Ozer Guney: 20193803
Gnananga Parfait Ouattara: 20201856

Performance et Analyse

Les algorithmes de la question 1, 2, 3 ont eu un taux de succès de 100% pour l'ensemble des grilles testées. Ceci est normal parce que la stratégie de recherche backtracking retourne une solution complète et la question 1, 2 et 3 l'utilise. Ci-dessous, nous détaillons les performances spécifiques de chaque algorithme, ainsi que les approches techniques employées.

Algorithme Question 1
Cet algorithme résolve les Sudoku à une cadence de 435 grilles par seconde. Cet algorithme utilise les 3 idées de bases de Norvig.

Algorithme Question 2
L'algorithme de la question 2 présente une cadence de résolution de 372 Sudoku par seconde. Cette performance, bien que légèrement inférieure à celle de l'algorithme de la question 1, reste impressionnante. Elle s'explique par l'adoption d'une stratégie de recherche en profondeur pure, sans l'intégration d'heuristiques spécifiques, ce qui impacte légèrement la vitesse de résolution tout en conservant une efficacité remarquable.

Algorithme Question 3
Avec une cadence de 412 Sudoku résolus par seconde, l'algorithme de la question 3 démontre également une grande efficacité. Cette approche utilise la technique des "naked pairs" pour améliorer le processus de résolution.
Technique Employée: Cet algorithme utilise trois fonctions clés: find_naked_pairs, eliminate_naked_pairs, et apply_naked_pairs_if_applicable. La fonction apply_naked_pairs_if_applicable est intégrée au sein de la fonction search3, où elle opère une recherche proactive de "naked pairs" au sein des grilles. Lorsque de tels paires sont identifiées grâce à la fonction find_naked_pairs , ils sont éliminés en appelant la fonction eliminate_naked_pairs, permettant ainsi d'accélérer la résolution en réduisant l'espace de recherche.

sources: GitHub - omerfarukeker/Sudoku-Solver: Solves the user entered Sudoku boards with commonly known strategies like hidden/naked pairs, X-wing, Y-wing etc.

Algorithme Question 4
Hill climbing a été un peu difficile à implémenter compte tenu du fait que l'algorithme donné dans le TP n'était pas suffisamment clair pour nous, au début. Après quelques difficultés, nous avons pu trouver une solution.
Sur un test de 100 grilles : on n'en résout. Toutefois nous pensons que la solution implémentée est correcte et respecte le principe du hill-climbing.

Algorithme Question 5

Hill climbing combinée avec le Simulated annealing a été aussi un peu compliqué à comprendre. Nous avons dû chercher plusieurs description de l'algorithme dont celle qui a été fournie dans le devoir afin de mieux l'appréhender (sources citées plus bas). Pour les performances nous remarquons une large amélioration (comme l'indique les figures résultats). Sur un set de 100, ça résous plus de la moitié.

(Sources:
https://www.researchgate.net/publication/319886025_b-Hill_Climbing_Algorithm_for_Sudoku_Game

https://orca.cardiff.ac.uk/id/eprint/27746/1/LEWIS%20metaheuristics%20can%20solve%20sudoku%20puzzles.pdf
)

RÉSULTATS

Question 1, 2 et 3

```
All tests pass.
Question 1. Solved 100/100 of 100sudoku.txt puzzles (avg 0.0023515 secs (425 Hz), max 0.0032914 secs).
Question 2. Solved 100/100 of 100sudoku.txt puzzles (avg 0.0026836 secs (372 Hz), max 0.0159084 secs).
Question 3. Solved 100/100 of 100sudoku.txt puzzles (avg 0.0024220 secs (412 Hz), max 0.0035705 secs).
PS G:\My Drive\Ecole\Cette Session\8. H24\IFT 3335\Devoir\TP1>
```

Question 4 Hill Climbing

```
All tests pass.
Solved 0 of 100  puzzles (avg 0.04 secs (28 Hz), max 0.06 secs).
```

Question 5 Simulated Annealing

```
 Best conflict: 0
 Solved 50 of 100  puzzles (avg 1.13 secs (0 Hz), max 1.61 secs).
 PS C:\Users\Narah\Desktop>
```