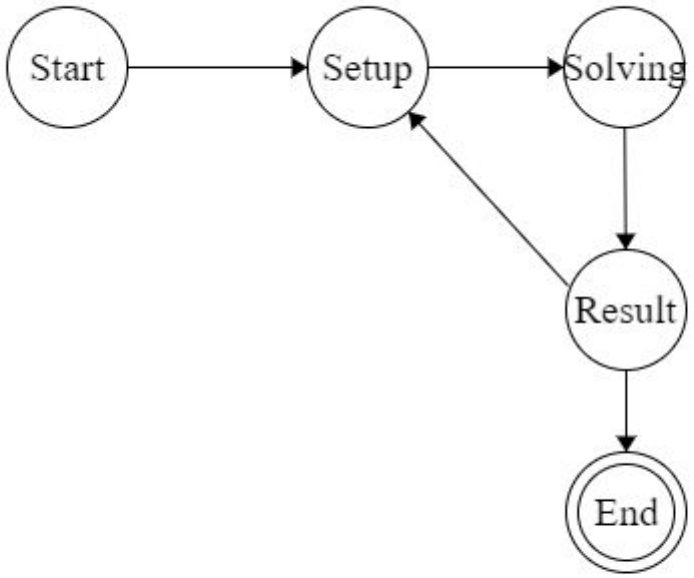
Introduction

Le problème des N-Reines est une extension du problème des 8 reines, qui consiste à placer 8 reines sur un échiquier de 8 case de côté sans que celles-ci puissent entrer en conflit.
Une reine est en conflit avec une autre si et seulement si les deux reines sont sur la même ligne, la même colonne ou la même diagonale.
Le problème des N-Reines est identique avec pour nombre de reines et de cases N.
Pour résoudre ce problème nous utiliserons Choco.
Choco est un solveur dédié à la programmation par contraintes, sous la forme d'un plugin java, qui permet, en ayant en entrée un certain nombre de contraintes trouver, selon la configuration, une seule ou toutes les solutions d'un problème.

Dans ce projet, nous imposerons une configuration de base valide à Choco, avec un nombre de reines déjà placées.
Notre objectif dans ce rapport sera d'analyser si Choco arrive ou non à trouver une solution, ainsi que le nombre de ces solutions, en fonction de cette configuration aléatoire et en combien de temps il fini son exécution.

Structure de l’application

Déroulement type



Notre programme procède de la manière suivante :

- initialisation des n premières reines sur l’échiquier
- résolution du problème via Choco
- extraction des résultats
- retour à l’initialisation si on souhaite plus de résultats

Choix d’implémentation

Nous avions choisi pour notre premier jet de ce projet d’utiliser la récursivité pour placer les reines, en bouclant jusqu’à avoir une configuration valide, mais les limitations de mémoires dues à la machine virtuelle de JAVA nous ont forcé à choisir une autre solutiuon.
Par conséquent, nous avons choisi d’utiliser des permutations successives de ce tableau

0	1	2	3	4	5	6	7

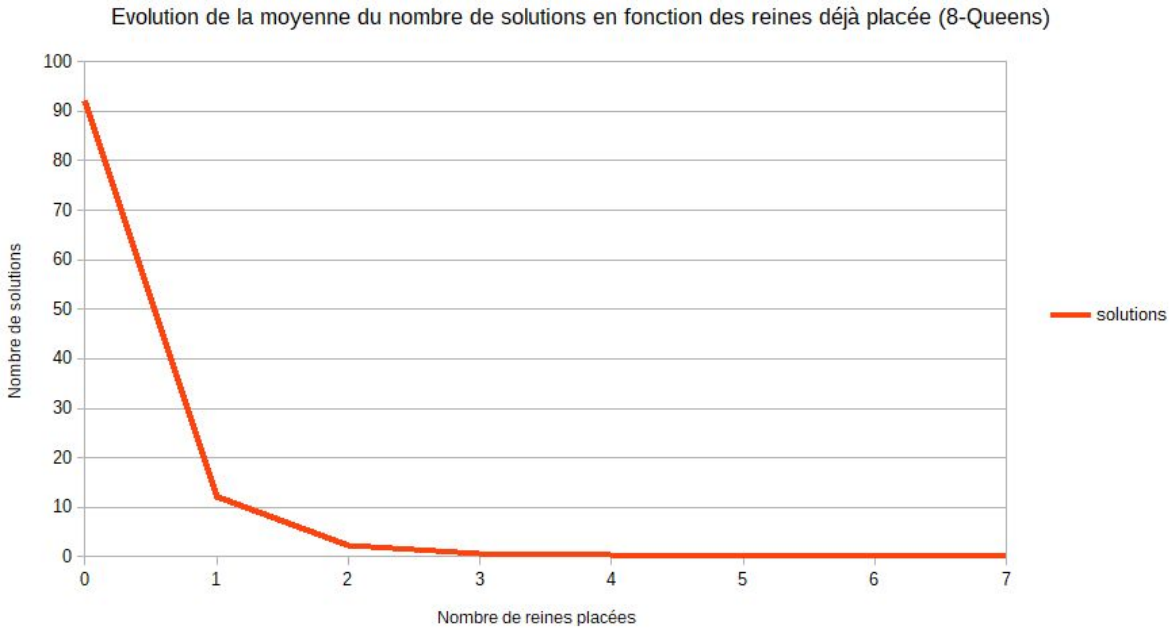
En effet, chaque valeur représente une ligne, et chaque case une colonne,de ce fait la vérification de conflit au niveau des lignes et des colonnes entre les reines est inutile, car chaque reine est seule sur sa ligne et sa colonne.
Il ne reste donc plus qu’à vérifier les diagonales.
On permute donc ce tableau jusqu’à ce qu’aucune des n premières reines ne soit dans la même diagonale qu’une autre des n premières reines, n étant le nombre de reines que l’on souhaite placer.

Une fois que l’initialisation est terminée, on initialise le solveur en lui passant en variables les reines déjà placées, et des variables représentant les reines qu’il reste à placer.
On continue de l’initialiser en lui demandant à ce que toutes les reines soient sur des lignes,colonnes et diagonales différentes.
Enfin, on lancer le solveur pour qu’il résolve le problème.

On réitère le processus, de l’initialisation des premières reines jusqu’à la résolution, autant de fois que nécessaire, avec un nombre de reines maximum et de reines déjà placées différent, puis on exporte les résultats sous format csv, ce qui permettra d’effectuer un traitement des informations sur un logiciel de type Microsoft Excel ou LibreOffice Calc.

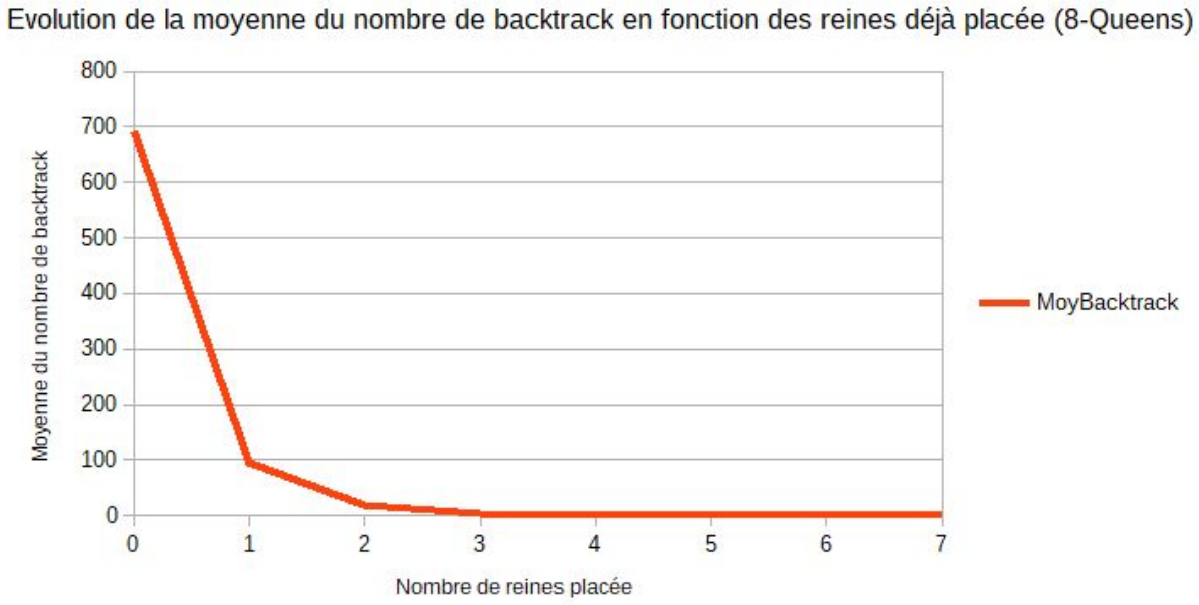
Résultats et Analyse

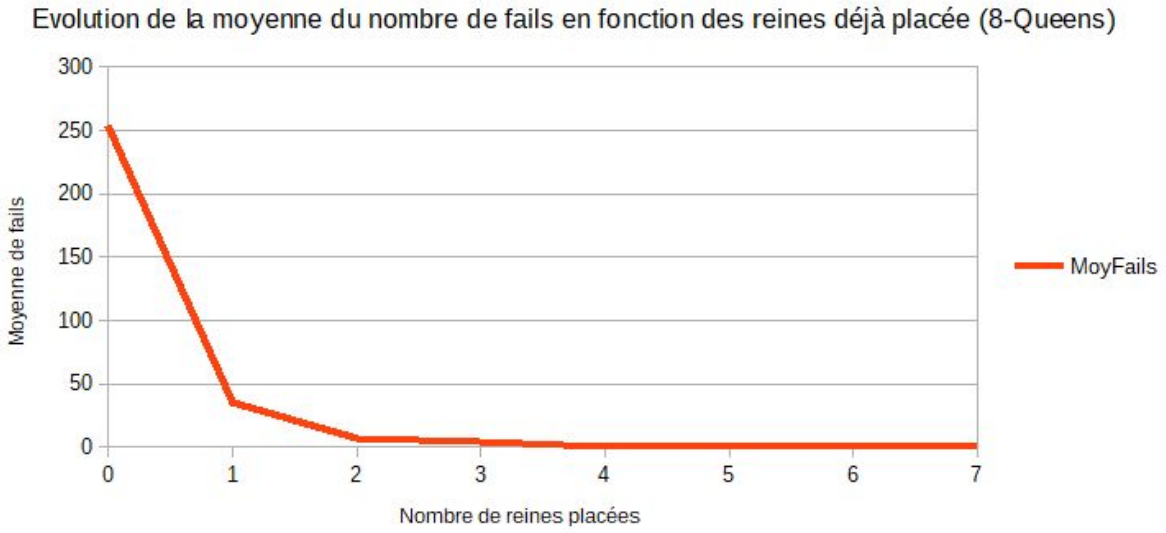
Ces résultats ont été obtenus après 1000 itérations du processus avec un nombre de reines maximum de 8, 10,11,12 et 13 reines, et un nombre de reines déjà placées entre 0 et k-1, k étant le nombre de reines à placer.
Il contiennent le nombre de reine k, le nombre de reines déjà placée n, ainsi que le nombre de solutions, de backtrack, de noeuds, et de “fails” de choco.
Les résultats sont sous forme de fichier .csv , utilisable sous un logiciel de type LibreOffice Calc ou Microsoft Excel.
L’analyse a été poussée pour le problème des 8 reines, mais aurait pu l'être de la même manière pour les autres instances de ce problème.



Comme on peut le voir, la moyenne du nombre de solutions baisse drastiquement une fois la première reine placée, puis continue a baisser jusqu’à être extrêmement proche de 0. Cependant, il semblerait qu’il y ait tout de même un faible pourcentage de chance de trouver une solution, d’après les résultats suivant (issus du fichier 8reines.csv)

Nombre de reines placée	Moyenne du nombre de solutions
0	92
1	12
2	2,248
3	0,6608695652
4	0,2881355932
5	0,1666666667
6	0,1692307692
7	0,246031746





D’après ces deux courbes, on observe une évolution similaire du nombre de fails et de backtrack à celle du nombre de solutions.
D’après la feuille de résultat (8reines.csv/8reines.ods/8reines.xslx), l’évolution du nombre de noeuds suit une courbe identique.

Nous pouvons en conclure qu’il est parfaitement inefficace d’essayer de force une ou plusieurs variables du solveur, car cela réduit drastiquement le nombre de possibilité, ainsi donc que le nombre de solutions. Il est cependant intéressant d’observer que le nombre de backtrack et de fails diminue également, alors que l’on pourrait s’attendre à l’inverse.