IFT3335-TP1-Rapport

Wenhao Xu, 20150702
Mingze Li, 20150696
Yu Deng, 20151659

Tâche 1 et Tâche 2

On test dans le programme **sudoku_final.py**, avec les appels suivants:

solve_all(from_file("100sudoku.txt"),"easy", None), ce qui est un appel du programme de Norvig original
solve_all_pur_profondeur(from_file("100sudoku.txt"),"easy", None), ce qui est un appel à une recherche pure en profondeur d'abord sans aucune heuristique.
Voici est le résultat:

```
[Running] python -u "/Users/kironrothschild/Desktop/A2022/IFT3335/IFT3335-A-A22/TP1/sudoku_final.py"
All tests pass.
Solved 100 of 100 easy puzzles (avg 0.00 secs (539 Hz), max 0.00 secs).
Solved 100 of 100 easy puzzles (avg 0.00 secs (512 Hz), max 0.01 secs by Deepth First Search).

[Done] exited with code=0 in 0.503 seconds
```

Après plusieurs exécutions différentes du programme, les performances de la recherche non heuristique DFS se dégradent à chaque fois par rapport à la recherche initiale, même si les valeurs ne sont pas exactement les mêmes à chaque fois.
Nous pouvons en déduire que pour le deuxième code(la recherche non heuristique DFS), nous avons besoin d'un peu plus de temps, même si la différence n'est pas très grande, mais cela montre aussi l'importance de l'heuristique.

Tâche 3

Dans la tâche 3, on a implémenté avec succès les heuristiques Naked Pairs/Triples & Hidden Pairs/Triples
Avec l'appel **solve_all_heuristique(from_file("100sudoku.txt"),"easy", None)**, on a obtenu le résultat:

```
[Running] python -u "/Users/kironrothschild/Desktop/A2022/IFT3335/IFT3335-A-A22/TP1/sudoku_final.py"
All tests pass.
Solved 100 of 100 easy puzzles (avg 0.00 secs (504 Hz), max 0.00 secs).
Solved 100 of 100 easy puzzles (avg 0.00 secs (502 Hz), max 0.01 secs by Deepth First Search).
Solved 100 of 100 easy puzzles (avg 0.00 secs (520 Hz), max 0.00 secs by Heuristics Search).

[Done] exited with code=0 in 0.648 seconds
```

On peut voir que, bien que l'ampleur soit faible, les performances de l'algorithme se sont quelque peu améliorées par rapport à l'algorithme initial.
On peut voir que, bien que l'ampleur soit faible, les performances de l'algorithme se sont quelque peu améliorées par rapport à l'algorithme initial. Cette amélioration modeste peut être due au fait que le cas Naked Pairs/Triples & Hidden Pairs/Triples n'est pas toujours satisfait, et que si l'heuristique utilisée est d'une grande complexité et n'est pas optimisée pour les performances (le cas où la condition n'est pas satisfaite). Il y a également un certain degré de perte de performance.

Tâche 4 et Tâche 5

Voici le résultat de l'appel
solve_all_hill_climbing(from_file("100sudoku.txt"),"easy", None):

```
[Running] python -u "/Users/kironrothschild/Desktop/A2022/IFT3335/IFT3335-A-A22/TP1/sudoku_final.py"
All tests pass.
Solved 100 of 100 easy puzzles (avg 0.00 secs (505 Hz), max 0.00 secs).
Solved 100 of 100 easy puzzles (avg 0.00 secs (504 Hz), max 0.01 secs by Deepth First Search).
Solved 100 of 100 easy puzzles (avg 0.00 secs (527 Hz), max 0.00 secs by Heuristics Search).
Solved 45 of 100 easy puzzles (avg 0.01 secs (126 Hz), max 0.02 secs by hill climbing).

[Done] exited with code=0 in 1.43 seconds
```

Il est facile de voir que pour l'algorithme Hill-Climbing, il y a une baisse significative du taux de réussite et des performances. Cela est dû aux limites de l'algorithme Hill-Climbing, qui produit une solution localement optimale mais pas nécessairement une solution globalement optimale. Il est donc possible que nous n'obtenions pas de réponse au Sudoku.
Il est facile de voir que pour l'algorithme Hill-Climbing, il y a une baisse significative du taux de réussite et des performances. Cela est dû aux limites de l'algorithme Hill-Climbing, qui produit une solution localement optimale mais pas nécessairement une solution globalement optimale. Il est donc possible que nous n'obtenions pas de réponse au Sudoku. Une optimisation de l'algorithme Hill-Climbing est l'algorithme Simulated-Annealing. Cependant, étant donné que l'algorithme Simulated-Annealing est très complexe, qu'il entraîne toujours une perte de performance significative (par rapport à l'algorithme initial) et qu'il nécessite l'utilisation de l'algorithme Hill-Climbing, nous n'avons finalement pas implémenté l'algorithme Simulated-Annealing.