Devoir 1 IFT3335
Report - Sudoku

Steve Levesque
Département d'informatique et
de recherche opérationnelle
Université de Montréal
steve.levesque@umontreal.ca

Weiyue Cai
Département d'informatique et
de recherche opérationnelle
Université de Montréal
weiyue.cai@umontreal.ca

Abstract

This report will go through multiple algorithms to solve Sudokus. The soduku grid has a dimension of 9 by 9 with a configuration of 9 numbers (1 to 9) with some already assigned to the board that cannot be modified. The algorithms will be tested on 100 sudokus in a file represented by a string (N strings for N sudokus, here N=100) of 81 numbers from 1 to 9 respectively and 0 or . (dot) for empty spaces.

1 Introduction

This document will provide explainations where needed with an overall benchmark of all algorithms. Hill-Climbing will have 2 algorithms entries because we both (Steve and Weiyue) made a version that we deemded representing what was asked. Both are implemented differently, but with the same goal. We both discussed to gather toughts for a common conclusion about Hill-Climbing, which will be discussed in its own section.

List of algorithm :
- Norvig's default configuration
- Replacing Norvig's 3rd criteria with random choosing (pure DFS)
- Heuristic: Naked Pairs
- Hill-Climbing (greedy search) - Steve
- Hill-Climbing (greedy search) - Weiyue

2 Implemented Heuristic (Question 3)

The heuristic chosen is Naked Pairs. Here is the explanation of this heuristic function described in the article written by by Angus Johnson.

If two cells in a group (each row, each column or each box) contain an identical pair of candidates and only those two candidates, then no other cells in that group could be those values. These 2 candidates can be excluded from other cells in the group.

In the example below, the candidates 6 and 8 in columns six and seven form a Naked Pair within the row. Therefore, since one of these cells must be the 6 and the other must be the 8, candidates 6 and 8 can be excluded from all other cells in the row (in this case just the highlighted cell).

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

In our implementation, we did this process for each box, column and row by using the most recently updated value. Even though this heuristic method can be effective by excluding naked pairs, it can also increase time complexity when checking each group of sudoku. This can explain why this method may cost more time than pure DFS. See section 4 for more details.

3 Hill-Climbing algorithm (Question 4)

Two versions of Hill-Climbing has been implemented by one of us respectively (2) to see if it was possible to make it in a way that it can work. Here is a description of each algorithm:

- **Steve**: Converting the grid data into a list of list with values containing random numbers and removing all hard values (those already on the board). There are 9 lists in total in the main list which represent a box, each having random assigned values by order of square. (i.e. if A1 is a hard square and A2, A3, B1 are not, then the list will have 3 values where index 0 equals A2, index 1 A3, etc.)

After, we swap the numbers by having a full combination possibility of each list, that represents swaps since index are sensible of their respective soft squares.

Exemple: [1,2,3] -> [1,3,2] -> [3,2,1] -> ...

This is done on a local (box) side since each list represents a box of 9 squares. However, this version of Hill-Climbing accepts that all possibilities can be testing with different boxes and that a backtracking which causes higher conflict cost is acceptable.

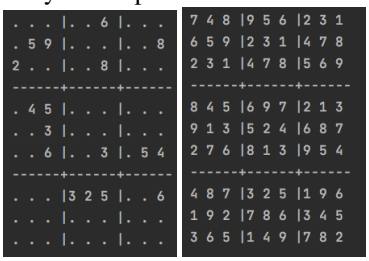
This view makes an awful lot of combinations that are impossible to solve in a respectable time theorically. But, since we have every possible solutions of swaps, there is 1 which is the potential solution. We have the chance as well that random assignation gives us all the good numbers at the right squares. (very unlikely)

However this method is flawed because it can technically go back at a higher cost for "the greater good" by checking all possible swaps of soft squares within a box.

- **Weiyue**: step1: consider sudoku as 3x3 boxes, initialize sudoku and ensure that there is no conflict in each box; step2: for each box, swap two of the filled digits (which were '0' or '.' in the original grid); step3: calculate the global conflict for each swap; if new conflict is smaller than the previous one, we keep this swap; if not, we withdraw this swap.

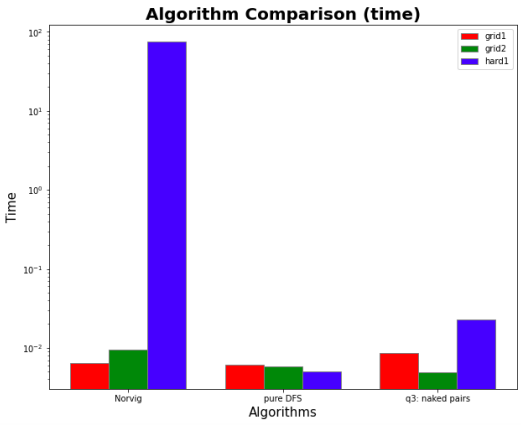
We can see that simple hill climbing can only find the local optima (the best status of each box) instead of minimal conflict of whole sudoku.

Figure 1: Result of hard1 obtained by Weiyue's implementation.



After consultation of both solutions, we came to a conclusion that Hill climbing algorithm can not solve 9x9 sudoku for the following reasons:
(1) Hill climbing can only find local optima, which are not necessarily the best possible solution (the global optimum) out of all possible solutions. [1]
(2) Hill climbing does not quite suffice to solve the puzzle. The problem comes from the completely random way in which the variables are filled in. It is also impossible for Random Restart Hill Climbing algorithm to solve 8x8 sudoku, [2] which can support our idea that simple hill climbing can not solve 9x9 sudoku.

[1] https://en.wikipedia.org/wiki/Hill_climbing
[2] https://www.cs.rochester.edu/u/brown/242/assts/termprojs/Sudoku09.pdf



	100 - Norvig	100 - pure DFS	100 - q3 naked pairs
count	100.000000	100.000000	100.000000
mean	0.004909	0.005989	0.006300
std	0.001350	0.002380	0.003241
min	0.003784	0.003835	0.003816
25%	0.004024	0.004335	0.003973
50%	0.004519	0.005073	0.005343
75%	0.005237	0.006995	0.007255
max	0.010928	0.019127	0.022458

Figure 2: Left - 3 default sudokus, Right - With 100 sudoku file

4 Comparison of all algorithms

Since hill climbing algorithm can not solve 9x9 sudoku, we can conclude that the success rate is 0 for both two implementations.

This section will focus on the comparison of performances of Norvig's heuristic, pure DFS and naked pairs heuristic. Firstly we ran several tests on grid1, grid2 and hard1 with these three methods.

We can observe that Norvig's heuristic function (based on minimum remaining values) is not a good option to solve hard1 sudoku. Surprisingly pure DFS takes less time on hard1 compared to grid1 and grid2.

To obtain a more general conclusion about the performance of these three methods, we ran tests on 100 start configurations. Here is the brief conclusion:

Algorithm	success rate	total time
minimum remaining values (Norvig)	100%	0.491
pure DFS (Q2)	100%	0.6
naked pairs (Q3)	100%	0.781

We can observe that minimum remaining values heuristic is more performed generally.

Because all these three methods use constraint propagation as basic algorithm for the search, we only analyze the complexity of heuristic used: (1) minimum remaining values (Norvig), complexity of min() is O(n); (2) pure DFS: O(1); (3) naked pairs: O(n^3) (worst case if we have to iterate all groups).

Even though heuristic function can be generally useful, we have to consider the trade off between the increase of complexity caused by heuristic function and the decrease of complexity of search function.

References

[1] Norvig's Sudoku - Report
`http://www.norvig.com/sudoku.html`

[2] Norvig's Sudoku - Full Code
`http://www.norvig.com/sudopy.shtml`

[3] Solving Sudoku
`http://www.angusj.com/sudoku/hints.php`