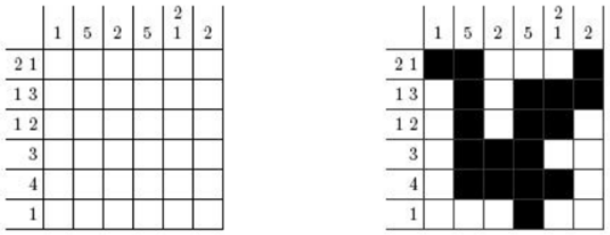
Nonollel: Nonograms Parallel Solver

Jason Eriksen (jce2148) & Xurxo Riesco (xr2154)
COMS 4995 Parallel Functional Programming
Final Project Proposal

Nonograms

Nonograms are logic puzzles consisting of an `m x n` grid and a set of `m + n` constraints, each consisting of a sequence of positive numbers denoting the number of consecutive squares that must be colored in either a row or a column. The objective of the puzzle is to color the whole grid while satisfying all constraints. Below is an example portraying an unsolved and solved version of the puzzle.



(a) 6 × 6 Nonogram (b) Solved Nonogram

Solvability

Solving nonograms is an NP-complete problem. In theory, it is possible for a puzzle to have multiple solutions, in which case all of them are considered to be correct. To begin solving a nonagram, basic mathematical rules can be used to fill out some portion of the grid without any multistep recursion needing to be used. For instance, in column 2 in the above example, the middle 4 squares of the column can be colored in because we know that the only way for 5 to fit in a column of height 6 can be with those squares filled in. However, after that, the implementation of a solver requires deeper recursion, making it very suitable for a computer to generate a game tree and then perform search on the given tree while backtracking upon contradictions between the possibilities for one row/column and the constraints on the crossing rows/columns.

Parallel Improvements

Due to the NP-complete and gridded nature of this problem, there is potential for significant speedups with parallelization. In our initial step, we can distribute the mathematical elimination step onto separate threads. We can then proceed with a multi-threaded generation of the game tree followed by depth first search in which each thread attempts to solve the puzzle based on a certain assumption, resetting and searching again when a contradiction is reached. The exponential growth of the game tree is expected to add significant overhead so we expect the speed-up to only be noticeable given a large enough problem where said overhead is outweighed.