The Team, and How to run the code

The Team: TheWalkingZed

**Me**: Joseph Park, 24136956, [josepjp2@uci.edu](mailto:josepjp2@uci.edu)

**Partner**: None

**By turning in this assignment, I do affirm that we did not copy any code, text, or data except CS-171 course material provided by the textbook, class website, or Teaching Staff.**

**Programming Languages and Versions used:**

* C++, c++11

**Environment Setup Required to Run:**

* Compiled via MinGW 4.7.2 32-bit and Qt 5.5
* Delivered executable compiled on openlab with gcc 5.2.0, using commands:
  + –static-libstdc++
  + –static-libgcc
  + –std=c++11

**Implementation:**

* Variable & Domain – Variable is a combination of an array position and a domain. The Domain is a vector of possible character values.
* Puzzle – Holds the puzzle and its attributes as a 2-dimensional array of variables (array locations, combined with a character face value and a domain of character values)
* Generator – Generates a puzzle based on the attributes of m (number of cells pre-assigned), n (size of the puzzle), p (row size of blocks), and q (column size of blocks).
* Reader – Reads and writes puzzles from and to files.
* Solver – Solves a puzzle based on a variety of heuristic methods, each of which can be toggled on and off. It generates a log file after solving the puzzle.
* Book Keeper – Stores assignments, removals, and search depth after every assignment.
* Logger – Writes to an output file with various attributes from the solver.

What My Team DID

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **I coded it.** | | | **I tested it thoroughly.** | | | **It ran reliably and correctly.** | | | **What was it?** |
| Yes | Partly | No | Yes | Partly | No | Yes | Partly | No |
| **Required Coding Project** | | | | | | | | | |
| **X** |  |  | **X** |  |  | **X** |  |  | Backtracking Search (BT) |
| **X** |  |  | **X** |  |  |  | **X** |  | Forward Checking (FC) |
| **X** |  |  | **X** |  |  |  | **X** |  | Minimum Remaining Values (MRV) |
| **X** |  |  |  | **X** |  |  |  | **X** | Degree Heuristic (DH) |
| **X** |  |  |  | **X** |  |  |  | **X** | Least Constraining Value (LCV) |
| **Extra Credit** | | | | | | | | | |
| **X** |  |  | **X** |  |  | **X** |  |  | Writing My Own Shell |
| **X** |  |  | **X** |  |  | **X** |  |  | Writing My Own Random Problem Generator |
| **X** |  |  |  | **X** |  |  |  | **X** | Arc Consistency AC-3/ACP/MAC |
|  |  | **X** |  |  | **X** |  |  | **X** | Local Search Using Min-Conflicts Heuristic |
|  |  | **X** |  |  | **X** |  |  | **X** | Advanced Techniques, Extra Effort, or Creativity Not Reflected Above\* |

\*Advanced Techniques, Extra Effort, or Creativity Not Reflected Above:

* None.

Analysis of Best Methods Combination

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FC** | **MRV** | **DH** | **LCV** | **(ACP)** | **(MAC)** | **(OTHER)** | **Average # Nodes** | **Average Time** | **Std. Dev. Time** |
| The blank row below is for the case of no heuristics and no constraint propagation. | | | | | | | | | |
|  |  |  |  |  |  |  | 27257296 | 383.6 | 303.6345 |
| **X** |  |  |  |  |  |  | 11670074 | 410.2 | 295.0092 |
|  | **X** |  |  |  |  |  | 12654146 | 408 | 295.3625 |
|  |  | **X** |  |  |  |  | 143949.4 | 601 | 268.7754 |
|  |  |  | **X** |  |  |  | 1895153 | 482.8 | 315.7687 |
|  |  |  |  | **(X)** |  |  | 30852591 | 364.8 | 315.8975 |
|  |  |  |  |  | **(X)** |  | 6918.2 | 459.8 | 295.3159 |
|  |  |  |  |  |  | **(X)** | None | None | None |
| **X** | **X** | **X** | **X** |  |  |  | 2372823 | 482.2 | 316.4403 |
| **X** | **X** | **X** | **X** | **X** | **X** |  | 5223.6 | 395.2 | 297.1499 |
| Fill in the blank row above with the combination you found to be fastest on PH1-5. | | | | | | | | | |

Note: The timeout used in the above tests with PH1-5 was 600 seconds, or 10 minutes.

**Did you get the results you expected? Why or why not?**

As a result of a less than stellar implementation of the heuristics and propagation, these operations end up taking more time to work than it reduces in the backtracking search. The only one faster than regular backtracking search was with a single option: Arc Consistency Preprocessor. While the degree heuristic and the least constraining value heuristic increases the average solution time by a large amount (DH causing all timeouts, and LCV taking up 482.8 seconds on average), combining them with the other heuristics and propagations causes the time to drop to around 395.2 seconds, with a drastically lower average node count, 5223.6 nodes. These were more or less the results I expected. The highlighted row is the best combination, and will be used in the next 4 sections.

**Did you implement any Advanced Techniques, Extra Effort, or Creativity not reflected above? If so, please tell us what you did.**

No.

Estimation of the critical value, Hardest R

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **N** | **P** | **Q** | **R** | **Average # Nodes** | **Average Time** | **Std. Dev. Time** | **# (%) Solvable** |
| **4** | **9** | **3** | **3** | **0.0494** | 357 | 0 | 0 | 100% |
| **8** | **9** | **3** | **3** | **0.0988** | 438.5 | 0 | 0 | 100% |
| **12** | **9** | **3** | **3** | **0.148** | 2321640 | 13.2 | 39.6 | 100% |
| **16** | **9** | **3** | **3** | **0.198** | 2078.4 | 0 | 0 | 100% |
| **17** | **9** | **3** | **3** | **0.210** | 5077.1 | 0.2 | 0.6 | 100% |
| **18** | **9** | **3** | **3** | **0.222** | 12304200 | 60.2 | 120.4 | 80% |
| **19** | **9** | **3** | **3** | **0.235** | 1186000 | 60.3 | 120.35 | 80% |
| **20** | **9** | **3** | **3** | **0.247** | 16349600 | 77.4 | 122.693 | 80% |
| **21** | **9** | **3** | **3** | **0.259** | 375106 | 2.1 | 3.72793 | 100% |
| **22** | **9** | **3** | **3** | **0.272** | 3365790 | 18.2 | 42.2985 | 100% |
| **24** | **9** | **3** | **3** | **0.296** | 976741 | 4.7 | 10.8356 | 100% |
| **28** | **9** | **3** | **3** | **0.346** | 3867.6 | 0 | 0 | 100% |
| **32** | **9** | **3** | **3** | **0.395** | 160.8 | 0 | 0 | 100% |
| **36** | **9** | **3** | **3** | **0.444** | 4.5 | 0 | 0 | 100% |

**Note: Timeout for these tests were set to 300 seconds.**

**Find the critical value of the “hardest ” for and your best combination above.**

**Based upon your results above, you estimate the “hardest ” =** .

**How does puzzle solvability for your best combination vary with ?**

**Is the critical value for “hardest ” approximately the same as the value of for which a random puzzle is solvable with probability ?**

From the sample population of the tests I have conducted, the “hardest R” value estimated from the first graph coincides with the critical point(plateau) in this graph, 0.8. It is not the same value of R for a probability of 0.5 (as there is none from my sample), but it is as close as it can get to it.

Largest N completable at the hardest R

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **“Hardest M” round ()** | **N** | **P** | **Q** | **# (%) Completed in 5 Minutes or Less** | **AVERAGE # NODES (Completed puzzles only)** | **AVERAGE TIME (Completed puzzles only)** | **STD.DEV. TIME (Completed puzzles only)** |
| 36 | **12** | **3** | **4** | 40% | 2938800 | 23.2 | 36.4118 |
| 56 | **15** | **3** | **5** | 40% | 968724 | 8.8 | 19.013 |
| 63 | **16** | **4** | **4** | 40% | 1678600 | 19.5 | 27.8011 |
| 80 | **18** | **3** | **6** | 0% | 0 | 0 | None |
| 99 | **20** | **4** | **5** | 0% | 0 | 0 | None |
| 109 | **21** | **3** | **7** | 0% | 0 | 0 | None |
| 142 | **24** | **4** | **6** | 0% | 0 | 0 | None |
| 180 | **27** | **3** | **9** | 0% | 0 | 0 | None |
| 194 | **28** | **4** | **7** | 0% | 0 | 0 | None |
| 222 | **30** | **5** | **6** | 0% | 0 | 0 | None |
| 253 | **32** | **4** | **8** | 0% | 0 | 0 | None |
| 303 | **35** | **5** | **7** | 0% | 0 | 0 | None |

Note: Timeout was set to 5 minutes.

Based on the table above, the “Largest ” is , as 12, 15,16 had a success rate at about 40%, which is far below 9’s. However, to avoid repeating (as the analysis has been done in step 3) for the next step, I will use as the largest N, as it has to best time performance of the 3 40%’s.

Is Hardest R constant as the sudoku puzzles scale?

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **N** | **P** | **Q** |  | **AVERAGE # NODES** | **AVERAGE TIME** | **STD.DEV. TIME** | **# (%) SOLVABLE** |
| **11** | **15** | **5** | **3** | **0.0494** | 122760 | 2 | 1.78885 | 100% |
| **22** | **15** | **5** | **3** | **0.0988** | 56709.6 | 1.1 | 1.44568 | 100% |
| **33** | **15** | **5** | **3** | **0.148** | 5495640 | 83.7 | 126.121 | 80% |
| **45** | **15** | **5** | **3** | **0.198** | 19058500 | 207.4 | 135.397 | 40% |
| **47** | **15** | **5** | **3** | **0.210** | 19334000 | 247.2 | 97.7362 | 30% |
| **50** | **15** | **5** | **3** | **0.222** | 22675300 | 272 | 90 | 10% |
| **53** | **15** | **5** | **3** | **0.235** | 24571000 | 301 | 0 | 0% |
| **56** | **15** | **5** | **3** | **0.247** | 26716280 | 301 | 0 | 0% |
| **58** | **15** | **5** | **3** | **0.259** |  |  |  |  |
| **61** | **15** | **5** | **3** | **0.272** |  |  |  |  |
| **67** | **15** | **5** | **3** | **0.296** |  |  |  |  |
| **78** | **15** | **5** | **3** | **0.346** |  |  |  |  |
| **89** | **15** | **5** | **3** | **0.395** |  |  |  |  |
| **100** | **15** | **5** | **3** | **0.444** |  |  |  |  |

**Note: Timeout was set to 5 minutes.**

**Find the critical value of the “hardest ” for**

**Based upon your results above, you estimate the “hardest ”** =

**How does puzzle solvability for your best combination vary with ?**

**Is the critical value for “hardest ” approximately the same as the value of for which a random puzzle is solvable with probability ?**

The hardest has around solvability probability of . The values for which the solvability probability is around would be around 0.259.

**Is hardest “hardest ”? Why or why not? How is the hardest related to the board parameter ? What is the shape of the function , which gives you the hardest ratio for ?**

The hardest is not approximately the hardest . However, based on the tests, 0.296 and 0.235 had the same solvability probability at . 0.296 had a slightly higher average time, which is why this value is considered to be the hardest for the largest , . As the puzzles scale, the number of cells, assignments, and backtracks increases. Thus, larger puzzles will take longer to complete, leading to higher average times and more chances of timeout. Based on these tests, the hardest R increases as increases although more tests at different s and hardest s are required for a more accurate relationship.

TODO:

* Clean up code, comment on code
  + Recompile in openlab
* Zip up and turn in new updated stuff