|  |
| --- |
| LOG515 – TP2 |
| Rapport de laboratoire |
| Résolveur de sudoku |

|  |
| --- |
| Samuel Beauchemin BEAS26079101, Marc-André Destrempes DESM14119009  6/12/2013 |

Table des matières

[Description de l’algorithme 2](#_Toc358835365)

[Description des problèmes rencontrés 3](#_Toc358835366)

[Description des améliorations que vous avez implémentées 3](#_Toc358835367)

[Autres éléments que vous jugez importants 3](#_Toc358835368)

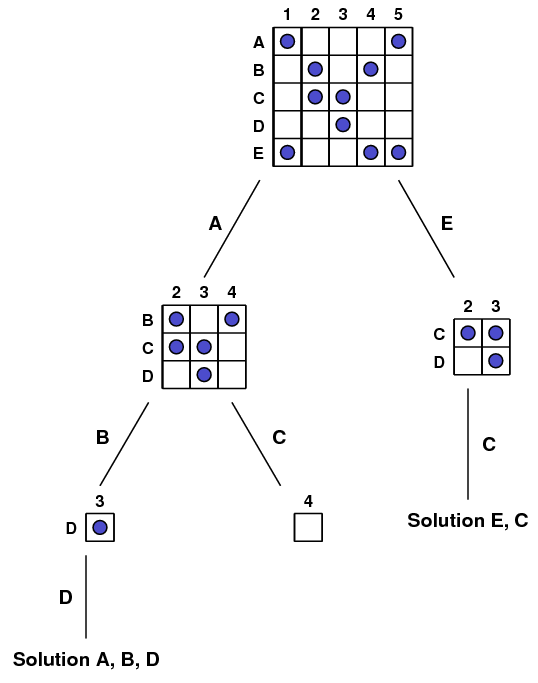
[Références 3](#_Toc358835369)

Table des figures

[Figure 1 – Résolution d’un problème de couverture exacte 2](file:///C:\Users\Marc-Andre\Documents\GitHub\LOG320.TP2\TP2.docx#_Toc358835380)

## Description de l’algorithme

L’algorithme utilisé a été l’« Algorithme X » de Donald E. Knuth. Cet algorithme sert à résoudre efficacement un problème de couverture exacte (« Exact Cover »).

Une couverture exacte est un sous-ensemble contenant toutes les valeurs de son ensemble parent. Chacune des valeurs n’est contenue qu’une seule fois dans le sous-ensemble.

Dans la figure 1, il est possible de voir la résolution d’un problème de couverture exacte. Il s’agit de prendre une colonne et de vérifier les lignes contenant la valeur de cette colonne. Dans l’exemple, la première colonne a d’abord été sélectionnée et les lignes à observer sont les lignes « A » et « E ». Ensuite, pour chacune des lignes choisies, la ligne est gardée comme faisant partie de la solution. Toutes autres lignes contenant les mêmes éléments que la ligne gardée en solution ainsi que cette dernière sont retirées de la matrice. Ce processus est répété jusqu’à ce que la matrice soit vide (solution trouvée) ou qu’une colonne soit vide (il faut revenir en arrière puisque les lignes gardées ne forment pas une solution valide).

Figure 1 – Résolution d’un problème de couverture exacte

Ainsi, il a été possible d’associer ce type de problème à la résolution de grilles de Sudoku. La matrice pour résoudre une grille de Sudoku a une taille de 324 de longueur par 729 de hauteur. Chacune des 729 lignes représente un des neuf différents chiffres pour chacune des cases (ligne et colonne) (9x9x9=729). La largeur de 324 représente les quatre contraintes présentes dans une grille de Sudoku, soit qu’il n’y a qu’un seul chiffre par case (9 colonnes x 9 lignes), qu’un chiffre ne revient qu’une fois par colonne (9 chiffres x 9 colonnes), qu’un chiffre ne revient qu’une fois par ligne (9 chiffres x 9 lignes) et qu’un chiffre ne revient qu’une fois par région (9 chiffres x 9 régions) (9x9 + 9x9 + 9x9 + 9x9 = 324). Il y a donc quatre bits d’activés par ligne dans la matrice. Il faut toutefois retirer l’information superflue pour que la matrice représente la grille de Sudoku à résoudre. Une fois que la matrice est créée, il ne suffit que de la résoudre avec l’algorithme de Donald E. Knuth.

## Description des problèmes rencontrés

D’abord, il a été décidé en premier d’implémenter des méthodes de résolution de grilles de Sudoku utilisées par des humains. Toutefois, lorsque l’implémentation a été terminée, la résolution était loin d’être optimale.

De plus, lorsque l’algorithme de Donald E. Knuth a été adopté pour faire le résolveur, la compréhension de la matrice à construire ainsi que les manipulations à lui faire au niveau de l’implémentation ont pris beaucoup de temps.

## Description des améliorations que vous avez implémentées

Après avoir utilisé la méthode d’implémentation de l’« Algorithme X » de Donald E. Knuth (« Dancing Links »), le résultat n’était pas plus performant que l’algorithme non optimisé. Des améliorations ont dû être apportées afin d’augmenter l’efficacité du résolveur.

D’abord, l’information sur la colonne de chacun des éléments a été gardée dans ceux-ci afin d’éviter plusieurs recherches exhaustives. Cet ajout a amélioré l’efficacité du résolveur, mais ce n’était toujours pas plus efficace que le résolveur non optimisé.

L’amélioration qui a causé une augmentation d’efficacité majeure a été le tri de chacune des colonnes en fonction du nombre d’éléments qu’elles ont. Ainsi, au lieu de prendre la colonne la plus à gauche de la matrice à chaque itération, la colonne avec le moins d’éléments est choisie et diminue grandement le temps d’exécution du résolveur.

## Autres éléments que vous jugez importants

Il aurait pu être possible que le résolveur soit plus rapide en changeant plusieurs types de variables, tel que changer des « Integer » par des « Byte ».

## Références

* <http://en.wikipedia.org/wiki/Exact_cover#Sudoku>
* <http://www.ocf.berkeley.edu/~jchu/publicportal/sudoku/0011047.pdf>
* <http://www.ocf.berkeley.edu/~jchu/publicportal/sudoku/sudoku.paper.html>
* <http://cgi.cse.unsw.edu.au/~xche635/dlx_sodoku/>
* <http://www.ocf.berkeley.edu/~jchu/publicportal/sudoku/presentationboard.pdf>
* <http://www.stolaf.edu/people/hansonr/sudoku/exactcovermatrix.htm>
* <http://www.ams.org/samplings/feature-column/fcarc-kanoodle>
* <http://stackoverflow.com/questions/1935120/representing-a-100k-x-100k-matrix-in-java>