Sudoku Slayer

Alice Hanigan John Summers

Abstract—**The use of Constraint Satisfaction Problems(CSP) for modeling and solving Sudoku Puzzles is a natural implimentation of CSP techniques. Genetic Algorithms(GA) offer a much broader approach to the search space, but are not traditionally employed to solve Sudoku puzzles due to the puzzles structure naturally lending itself to other algorithms. The size of a Sudoku's search space is considerable and because of this, we decided to compare the use of CSP and GA's for solving Sudoku puzzles.**

I. Introduction

II. Literature Review

A. Solving Sudoku as a CSP

B. Solving Sudoku with GA

It is crucial, in designing a GA, to choose parameters which best fit the problem to be solved. Ahmad Hassanat [3] writes that there are four such parameters used by GA: crossover rate, mutation rate, population size, and the number of generations. Due to computational constraints, we were limited in our options for population size and number of generations, and as such we turned our focus primarily towards adjusting the mutation and crossover rates.

We also needed to consider what methods to use in designing our GA; specifically, we needed to decide on a method for each of evaluation, selection, crossover, and mutation. In order to better determine which methods to use, we turned towards previous experiments with and evaluations of GA. Nisha Saini [4] states that the most commonly-used selection methods are Roulette Wheel Selection, Rank Selection, Tournament Selection, and Boltzmann Selection. Of these, only Roulette and Tournament Selection were offered through the Python library we used, and so we considered the advantages and disadvantages of these methods. Roulette Wheel Selection suffers from premature convergence [4], and additionally, as we discovered ourselves, is poorly-suited to minimization problems, as it favors individuals with high fitness values. This left Tournament Selection, which preserves population diversity at the cost of degrading convergence speed [4].

Mutation is a necessary complement to crossover, as it allows the GA to escape from local optima that it otherwise would be constrained to [3]. Finding the ideal mutation rate, we were surprised to find, required considering more factors than we had expected: population size, crossover rate, and number of generations all are variables in determining the best mutation rate [3]. In fact, Hassanat proposes a dynamic approach, one which increases the mutation rate and decreases the crossover rate with each generation. Considering the complexity that Hassanat describes in this evaluation of mutation rates, we chose to focus on a set population size and crossover rate, in order to better tune our algorithm, and to create a stable environment in which to compare the respective approaches we used.

Another variable that we discovered, one which we had never considered could be changed, was the number of parents used in crossover. Traditionally, crossover is performed between two "parent" data operands, but Eiben [5] proposes a method by which crossover is performed between three or more parents. This method, although not a new theory, has seen little practical analysis as of yet [5], and as such, we believe this would be a fascinating subject to explore if we were to extend this project. However, to keep within our limited scope for this project, we chose to focus on binary operators for crossover.

III. Methodology

IV. Results

V. Discussion

VI. Conclusion

Although we found ourselves constrained by our circumstances in our choice of method and data, we have succeeded in creating both a deterministic algorithm that solves sudoku puzzles as a CSP, and a genetic algorithm that solves these puzzles through evolution. From our research and work, we would conclude that the former is a more effective approach to such a problem, but there is still much more that could be done to explore this comparison. In particular, incorporating multi-parent crossover methods, implementing dynamic mutation rates, testing our methods upon other sudoku variants (such as Killer or X Sudoku), and incorporating other emerging methods for GA would allow us to explore whether there might be more to be said on this comparison.

References

[1] Simonis, Helmut, 2006, "Sudoku as a Constraint Problem", http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.2964&rep=rep1&type=pdf

[2] Tayo, Takayuki, "Complexity and Completeness of Finding Another Solution and Its Application to Puzzles", Takahiro Seta, http://www-imai.is.s.u-tokyo.ac.jp/ yato/data2/SIGAL87-2.pdf

[3] Hassanat, Ahmad, "Choosing Mutations and Crossover Ratios for Genetic Algorithms-A Review with a New Dynamic Approach", Khalid Almohammadi, Esra'a Alkaween, Eman Abunawas, Awni Hammouri, V.B. Sura Prasath, https://www.mdpi.com/2078-2489/10/12/390/pdf

[4] Nisha Saini, "review of Selection Methods in Genetic Algorithms", https://www.ijecs.in/index.php/ijecs/article/download/2562/2368/

[5] A. E. Eiben, "Multi-Parent Recombination", https://www.cs.vu.nl/ gusz/papers/Handbook-Multiparent-Eiben.pdf