# Sudoku Solver Summary

This project improves classical Sudoku-solving methods by integrating Breadth-First Search (BFS), Depth-Limited Search (DLS), and Constraint Satisfaction Problem (CSP) techniques into a multi-threaded framework. These enhancements enable the solver to handle both standard (9x9) and larger (16x16) grids efficiently, offering scalability and adaptability to different puzzle complexities.

The solver uses a graph-based representation where each cell is a vertex, and edges connect constrained cells within the same row, column, or box. BFS explores the puzzle level by level, ensuring all potential states are examined. DLS improves on this by introducing a depth limit, cutting off less promising branches to reduce redundant computations. CSP takes the optimization further by employing intelligent heuristics like Minimum Remaining Values (MRV) and Least Constraining Values (LCV). These heuristics prioritize the most constrained cells and the least restrictive values, significantly reducing the search space. The solver runs all three algorithms in parallel using Java's Executor Service, leveraging multi-core processors to minimize runtime.

On execution, CSP consistently outperformed BFS and DLS in speed and efficiency. For a standard 9x9 puzzle, CSP solved it in 12.15 seconds, compared to 35.66 seconds for DLS and 65.91 seconds for BFS. For a larger 16x16 grid, CSP completed the puzzle in just 0.08 seconds, while DLS and BFS required 73.00 and 269.53 seconds, respectively. The efficiency of CSP stems from its strategic pruning of the search space, which avoids expanding unnecessary nodes. While BFS and DLS expand the same number of nodes, DLS manages its search depth more effectively, leading to faster convergence. However, both BFS and DLS face scalability challenges on larger grids due to higher memory usage and runtime, making CSP the preferred choice for complex puzzles.

The solver's flexibility to handle grids of varying sizes is a significant improvement over traditional methods. By combining graph-based modeling with multithreading, the runtime is drastically reduced, allowing simultaneous exploration of BFS, DLS, and CSP strategies. CSP, in particular, demonstrates clear advantages in both speed and memory efficiency, especially for larger puzzles. The multi-threaded approach ensures adaptability and optimal performance across different puzzle complexities.

These enhancements make the Sudoku solver highly efficient, showcasing the potential of combining classic algorithms with modern optimization techniques. By integrating heuristics like MRV and LCV, the solver not only improves upon traditional methods but also opens avenues for applying these strategies to other constraint-based problems. This project illustrates the power of advanced algorithms and parallel execution in tackling complex computational challenges.