COMS4995 Parallel Functional Programming

Parallel 0/1 Knapsack Solver Using Branch-and-Bound

Jessica Xu
UNI: lx2297

1 Overview

The goal of this project is to implement a solver for the classical 0/1 Knapsack Problem using Haskell, first in a sequential version and then in a parallel version using `Control.Parallel` and `Control.Parallel.Strategies`. The 0/1 knapsack is NP-hard, and branch-and-bound methods naturally expose parallel structure. Our objective is to evaluate how parallel exploration of the branch-and-bound search tree improves performance compared to a sequential baseline.

2 Background

Definitions

We are given a set of n items, where each item i has a positive weight w_i and positive value v_i, and a knapsack with capacity W. The 0/1 Knapsack Problem asks us to select a subset of items such that:

- the total weight does not exceed W, and
- the total value is maximized.

This is an NP-hard combinatorial optimization problem (Karp, 1972). A typical exact solver uses a branch-and-bound recursion over decisions to include or exclude each item, computing an upper bound at each node and pruning subtrees whose bound cannot exceed the best known solution.

Problem

A naïve solver enumerates all 2^n subsets. Branch-and-bound significantly reduces the search space by computing upper bounds for each partial assignment. A standard choice is to sort items by value-to-weight ratio and use a fractional knapsack relaxation to compute a fast bound. Because including vs. excluding an item leads to independent subtrees, the method parallelizes well.

3 Algorithm

Our sequential solver will follow the standard branch-and-bound design:

- Sort items by value-to-weight ratio.
- The recursive search function keeps track of: current index, current weight, current value, and remaining bound.
- Compute an upper bound on the best possible value reachable from this node. If this bound is no better than the global best, prune.
- Otherwise branch: include the next item (if capacity allows) or exclude it.
- Update the global best when reaching a leaf node.

The parallel version will extend the recursion using Haskell's `par` combinator or `parList rdeepseq`. In particular:

- Expand the first few levels of the recursion to generate a frontier of independent subproblems.
- Evaluate these subtrees in parallel.
- Use a shared mutable reference (e.g. `IORef` or `MVar`) for the global best value, updated with compare-and-swap semantics.

This is similar in spirit to the parallel branch-and-bound work found in general combinatorial search literature and prior Haskell work.

4 Objectives and Workflow

Experiment preparation

We will write a generator for knapsack instances, including:

- uniformly random weights and values,
- strongly correlated instances ($v_i \approx w_i$),
- weakly correlated or uncorrelated instances,
- adversarial instances where many items have identical ratios (which reduces pruning effectiveness).

For small instances, we will verify correctness using a standard dynamic programming solver implemented in Python.

Experiment design

We will measure the performance of:

- the sequential branch-and-bound solver, and
- the parallel solver with various numbers of threads (`-N1`, `-N2`, `-N4`, `-N8`).

We will compare:

- elapsed runtime,
- degree of pruning,
- number of nodes explored,
- speedup and parallel efficiency,
- effects of different initial frontier depths.

We will use GHC's runtime profiling tools (`+RTS -s`, spark profiles, event logs) to examine parallel behavior.

5 References

- GeeksforGeeks, "0/1 Knapsack using Branch and Bound". https://www.geeksforgeeks.org/0-1-knapsack-using-branch-and-bound/
- GeeksforGeeks, "0/1 Knapsack using Least Cost Branch and Bound". https://www.geeksforgeeks.org/dsa/0-1-knapsack-using-least-count-branch-and-bound/
- J.C. Zúñiga-Díaz, M.A. Camacho-Cárdenas, J. Lattimore-Cruz, "A Multi-Branch-and-Bound Parallel Algorithm to Solve the Knapsack Problem 0–1 on a Multicore Cluster", *Applied Sciences*, 2019. https://www.mdpi.com/2076-3417/9/24/5368
- S. Sahni, "Anomalies in Parallel Branch-and-Bound Algorithms: the 0/1 Knapsack Problem." https://www.cise.ufl.edu/~sahni/papers/anomaly.pdf
- S. Hildebrandt, C. Hanson, "0-1 Knapsack Optimization with Branch-and-Bound", MICS Symposium 2016. https://www.micsymposium.org/mics2016/Papers/MICS_2016_paper_42.pdf
- P. Trinder, K. Hammond, H. Loidl, G. Jones, "Algorithm + Strategy = Parallelism", a well-known introduction to parallel functional programming.
- B. Archibald, P. Maier, C. McCreesh, R. Stewart, P. Trinder, "Replicable Parallel Branch and Bound Search", 2017. https://arxiv.org/abs/1703.05647
- Wikipedia, "Knapsack Problem". https://en.wikipedia.org/wiki/Knapsack_problem