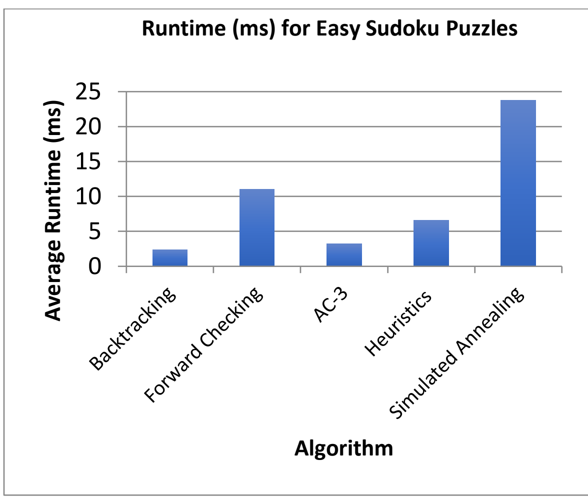
CS580 – Final Project Report

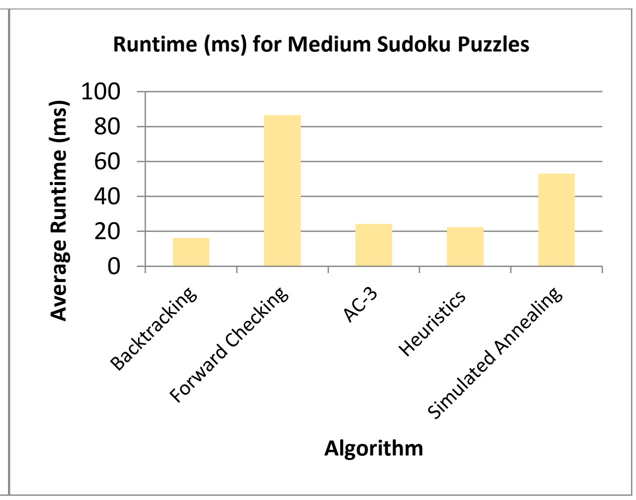
Team Members:

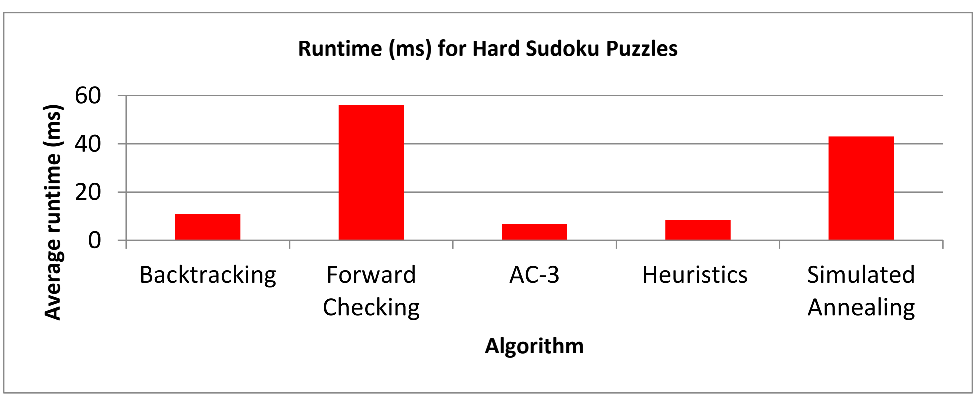
1. Joshua Lilly
2. Prasanna Venkatesh Parthasarathy
3. Siddi Avinash Chenmilla
4. Shubham Pudi
5. Bedor

1. Algorithms and Running Time:

	EASY			MEDIUM			HARD		
	Slowest (ms)	Fastest (ms)	Average (ms)	Slowest (ms)	Fastest (ms)	Average (ms)	Slowest (ms)	Fastest (ms)	Average (ms)
Backtracking	3	2	2.4	44	3	16.2	22	4	11
Forward Checking	15	6	11	191	16	86.6	88	26	56
AC-3	4	3	3.2	54	5	24.2	15	3	6.8
Heuristics - Vertex and Value Ordering	9	5	6.6	30	9	22.4	16	5	8.4
Simulated Annealing	30	19	23.8	53	53	53	43*	43*	43







Based on our data, which algorithm to use for Sudoku varies depending on the difficulty. For easy and medium puzzle, we found that the fastest algorithm on average was backtracking. We showed that on average for easy puzzles backtracking just slightly outperformed AC3, but this was not the case for the medium puzzle. In the medium puzzle AC3 dropped significantly, but backtracking was still the best algorithm, in fact the vertex order heuristic out performed AC3 for medium puzzles. For the hard puzzle, AC3 was the dominant algorithm by a significant amount outperforming backtracking and vertex order. Unfortunately, for medium and hard puzzles Simulated annealing typically did not provide a solution in the given time.

2. Problem Formulation and Algorithm Description:

Sudoku:

A Sudoku board with Grid Size 'N' has N rows and N columns with a total of N*N cells. Each cell can be assigned with values from 1 to N.

Cell: A cell is a 'square' in a Sudoku grid.
Grid: A grid represents the Sudoku board.
Peers: All the cell's neighbors; neighbors are cells that are in the same unit of the cell.
Unit: A collection of cells, for each row, column and the region which is of size sqrt(N) x sqrt(N) size.

Sudoku as CSP:

Variables: Each cell in the Grid ranging from 1 to (N*N).

Domain: The domain is any digit ranging from 1 to N.

Constraints: The constraints are:

- Same digit can't appear more than once in the same row.
- Same digit can't appear more than once in the same column.
- Same digit can't appear more than once in the same region.

Forward Checking:

Forward checking is the method of finding and eliminating the list of possibilities that do match the constraints from the domains of unassigned variables, in advance. Forward checking essentially allows to detect the failure earlier thus resulting in an efficient backtracking and reduced search tree size.

Our implementation of Forward Checking uses a HashMap of Cells to List of Possible Assignments for all unassigned cells. Upon assigning a domain value to a Cell, the map is consulted for detecting any unassigned cell that will run out of possible assignments and does backtracking if there is one. Also, constraints are re-evaluated for the cell and its peers and the map is updated with that information. This allows to detect an upcoming failure and try different possible assignments to mitigate it.

AC-3:

AC3 was helpful because it reduced the domain on the problem that was considered for the backtracking algorithm. That is, whenever backtracking may have to try nine different values for a square, removing inconsistent values allows almost all squares to have less feasible values for any given square. We were able to remove values from the domain due to the given numbers in the puzzle. However, even given this reduction, it was usually the case that backtracking with no heuristic was better than AC3. This is since removing inconsistent edges in the consistency graph is still a time intensive operation.

Simulated Annealing:

The setup of Sudoku for simulated annealing was to use a two-dimensional array for the Sudoku board. We then used the board to randomly generate values inside of each cell. Then we calculated the number of violations in the board. Then if the new configuration is better than the previous we keep the new configuration. If the configuration is worse, we keep it within some probability, which is based on randomly generating a number. Then we generate the neighboring solutions and count the violations. While the number of violations is not zero, we continue this process, for the board we just generated. The temperature 1.5 and alpha is set to 0.99999, then the cooling rate was defined to be temperature is equal to itself times alpha. We chose alpha and temperature from empirically realizing that the solver has a better chance of finding the solution if the temperature-cooling rate is slower. We tried other values, however, in our code these values seemed to provide better results.

3. Comments on Improvements:

For the easy and medium puzzles, the best algorithm was backtracking. To improve backtracking there are numerous things that in this project we implemented to improve performance for it. However, many of the improvements were marginal at best. However, for all algorithms, something we could do better in our code is trying to be cleverer with the algorithms in a way where we don't have to make as many copies of nodes. Additionally, one thing that could potentially improve performance is doing inconsistency removal in parallel. For example, perhaps a better solution could be obtained by concurrently removing inconsistencies for a sub problem, like a row or column concurrently with other sub problems on the board. We could potentially break the board up into numerous sub problems and have worker threads each preprocess a section of the board. Of course, this is still subject to other costs, such as the locking of a mutex and combining the sub problems together. We could potentially use a highly concurrent framework such as OpenCL or CUDA. Additionally, we hypotensive that Simulated Annealing would improve if we could do multiple instantiations of Simulated Annealing and give every instance its own thread. This would allow multiple workers to simultaneously work on solving the same problem and could improve results.

4. Detailed Report:

EASY TEST CASES:

<table>
<tr><td rowspan="5">Easy1</td><td rowspan="5">0 0 3 0 2 0 6 0 0
9 0 0 3 0 5 0 0 1
0 0 1 8 0 6 4 0 0
0 0 8 1 0 2 9 0 0
7 0 0 0 0 0 0 0 8
0 0 6 7 0 8 2 0 0
0 0 2 6 0 9 5 0 0
8 0 0 2 0 3 0 0 9
0 0 5 0 1 0 3 0 0</td><td>Backtracking</td><td>2</td></tr>
<tr><td>Forward Chaining</td><td>15</td></tr>
<tr><td>AC-3</td><td>3</td></tr>
<tr><td>Vertex Ordering</td><td>9</td></tr>
<tr><td>Simulated Annealing</td><td>30</td></tr>
</table>

<table>
<tr><td rowspan="5">Easy2</td><td rowspan="5">0 0 0 0 9 0 0 0 3
0 0 6 0 2 0 0 0 0
7 3 0 4 0 5 2 1 0
0 2 0 3 8 6 0 0 0
0 4 5 0 0 0 3 6 0
1 6 0 0 0 4 8 9 0
6 7 0 0 0 0 0 3 1
3 8 0 2 0 0 9 7 0
0 0 0 6 3 7 0 2 0</td><td>Backtracking</td><td>2</td></tr>
<tr><td>Forward Chaining</td><td>11</td></tr>
<tr><td>AC-3</td><td>3</td></tr>
<tr><td>Vertex Ordering</td><td>5</td></tr>
<tr><td>Simulated Annealing</td><td>22</td></tr>
</table>

<table>
<tr><td rowspan="5">Easy3</td><td rowspan="5">0 2 0 3 0 0 6 0 9
0 0 5 0 6 4 0 0 7
0 1 6 0 7 0 0 0 0
0 0 0 0 9 0 0 3 6
6 7 0 0 0 1 0 0 5
0 3 0 8 5 0 0 7 0
0 8 9 0 1 3 0 0 0
0 0 0 5 2 0 0 6 1
0 6 1 9 0 7 0 5 8</td><td>Backtracking</td><td>2</td></tr>
<tr><td>Forward Chaining</td><td>6</td></tr>
<tr><td>AC-3</td><td>3</td></tr>
<tr><td>Vertex Ordering</td><td>6</td></tr>
<tr><td>Simulated Annealing</td><td>24</td></tr>
</table>

<table>
<tr><td rowspan="5">Easy4</td><td rowspan="5">0 0 3 8 0 0 0 2 0
1 0 0 0 0 2 8 0 0
0 0 7 0 3 0 0 9 0
0 6 0 0 0 0 4 0 0
0 1 9 0 5 4 2 0 8
0 4 8 9 6 7 0 1 0
0 0 4 7 8 1 0 5 2
5 0 0 0 0 0 0 0 1
6 7 0 4 0 5 0 8 3</td><td>Backtracking</td><td>3</td></tr>
<tr><td>Forward Chaining</td><td>8</td></tr>
<tr><td>AC-3</td><td>3</td></tr>
<tr><td>Vertex Ordering</td><td>6</td></tr>
<tr><td>Simulated Annealing</td><td>19</td></tr>
</table>

<table>
<tr><td rowspan="5">Easy5</td><td rowspan="5">9 0 5 0 4 0 0 0 1
1 0 0 0 0 9 0 4 0
7 6 0 1 0 0 9 0 0
0 2 0 0 0 7 8 0 9
8 7 0 0 5 0 0 6 0
0 1 3 9 8 6 0 0 0
3 0 8 0 0 5 2 0 0
0 0 1 7 9 0 0 8 0
0 4 7 0 0 0 5 9 6</td><td>Backtracking</td><td>3</td></tr>
<tr><td>Forward Chaining</td><td>15</td></tr>
<tr><td>AC-3</td><td>4</td></tr>
<tr><td>Vertex Ordering</td><td>7</td></tr>
<tr><td>Simulated Annealing</td><td>24</td></tr>
</table>

MEDIUM TEST CASES

<table>
<tr><td rowspan="5">1

Medium1</td><td rowspan="5">0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 5 3 7
9 0 0 0 0 5 8 0 0
0 2 0 0 6 0 0 0 0
0 8 5 0 3 0 2 0 0
0 6 0 0 0 4 0 0 9
0 0 4 7 0 0 0 0 5
0 0 0 1 0 0 3 2 6
0 0 0 0 0 0 0 0 0</td><td>Backtracking</td><td>24</td></tr>
<tr><td>Forward Chaining</td><td>180</td></tr>
<tr><td>AC-3</td><td>19</td></tr>
<tr><td>Vertex Ordering</td><td>42</td></tr>
<tr><td>Simulated Annealing</td><td>53</td></tr>
</table>

<table>
<tr><td rowspan="5">Medium 2</td><td rowspan="5">0 0 0 0 0 0 0 0 0
0 0 0 0 0 5 8 9 1
6 0 0 0 0 3 0 2 0
0 0 0 0 0 0 0 0 0
0 0 0 2 0 0 7 0 0
0 5 0 0 0 0 2 3 8
0 0 9 6 0 0 0 4 0
0 0 1 0 9 0 0 0 0
2 0 7 0 8 0 1 0 0</td><td>Backtracking</td><td>3</td></tr>
<tr><td>Forward Chaining</td><td>23</td></tr>
<tr><td>AC-3</td><td>37</td></tr>
<tr><td>Vertex Ordering</td><td>17</td></tr>
<tr><td>Simulated Annealing</td><td>53*</td></tr>
</table>

<table>
<tr><td rowspan="5">Medium 3</td><td rowspan="5">0 0 0 2 0 3 0 8 0
1 0 0 0 0 0 0 0 4
0 9 5 6 0 8 0 0 0
0 2 0 0 3 0 0 0 0
0 0 0 5 0 0 1 7 0
4 0 7 0 0 0 6 5 0
3 1 0 0 0 0 0 0 0
0 0 6 0 7 0 0 0 0
0 0 8 0 0 0 0 0 5</td><td>Backtracking</td><td>3</td></tr>
<tr><td>Forward Chaining</td><td>16</td></tr>
<tr><td>AC-3</td><td>5</td></tr>
<tr><td>Vertex Ordering</td><td>9</td></tr>
<tr><td>Simulated Annealing</td><td>53*</td></tr>
</table>

<table>
<tr><td rowspan="5">Medium 4</td><td rowspan="5">0 0 0 0 2 0 0 8 0
9 0 1 3 0 7 0 0 0
3 0 6 0 0 0 0 0 9
0 0 0 1 0 0 0 3 0
0 0 0 0 6 8 0 0 0
0 9 0 5 0 0 0 0 0
5 0 0 0 0 0 8 0 2
0 7 0 0 0 6 0 0 0
0 0 0 9 4 0 5 0 1</td><td>Backtracking</td><td>4</td></tr>
<tr><td>Forward Chaining</td><td>23</td></tr>
<tr><td>AC-3</td><td>6</td></tr>
<tr><td>Vertex Ordering</td><td>14</td></tr>
<tr><td>Simulated Annealing</td><td>53*</td></tr>
</table>

<table>
<tr><td rowspan="5">Medium 5</td><td rowspan="5">5 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 6 9
0 0 7 9 0 0 0 0 3
0 0 5 0 0 0 0 0 0
6 0 0 0 0 5 7 0 0
0 0 0 0 3 8 0 1 0
3 0 0 0 0 0 1 9 0
0 0 6 2 9 0 0 0 0
9 0 2 3 0 0 5 0 0</td><td>Backtracking</td><td>44</td></tr>
<tr><td>Forward Chaining</td><td>191</td></tr>
<tr><td>AC-3</td><td>54</td></tr>
<tr><td>Vertex Ordering</td><td>30</td></tr>
<tr><td>Simulated Annealing</td><td>53*</td></tr>
</table>

HARD TEST CASES

<table>
<tr><td rowspan="5">Hard 1</td><td rowspan="5">0 0 0 3 6 1 2 0 0
7 0 0 0 0 0 0 4 0
8 0 0 0 2 0 0 0 0
0 7 5 0 0 0 0 0 0
0 0 0 0 0 0 0 5 8
0 0 0 0 0 0 4 0 1
0 0 0 0 8 0 0 6 0
0 1 0 2 3 4 0 0 0
9 0 0 0 0 0 0 7 0</td><td>Backtracking</td><td>11</td></tr>
<tr><td>Forward Chaining</td><td>30</td></tr>
<tr><td>AC-3</td><td>15</td></tr>
<tr><td>Vertex Ordering</td><td>6</td></tr>
<tr><td>Simulated Annealing</td><td>43*</td></tr>
</table>

<table>
<tr><td rowspan="5">Hard 2</td><td rowspan="5">0 0 3 2 0 0 0 0 0
9 6 0 0 4 0 0 1 0
0 0 0 0 6 5 0 0 0
2 0 9 0 0 0 0 0 0
3 0 8 0 0 0 0 0 0
0 0 0 0 1 0 0 0 4
0 0 0 0 0 0 0 9 5
1 0 0 0 0 0 4 0 0
0 7 2 0 0 3 0 0 8</td><td>Backtracking</td><td>22</td></tr>
<tr><td>Forward Chaining</td><td>59</td></tr>
<tr><td>AC-3</td><td>5</td></tr>
<tr><td>Vertex Ordering</td><td>5</td></tr>
<tr><td>Simulated Annealing</td><td>43*</td></tr>
</table>

<table>
<tr><td rowspan="5">Hard 3</td><td rowspan="5">0 5 0 0 0 3 0 0 0
0 6 2 0 0 5 1 0 0
0 0 0 0 9 8 0 4 0
0 3 4 0 0 6 8 0 0
0 0 0 0 2 0 4 6 0
0 0 9 0 0 0 3 0 0
0 0 0 0 0 0 0 7 8
0 0 0 0 0 9 2 0 0
0 0 0 4 5 0 0 0 0</td><td>Backtracking</td><td>4</td></tr>
<tr><td>Forward Chaining</td><td>26</td></tr>
<tr><td>AC-3</td><td>6</td></tr>
<tr><td>Vertex Ordering</td><td>16</td></tr>
<tr><td>Simulated Annealing</td><td>43*</td></tr>
</table>

<table>
<tr><td rowspan="5">Hard 4</td><td rowspan="5">0 0 0 5 0 0 0 2 0
0 4 0 7 0 0 0 0 1
0 0 5 0 0 0 3 0 0
7 0 0 0 1 0 8 0 0
9 0 4 0 0 3 0 0 5
0 0 1 0 0 0 6 0 4
0 0 7 0 0 5 0 3 0
0 6 0 0 0 2 0 0 0
2 0 0 0 0 0 0 0 9</td><td>Backtracking</td><td>11</td></tr>
<tr><td>Forward Chaining</td><td>88</td></tr>
<tr><td>AC-3</td><td>5</td></tr>
<tr><td>Vertex Ordering</td><td>9</td></tr>
<tr><td>Simulated Annealing</td><td>43*</td></tr>
</table>

<table>
<tr><td rowspan="5">Hard 5</td><td rowspan="5">0 7 0 8 0 0 9 0 0
0 0 3 0 0 6 0 0 0
0 0 0 0 1 0 6 0 0
0 0 5 0 2 0 0 0 8
0 6 0 0 0 0 0 0 9
7 0 0 9 0 0 0 0 0
0 0 9 2 0 3 7 0 0
1 0 2 5 0 0 0 0 0
4 0 0 0 0 8 0 5 0</td><td>Backtracking</td><td>7</td></tr>
<tr><td>Forward Chaining</td><td>77</td></tr>
<tr><td>AC-3</td><td>3</td></tr>
<tr><td>Vertex Ordering</td><td>6</td></tr>
<tr><td>Simulated Annealing</td><td>43*</td></tr>
</table>