Sudoku Graph Coloring Solver

Orhan Berkay Yılmaz - 150210054

June 2, 2025

1 Overview

This project implements a Sudoku solver using graph coloring algorithms. The implementation treats Sudoku as a graph coloring problem where each cell is a vertex and edges represent constraints (same row, column, or block).

There are 2 executables 'main' and 'main$_{\mathrm{simpl}}$'. 'main$_{\mathrm{simpl}}$' only provides dependency creation, greedy coloring and 4x4 board. 'main' is way more configurable and supports the following solvers and both 4x4 and 9x9 boards.

1.1 Graph Representation

- Each Sudoku cell is represented as a vertex in the graph
- Edges connect vertices that cannot have the same value
- Three types of constraints create edges:
  1. Row constraints: cells in the same row
  2. Column constraints: cells in the same column
  3. Block constraints: cells in the same 2×2 or 3×3 block
- The graph is represented using an adjacency matrix for efficient constraint checking

1.2 Solver Types

1. Greedy Solver
   - Simple vertex coloring approach
   - O(n^2) time complexity
   - May not find optimal solutions

- Colors vertices in order, using first available color
- Fast but may get stuck in local minima

2. DSatur Solver

    - `https://en.wikipedia.org/wiki/DSatur`
    - Uses saturation degree heuristic
    - More efficient than greedy
    - Better solution quality
    - Saturation degree = number of different colors used by neighbors
    - Always picks vertex with highest saturation degree
    - If tied, picks vertex with highest degree

3. Backtracking Solver

    - Complete search algorithm
    - Uses MRV (Minimum Remaining Values) heuristic
    - Guarantees solution if one exists
    - Tries all possible colors for each vertex
    - Backtracks when no valid color is found
    - Uses MRV to pick most constrained vertex first
    - Maintains best attempt for partial solutions

4. Heuristic Kempe Solver

    - Based on: `https://www.cs.princeton.edu/~appel/Color.pdf` p.9
    - Uses degree-based vertex selection
    - Handles complex constraint propagation
    - Prioritizes vertices with degree < K
    - Uses color constraints from neighbors
    - Maintains best attempt for partial solutions
    - Can handle complex constraint chains

1.3 Step Counting

- Each solver tracks number of coloring attempts
- Greedy/DSatur: Count vertex coloring attempts
  - One step per vertex coloring attempt
  - Includes failed attempts
- Backtracking/Kempe: Count color assignment attempts
  - One step per color assignment
  - Includes backtracking steps
  - Higher step count indicates more complex solving process

2 Usage

```
make all
./main <solver_type> [board_size]
```

Where:

- solver_type: greedy, dsatur, backtrack, kempe
- board_name: 4x4 (default) or 9x9 or 9x9_extreme

2.1 Example Output

```
Using Heuristic Kempe solver on 4x4 board:
 4  1 -1  3
-1  2 -1  4
 2 -1  4  1
 1  4 -1 -1
=========================
Solving Sudoku...
=========================
 4  1  2  3
 3  2  1  4
 2  3  4  1
 1  4  3  2
Steps taken: 6
```