Sudoku problem solver

How effective is parallelization?

Creators

Abdullah AL Hinaey - Brute Force Implementation

JohnMichael Kane - Dancing Links Implementation

Malon Masia - Report / Presentation

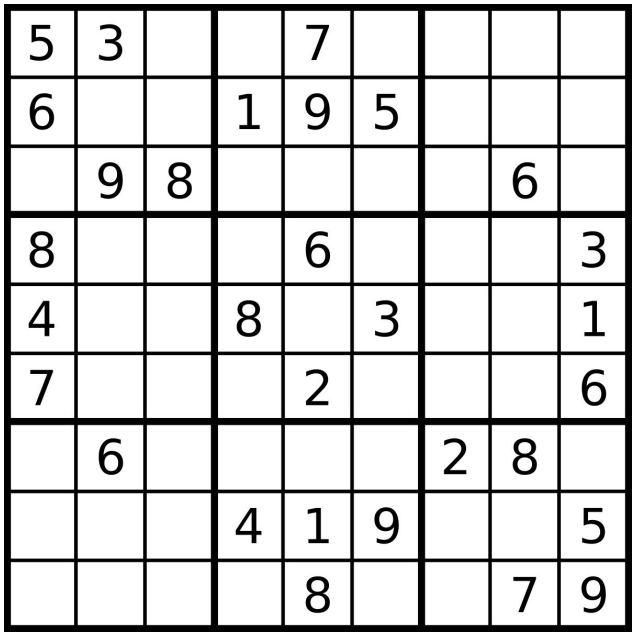
Soleil Cordray - Propagation and Search Implementation

Abstract

In computing, most problems that can be solved sequentially also have a parallel solution. Not all the time but in some cases parallelization can make a program more efficient than its sequential counterpart, especially when it comes to large data. When it comes to algorithms that can solve a Sudoku board, some of these include brute force, dancing links, and propagation and search. These algorithms can be implemented using sequential techniques or parallelization techniques. Our goal with this paper is to prove that parallelization techniques are more efficient at finding solutions than sequential techniques.

What is Sudoku?

Modern day Sudoku is a popular combinatorial puzzle game that gained its prominence in Japan, 1986. The game came to the United States a few decades later in 2004, and has since become one of the staple puzzle pastimes. Sudoku boards, depending on their difficulty and how familiar the player is with the game, can take anywhere from 5 to 25 minutes to solve.



Problem and solution

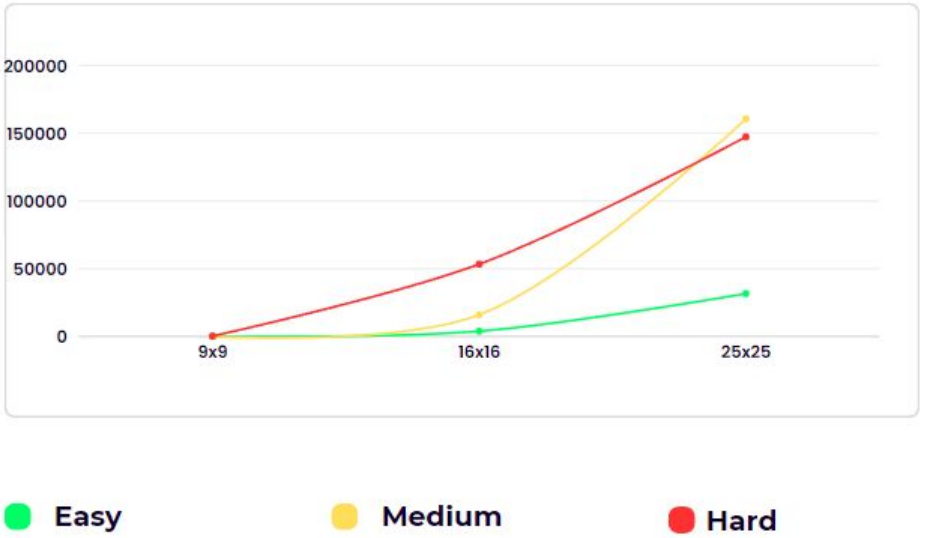
Computers can obviously solve these kinds of puzzles a lot faster, but how do different algorithms compare to one another? and what is the effectiveness of parallelization techniques to help improve those times?

For this project we will be looking at three different algorithms: bruteforce, propagation and search, and dancing Links. Beyond this it is we will also be looking at board difficulty to get a better sense as to how these algorithms both sequentially and parallelized will handle more complicated boards.

Key Challenges

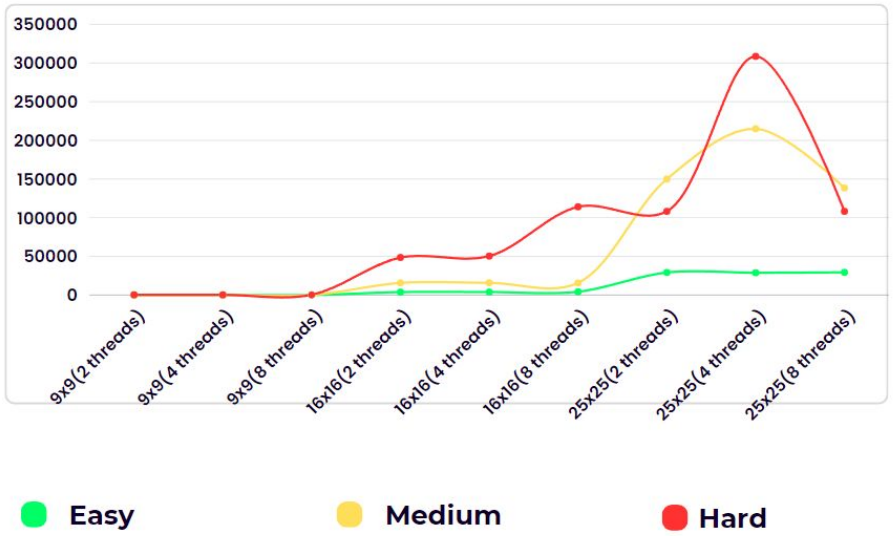
Sequential Brute Force

- How It Works
- The Execution
- The performance



Parallelized Brute Force

- Thread Management Techniques
- Challenges and Performance Insights
- Primary Reason for Slowdown

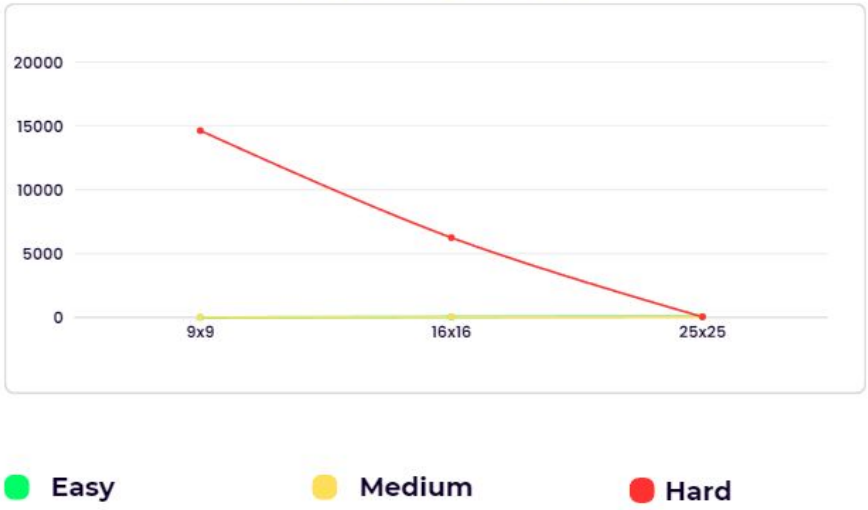


Propagate-Cross-Search

- Comprehensive solution to solving Sudoku boards!
- Technique:
  1. Check linear constraints (stand-alone values).
  2. Cross-reference rows, columns, and sections (intersecting values) if any remaining empty cells.
  3. Depth first search if any remaining empty cells.
- Compile:
  - g++ -std=c++17 -g -o SudokuSolver SudokuSolver.cpp Game.cpp Grid.cpp PossibleGrid.cpp
  - ./SudokuSolver

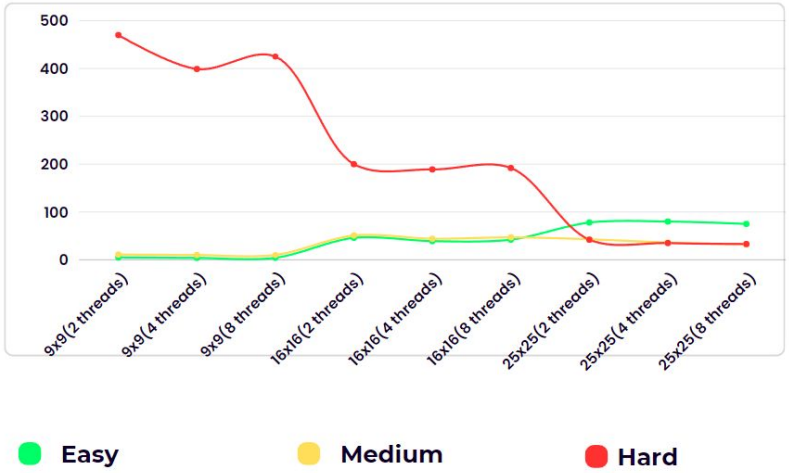
Sequential Propagate-Cross-Search

- Overall: Runtimes merge as difficulty and size increases
- Special case:
  - Runtime shortened on harder board sizes. Sudoku boards are harder when empty cells are close together, and my algorithm does really well at checking those cells due to the mass row/column checks.



Parallelized Propagate-Cross-Search

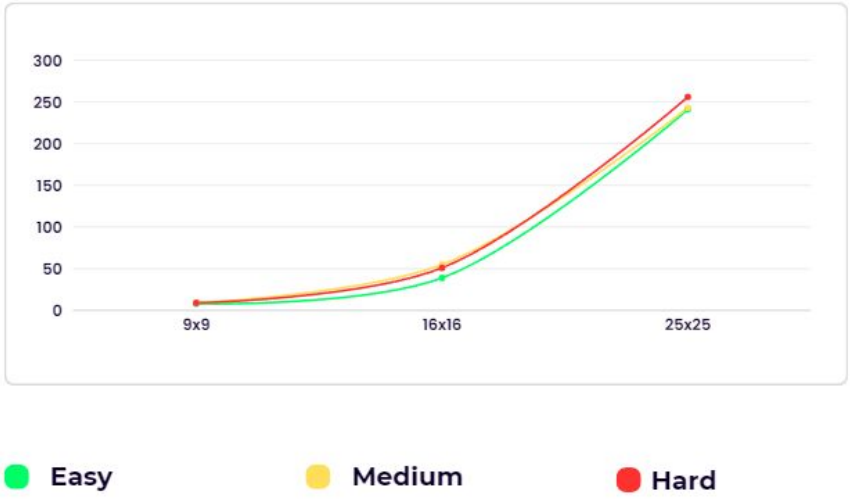
- Overall
  - Runtimes fluctuate, but follow a similar pattern to the sequential runtimes.
- Key idea:
  - More threads does not necessarily mean more speed. There must be a balance between the work done.



Dancing Links Sudoku

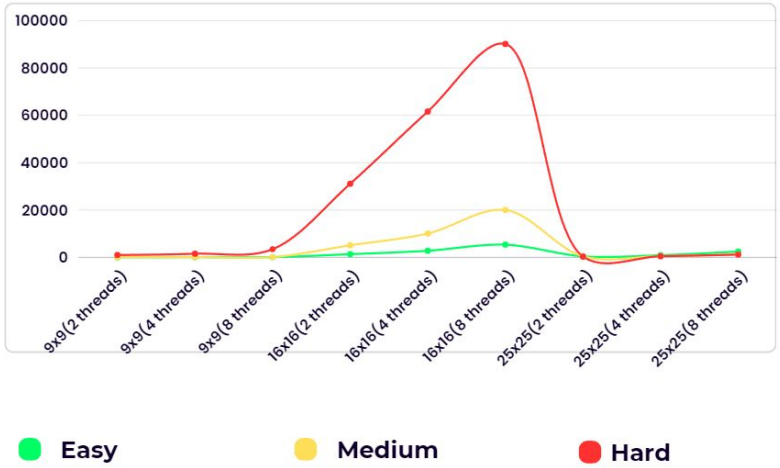
Sequential Dancing Links

- Runtime increases proportionally with the board size.
- Slight, albeit negligible differences between the difficulties. Relatively the same runtime across difficulties.
- Very efficient across different algorithms



Parallelized Dancing Links

- Parallelization works best for the larger problems and the smaller ones, but struggles with the harder problems and the intermediate length (16x16).
- More threads leads to a slower runtime.
- Hard 16x16 implementation is MUCH greater than everything else in terms of runtime.



Research Takeaways

- Parallelization may not always be the most effective method
  - More complex and difficult to implement than sequential method
  - Might not always yield fast run times
  - Sometimes leads to worse runtimes
- Better understanding of how parallelization works
  - Learning through mistakes
  - Have deeper understanding as to when parallelization might be most beneficial
- There is always more to look into for this project