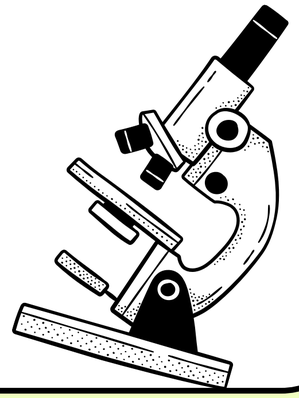
Futoshiki Puzzle Solver

Naruebet Songsri - 6580006
Kritsadapon Wai-On - 6481314

Success Criteria

Achieve a significant speedup of 5x or more in solving Futoshiki Puzzle using the Parallel Solver compared to the Sequential Solver.



Component

- gui.py - GUI for board input
- Main - Read board data json file, Initialise board and start solver
- FutoshikiPuzzle - Puzzle object
- PerformanceTest - Run time timer
- Solver, ParSolver - Solver object with backtracking

How it work

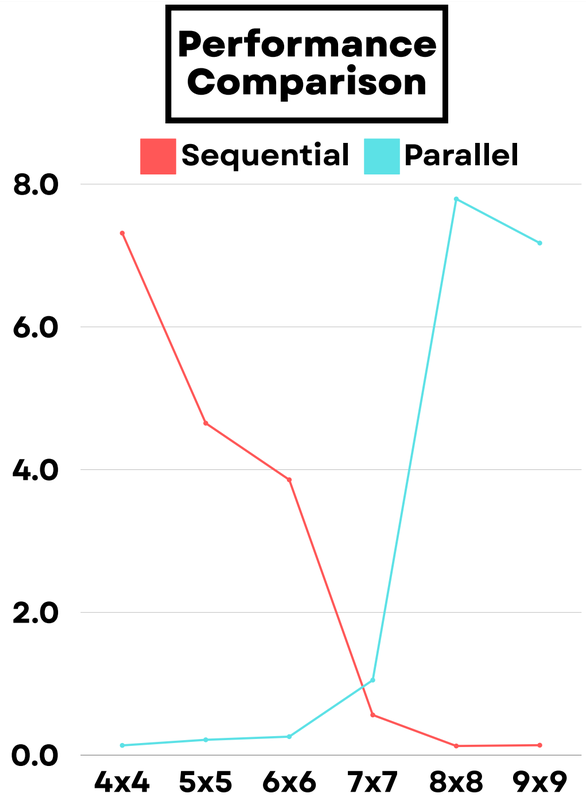
1. Check if the board is solved.
2. Find the First Empty Cell
3. For each valid number for the first empty cell, spawns a Future to attempt to solve
4. Each Future uses a recursive backtracking
5. Waits for any of the futures to complete the promise with a solution

Backtracking Logic: For each cell, if it's not empty, it moves to the next cell. If it's empty, it tries each possible number, sets the number in the cell, and recursively attempts to solve the puzzle from that point. If a solution path fails, it resets the cell (backtracks) and tries the next number. If it successfully fills all cells, it returns true.

Github Repository

Data Analysis Sheet

Performance Comparison



Run Time Comparison

Size	Mean Runtime - Parallel
4x4	32.71
5x5	52.00
6x6	57.39
7x7	572.75
8x8	339.05
9x9	33674.11

Size	Mean Runtime - Sequential
4x4	4.47
5x5	11.18
6x6	14.87
7x7	602.20
8x8	2642.01
9x9	241606.36

Results

For smaller board size the run time of the Sequential Solver was faster with a 4x4 board being 7x faster than the Parallel Solver. However, the difference in run time become less noticeable as the size of the board increases. After 7x7 the Parallel Solver take the lead and we see a speed up of up to 7.8x for the 8x8 board. Our 9x9 board result is slightly skewed as we used board with multiple possible solution to allow the Sequential Solver to run faster as it was taking multiple hours to solve the board.
The run time of the Parallel Solver also see a massive speed up over the Sequential Solver when there are more constraints added.

Conclusion

The Parallel Solver achieved our success criteria and surpassed our expectation in the speed up of the run time. However, it seem that with smaller workload the Sequential Solver tends to be faster, when more computation power is require, the Parallel Solver become faster which is due to the Initialisation and Communication overhead slow down the Parallelisation on smaller workload.