J.T. Timpanaro - jtt2120

Parallel Functional Programming

November 22, 2019

Project Proposal – 0-1 Knapsack Problem

My proposal for the final project is a parallelized version of a 0-1 Knapsack Problem solver. The Knapsack Problem is NP-Complete combinatorial problem. One is given a set of items, each item having a weight and value, and a maximum capacity. The goal is to maximize the value of items taken while keeping the total weight below the capacity. The 0-1 Knapsack Problem is the most common variation, in which one can only take 0 or 1 of each item.

There exist two possibilities for each item: take or don't take. This means there are 2^n potential solutions for the problem. The naïve brute-force solution to this problem is to compute the total value and weight for each of these solutions, filter out the invalid solutions, and return the solution with the greatest value.

Being $O(2^n)$, this algorithm scales very poorly, so it stands to reason it would benefit from parallelization. Since each solution is computed individually, this work can be split up amongst available processors, greatly improving the speed of the program. A parallel version of this program might work as follows: generate the potential solutions, compute the total value and weight for each of these in parallel, and then filter out the invalid solutions and return the greatest one.

There also exists a dynamic programming solution to the knapsack problem. This problem involves a matrix M, and we define $M[i,w]$ to be the maximum value we can obtain

with weight less than or equal to *w*, using the first *i* items in the list. This solution runs in pseudo-polynomial time O(nW), where W is the capacity of the knapsack and n is the number of items. I would also like to explore improving this solution in the project, possibly by parallelizing the computation of each matrix row.