Development of a Sudoku Solver: C1 Research Computing Coursework

Vishal Jain

December 12, 2023

Contents

1 Introduction

"Sudoku is a denial of service attack on human intellect" - Ben Laurie

This report details the development of a Sudoku solver inline with the requirements of the C1 Research Computing coursework. The programme takes as input an incomplete grid in the form of a text file with a 9x9 grid of numbers with zero representing unknown values and '—','+','-' separating cells and , i.e.:

```
$ cat input.txt
000|007|000
000|009|504
000|050|169
---+---+---
080|000|305
075|000|290
406|000|080
---+---+---
762|080|000
103|900|000
000|600|000
```

and outputs the completed grid in the same form.

1.1 Background and motivation

Sudoku, a logic-based combinatorial number-placement puzzle, presents a paradigmatic example of a constraint satisfaction problem (CSP), a class of problems fundamental to the field of computer science. The puzzle's structure, consisting of a 9x9 grid divided into subgrids, adheres to stringent placement rules, thereby embodying the essence of CSPs where the objective is to find a solution that satisfies all given constraints.

This project, centered on the development of a Sudoku solver, is primarily motivated by the pedagogical value inherent in addressing such a well-defined and constrained problem space. Sudoku solvers exemplify the application of algorithmic strategies to a finite, yet non-trivial problem domain. This aligns with the core objectives of the C1 Research Computing coursework, which emphasizes the development of computational solutions that are both efficient and effective.

The logical structure and deterministic nature of Sudoku make it an ideal candidate for exploring various algorithmic approaches, from brute-force methods to more sophisticated heuristic algorithms. Furthermore, the project offers a platform for implementing and refining software development practices, particularly in areas such as code clarity, modular design, and performance optimisation.

In essence, the construction of a Sudoku solver serves not only as an exercise in algorithm implementation but also as a microcosm for broader software development and computational problem-solving skills. It provides a controlled environment to experiment with and evaluate different computational strategies, thereby contributing to a deeper understanding of both the specific problem of Sudoku and the general principles of algorithm design and optimization in computer science.

2 Selection of Solution Algorithm

The choice of an algorithm for solving Sudoku is predicated upon a set of criteria that balance efficiency, complexity, and the ability to effectively navigate the vast solution space inherent to the puzzle. The decision-making process encompassed an evaluation of several algorithmic paradigms, ultimately converging on the selection of the backtracking algorithm for its suitability to the problem at hand.

Backtracking, a depth-first search approach, was chosen primarily for its inherent simplicity and robustness in tackling constraint satisfaction problems such as Sudoku. This algorithm systematically explores the puzzle's grid, placing numbers in a trial-and-error manner and retracting (backtracking) erroneous placements, thereby navigating the solution space in a methodical and exhaustive manner. Its recursive nature allows for an elegant solution that can be implemented in a few lines of code, making it an ideal choice for the first iteration of the solver.

Alternative approaches, including heuristic-driven algorithms like stochastic search and constraint propagation, were also contemplated. While these methods offer potentially faster solutions by reducing the search space more strategically, they introduce a layer of complexity that may not align with the educational objectives of the coursework. The deterministic and transparent process of backtracking provides a more straightforward demonstration of the principles underpinning CSPs and aligns more closely with the coursework's emphasis on clarity and fundamental algorithmic understanding.

In summary, the selection of backtracking as the solution algorithm is underpinned by its educational value, ease of implementation, and its alignment with the pedagogical goals of developing a deeper understanding of CSPs in a computationally efficient and conceptually clear manner.

3 Prototyping

Conceptually the problem can be broken down into three distinct parts: - Parsing the input file into a data structure - Defining the sudoku board data structure - Defining a solver

The idea was

4 Choice of solving algorithm

5 Development, Experimentation and Profiling

Describe the development process, including the experimentation with different algorithms or techniques and any profiling done for performance optimization.

6 Validation, Unit Tests and CI Setup

Explain how you validated the solution. Discuss the unit tests written for the project and any continuous integration setup.

7 Packaging and Usability

Detail how the project is packaged for use. Discuss any steps taken to ensure the usability of the software, including documentation.

8 Summary

Conclude with a summary of the project, key learnings, and possible future improvements.