# Homework #3 Constraint Satisfaction

# Problem Solver Experiment Results

**Experiment Introduction:**

The CSP solver program was run on two primary types of constraint satisfaction problems (CSP). The first is the tri-coloring problem for the states and territories in Australia.[[1]](#footnote-1) The second type of problem is Sudoku, which is substantially more complex problem than map tri-color. Below is a summary of the problem scopes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem** | | **Number of Variables** | **Domain Size** | **Number of Binary Constraints** | **Search**  **Tree Size[[2]](#footnote-2)** |
| Australia Tri-Coloring | | 7 | 3 | 9 | 2187 |
| Sudoku | 4x4 | 16 | 4 | 52 | 4.295 \* 109 |
| 9x9 | 27 | 9 | 810 | 5.815 \* 1025 |
| 16x16 | 256 | 16 | 4992 | 1.798 \* 10308 |

Six Sudoku puzzles were used. The first puzzle is one of the hardest 2x2 puzzles where only 4 spaces are labeled. The remaining 5 puzzles were 9x9 and were separated into five categories depending on the puzzle’s level of difficulty. The five categories were: Easiest, Easy, Medium, Hard, and Hardest.[[3]](#footnote-3) Appendix A contains the Sudoku puzzles used.

**Experiment Metrics:**

Three primary metrics were used to assess the time complexity of the CSP solver on a given problem. They are:

1. **Backtrack Function Call Count** – This quantifies the number of times the “Backtrack” function is called. If the backtracking algorithm executed with optimal efficiency, this number would equal the number of variables in the CSP. The greater the difference between this metric and the number of variables indicates the algorithm needed to do more searching.
2. **Variable Value Setting Count** – This quantifies the number of times any variable in the CSP was assigned a value; this count is incremented irrespective of whether the value setting is consistent or inconsistent. Using the pseudocode on page 215 of Russell and Norvig, this metric corresponds to the number of times the *for* loop is run.
3. **Algorithm Runtime** – This quantifies in seconds how long the CSP solver took to find a solution. This metric is subject to variation due to computer system variation and is only a rough estimate of algorithm performance. This is measured in seconds.

If the backtracking algorithm executes on a CSP with optimal efficiency, this backtrack function call and variable value setting counts would equal the number of variables in the CSP. The greater the difference between these metrics and the number of variables indicates the algorithm needed to do more searching.

**Experiment Results:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Metric** | **Type** | **Australia** | **Hardest**  **4x4 Sudoku** | **Easiest**  **9x9 Sudoku** | **Easy 9x9 Sudoku** | **Medium**  **9x9 Sudoku** | **Hard 9x9 Sudoku** | **Hardest**  **9x9 Sudoku** |  |
| **Backtrack Function Call Count** | With FC | 7 | 16 | 81 | 132 | 341 | 286 | 515 |  |
| No FC | 7 | 42 | 777 | N/A | N/A | N/A | N/A |  |
| **Variable Value Setting Count** | With FC | 7 | 16 | 81 | 138 | 370 | 313 | 570 |  |
| No FC | 14 | 138 | 6487 | N/A | N/A | N/A | N/A |  |
| **Algorithm**  **Runtime (s)** | With FC | 0.000 | 0.000 | 0.047 | 0.079 | 0.188 | 0.141 | 0.250s |  |
| No FC | 0.001 | 0.001 | 0.669 | N/A | N/A | N/A | N/A |  |

Table - CSP Solver Experiment Results With and Without Forward Checking

Table 1 contains the experimental results comparing the efficiency of the CSP solver with and without forward checking. Without forward checking, the CSP solver was only able to solve the most trivial puzzles (e.g. map of Australia, the hardest 4x4 Sudoku, and the easiest 9x9 Sudoku). When the CSP solver was run on the next most difficult 9x9 Sudoku (i.e. easy), the algorithm ran for over 12 hours without solving the problem. For that reason, no results are provided on those problems without forward checking.

When forward checking was used, it was able to solve all puzzles up to 9x9 Sudoku. What is more, it was always faster and more efficient (for both Backtrack function calls and variable value settings) even with the overhead of running the forward checks. This is because forward checking prunes (i.e. reduces) the search tree substantially by removing impossible solutions.

# Appendix – Sudoku Board Files

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** |
| **A** | 4 |  |  |  |
| **B** |  |  |  | 2 |
| **C** | 3 |  |  |  |
| **D** |  |  |  | 1 |

Figure – Hardest 4x4 Sudoku Puzzle

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **A** | 9 | 2 |  | 6 |  | 8 |  | 3 |  |
| **B** | 3 |  | 7 |  |  |  | 1 | 6 | 8 |
| **C** | 6 |  |  | 7 | 4 | 3 |  | 9 |  |
| **D** | 2 |  | 4 | 5 | 3 |  |  |  | 6 |
| **E** | 8 |  |  | 1 | 7 |  |  | 5 | 9 |
| **F** |  | 1 | 9 |  |  | 2 | 4 | 7 |  |
| **G** |  | 3 | 8 |  | 5 | 1 | 6 |  | 7 |
| **H** |  | 5 | 2 |  |  |  | 9 |  | 1 |
| **I** |  | 9 |  | 4 | 2 | 7 | 3 |  |  |

Figure - Easiest Difficulty 9x9 Sudoku Puzzle

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **A** |  |  |  | 3 | 2 | 7 |  | 5 | 1 |
| **B** |  |  |  |  | 8 |  |  |  | 7 |
| **C** |  |  |  | 5 |  | 9 | 4 |  |  |
| **D** |  |  | 3 |  |  |  |  | 9 | 6 |
| **E** | 7 |  | 2 | 1 | 9 | 3 | 5 |  | 4 |
| **F** | 8 | 4 |  |  |  |  | 1 |  |  |
| **G** |  |  | 7 | 8 |  | 6 |  |  |  |
| **H** | 5 |  |  |  | 3 |  |  |  |  |
| **I** | 6 | 1 |  | 9 | 5 | 2 |  |  |  |

Figure – Easy Difficulty 9x9 Sudoku Puzzle

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **A** |  | 3 | 9 |  |  |  |  |  | 6 |
| **B** |  |  |  |  | 7 |  |  |  |  |
| **C** |  |  | 1 | 5 |  | 9 |  | 7 |  |
| **D** |  |  | 3 |  |  | 7 | 1 | 6 |  |
| **E** |  |  | 8 | 3 | 5 | 6 | 2 |  |  |
| **F** |  | 6 | 5 | 4 |  |  | 3 |  |  |
| **G** |  | 9 |  | 7 |  | 8 | 6 |  |  |
| **H** |  |  |  |  | 4 |  |  |  |  |
| **I** | 2 |  |  |  |  |  | 9 | 5 |  |

Figure – Medium Difficulty 9x9 Sudoku Puzzle

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **A** | 8 |  |  |  | 4 |  |  |  | 9 |
| **B** |  | 1 | 4 |  | 6 | 8 |  |  |  |
| **C** | 3 | 6 |  | 7 |  |  |  |  |  |
| **D** |  |  |  |  |  |  | 9 |  |  |
| **E** | 2 | 9 | 7 |  |  |  | 8 | 3 | 5 |
| **F** |  |  | 5 |  |  |  |  |  |  |
| **G** |  |  |  |  |  | 3 |  | 5 | 7 |
| **H** |  |  |  | 5 | 8 |  | 6 | 9 |  |
| **I** | 5 |  |  |  | 9 |  |  |  | 4 |

Figure – Hard Difficulty 9x9 Sudoku Puzzle

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **A** | 4 | 9 |  |  |  |  | 5 | 6 |  |
| **B** |  |  |  | 9 |  | 2 |  |  |  |
| **C** |  | 6 |  |  |  |  |  |  | 8 |
| **D** | 3 |  |  | 2 | 8 |  |  | 7 | 1 |
| **E** |  |  |  |  |  |  |  |  |  |
| **F** | 9 | 8 |  |  | 3 | 7 |  |  | 4 |
| **G** | 7 |  |  |  |  |  |  | 4 |  |
| **H** |  |  |  | 3 |  | 8 |  |  |  |
| **I** |  | 1 | 9 |  |  |  |  | 8 | 2 |

Figure – Hardest Difficulty 9x9 Sudoku Puzzle

1. See file “australia.txt” for the Australia state and province tri-coloring problem. [↑](#footnote-ref-1)
2. Search tree size for Sudoku does not account for already assigned values. [↑](#footnote-ref-2)
3. The easiest puzzle came from the website: <http://www.sudokukingdom.com/very-easy-sudoku.php>. The remaining for puzzles were generated by the website: <http://www.websudoku.com/>. The difficulty classification for all puzzles was provided by the source website. [↑](#footnote-ref-3)