CW1 Report

F29AI - Artificial Intelligence

Keqin ZHANG (H00460395)   Yuwei ZHAO (H00460398)

OUC CW1 33   2025-11-10

Abstract

This report involves the solutions to the tasks outlined in Coursework 1 for the `F29AI - Artificial Intelligence` course. The main objective of the coursework is to search algorithms and automated planning using *PDDL*.

See Resources on git@github.com:ZhangKeqin0307/coursework1.git.

1 Introduction

2 Procedure

2.1 Part 1 - Solving and Analyzing Sudoku with Search Algorithms

2.1.1 Part 1A

< Brief Description >

Assume that a *Sudoku* consists of a 9×9 grid. The objective is to fill the grid with digits in such a way that each row, each column, and each of the 9 principal 3×3 sub-squares contains all of the digits from 1 to 9. An approach to an intelligent sudoku solver is needed.

- Define Sudoku formally as a constraint satisfaction problem.
- What are the variables, domains, and constraints?
- Discuss the time complexity of brute-force search vs. backtracking in Sudoku.

< Solution >

A CSP(constraint satisfaction problem) should involve the following three components: Variables, Domains and Constraints. Therefore, we can define the Sudoku problem as follows:

$$\text{Sudoku} = \langle V, D, C \rangle$$

where

1. V: The set of 81 variables, $V = \{V_{i,j} \mid i, j \in \{1, 2, \ldots, 9\}\}$.
2. D: The domain $D_{i,j}$ for each variable $V_{i,j}$ is defined as:
   - $D_{i,j} = \{k\}$, if $V_{i,j}$ is a given cell with value k.
   - $D_{i,j} = \{1, 2, \ldots, 9\}$, if $V_{i,j}$ is an empty cell.
3. C: The set of 27 ”All-Different” constraints:
   - C_{row}: For each row i, all variables $V_{i,1}, V_{i,2}, \ldots, V_{i,9}$ must have different values.
   - C_{col}: For each column j, all variables $V_{1,j}, V_{2,j}, \ldots, V_{9,j}$ must have different values.
   - C_{subgrid}: For each 3×3 subgrid, all 9 variables within that subgrid must have different values.

Time Complexity Analysis:

- Brute-force Search Algorithm:
  For each of the k spaces, there are 9 possible choices of numbers. This results in a total of $9 \times 9 \times \ldots \times 9$ (k times) combinations. Therefore, the time complexity of the brute-force search algorithm is $O(9^k)$. When the worst-case scenario occurs, the algorithm needs to explore all possible combinations, leading to the $O(9^{81})$ time complexity.
- Backtracking Search Algorithm:
  It checks the validity of constraints (row, column, and 3×3 sub-grid) **immediately** after assigning a number to a cell. If a conflict is detected (i.e., the current partial solution is invalid), the algorithm recursively ”backtracks” to the previous step to try a different number. This process effectively **prunes** large sub-trees of the search space that are known to be invalid.

  While the theoretical worst-case time complexity remains $O(9^k)$ (similar to brute-force), the average-case performance is drastically faster. This is because the *effective branching factor* b becomes significantly smaller than 9 ($b \ll 9$) as the constraints restrict the number of valid choices for each subsequent cell.

2.1.2 Part 1B

< Brief Description >

Build an intelligent *Sudoku* Solver.

< Solution >

2.2 Part 2 - Automated Planning

2.2.1 Part 2A: Modelling the Domain

2.2.2 Part 2B: Modelling the Problems

2.2.3 Part 2C: Extension

3 Reflection and Analysis

4 Conclusion