Karl Eknefelt karek590
Oskar Johannesson oskjo806

TNM096 Lab 2

Task 1 - Which of the algorithms DFGS/AC3/MC/BT work for solving the sudoku in a timely manner?

Table 1: Comparison of Sudoku solvers

	DFGS	AC3	min_conflicts	BT	BT+MRV	BT+FC	BT+MRV+FC
Easy	Fail	0.03	77.8	40	too slow	0.02	0.02
Harder	too slow	fail	fail	too slow	too slow	81	0.03
Hardest	too slow	fail	fail	too slow	too slow	76	0.02

a) Why?

- DFGS does not check if assignments are legal and simply checks all combinations, there are way too many combinations, and not enough memory to store the tree.
- AC3 should work since it, we're not sure why it fails. Does the order matter?
- min_conflicts generates too many conflicts in the harder sudoku boards, and it can not correct it in the given number of steps.
- BT is a more efficient version of DFGS. It only generates nodes that don't violate the constraints. Furthermore, if no legal child nodes can be produced, the parent assignment is removed from the board state. It has a lower average branching factor.

In conclusion, there is no obvious winner (except for the modified BT algorithms which come later). AC3 is the fastest but fails on the harder puzzles. BT is much slower but might be able to solve the harder puzzles given enough time.

b) Which of the BT configurations work best?

- BT + MRV can not even solve the easy puzzle.
- BT + FC is much faster, and it's able to solve the harder Sudoku boards.
- **BT + MRV + FC** is much faster on all boards.

Task 2 - N-Queens

a) How large can n be for each of the algorithms?

DFTS and BT become very slow around n=30.
AC3 always fails.
min_conflicts can solve n>1000 in just a few seconds, given enough steps.

b) What Backtracking search settings work the best? Why?

From table 2: **BT+MRV+FC** is the fastest and works best

Karl Eknefelt karek590
Oskar Johannesson oskjo806

c) How many steps does Min-Conflicts require to do its work?

To solve all puzzles up to n=30, a few hundred steps are enough.

d) Compare the nature of the heuristics deployed in traditional state-based search and constraint-based problem solving

How are we supposed to answer this question? Specifically?
- State-based solvers iterate through many possible solutions and sort unvisited potential solutions intelligently by some metric (e.g manhattan distance).
- CSP solvers utilize defined constraints to avoid paths that can't lead to a solution.
- For the 8-bit puzzle, the solution is known, and the game is to reach the solution in the least number of moves.
- For the Sudoku, the goal is to find the solution, which is not known at the start. Therefore it's difficult to model a heuristic.
- For other problems, there might not even be a solution, but you want to come as close to it as possible. These problems are also more easily modeled as a CSP.

Table 2: Comparison of N-Queens solvers

n	tree search	min_conflicts	AC3	BT	BT+FC	BT+MRV	BT+FC+MRV
10	0	0	Failure	0	0	0	0
11	0	0	Failure	0	0	0	0
12	0	0	Failure	0	0	0.01	0
13	0	0	Failure	0	0	0	0
14	0.07	0	Failure	0.02	0.01	0.03	0
15	0.05	0	Failure	0.03	0.02	0	0
16	0.41	0	Failure	0.15	0.12	0.02	0
17	0.24	0	Failure	0.14	0.07	0.02	0
18	2.14	0	Failure	0.62	0.37	0.01	0
19	0.14	0	Failure	0.06	0.03	0.18	0
20	12.07	0	Failure	3.34	1.92	0	0
21	0.54	0	Failure	0.17	0.11	0	0
22	118.37	0	Failure	30.53	15.21	0.04	0
23	1.76	0	Failure	0.44	0.26	0.01	0
24	32.5	0	Failure	8.33	4.45	0	0
25	4.06	0	Failure	0.86	0.62	0.01	0
26	34.6	0	Failure	7.52	3.99	0.17	0
27	40.32	0	Failure	9.28	4.71	0.09	0
28	300.14	0	Failure	64.19	33.88	0.05	0
29	158.33	0	Failure	37.89	17.93	0.01	0