The link to the capstone project description
https://www.cs.rit.edu/~ark/mastersprojects/seesawsearch.shtml

Prof. Alan Kaminsky
Rochester Institute of Technology -- Department of Computer Science

---

Background

A combinatorial optimization problem consists of a set of *configurations,* one or more *constraints,* and one or more *optimizations.* Example: the knapsack problem.

- Configuration = a subset of a set of items, each item having a weight and a value
- Constraint = sum of item weights in the configuration ≤ capacity
- Optimization = sum of item values in the configuration is maximized

A *stochastic local search* algorithm goes like this: Start with an arbitrary configuration; repeatedly make a randomly chosen small change to the configuration; after examining a large number of configurations, report the best one. Example: the knapsack problem.

- Initial configuration = an arbitrary subset of the set of items
- Configuration change = remove a randomly chosen item in the configuration or add a randomly chosen item not in the configuration
- Best configuration = has the largest total item value with total item weight not exceeding capacity

A *massively parallel stochastic local search* algorithm consists of performing numerous stochastic local searches in parallel, then reporting the best configuration found over all the individual searches. Doing the searches in parallel lets the algorithm explore more of the configuration space in the same running time, potentially improving the solution as compared to just a single search.

A *seesaw search* algorithm is a particular kind of stochastic local search. The seesaw search consists of two phases. In the *optimizing phase,* the configurations are changed so as to increase the degree of optimization, without regard for whether the constraints are met. The optimizing phase continues until the configuration cannot be optimized further. In the *constraining phase,* the configurations are changed so as to increase the degree to which the constraints are satisfied, without regard for whether the optimization criteria are met. The constraining phase continues until all the constraints are met. The seesaw search alternates between the optimizing phase and

the constraining phase. After examining a large number of configurations, the best one is reported. Example: the knapsack problem.

- Optimizing phase = add a randomly chosen item not in the configuration; ends when total item weight > capacity
- Constraining phase = remove a randomly chosen item in the configuration; ends when total item weight < capacity

A *massively parallel seesaw search* algorithm consists of performing numerous seesaw searches in parallel, then reporting the best configuration found over all the individual searches.

See my textbook *Big CPU, Big Data, Second Edition*, Chapter 34 for a complete massively parallel seesaw search program for the knapsack problem that runs on a GPU accelerated node. The program is written in Java using the Parallel Java 2 Library; the GPU kernel is written in C using Nvidia's CUDA.

Prerequisites

To do this project, you must have completed the following course with a grade of B or better:

- CSCI 654 Foundations of Parallel Computing

Description

More than one student may do this project. Each student must choose a different combinatorial optimization problem to study.

- Identify a combinatorial optimization problem amenable to being solved by a massively parallel seesaw search, such as minimum vertex cover, minimum graph coloring, maximum independent set, etc.
- Design an exact search algorithm for the chosen problem; that is, an algorithm that finds the true optimum solution. (The exact search is expected to be practical only for small problem instances.)
- Implement a parallel exact search program for the chosen problem.
- Design a seesaw search algorithm for the chosen problem.
- Implement a massively parallel seesaw search program for the chosen problem.
- Investigate the quality of the seesaw search program's solutions; that is, how close the program's solutions are to the true optimum solutions.
- Research the literature and compare the seesaw search program's results (running time, solution quality) to other published programs.

Note: The programs must be written in Java using the Parallel Java 2 Library. The programs may run on a multicore node, a cluster, or a GPU accelerated node.

Note: Seesaw search algorithms for the following problems have already been studied. You must pick a problem other than these:

- Bin packing
- Knapsack
- Maximum satisfiability (MAX-SAT)
- Minimum vertex cover

Timeline and Milestones

You must meet with Prof. Kaminsky each week to discuss your progress. Specific graded milestones and their due dates are listed below.

- By week 3: Literature review. Introduction (including a description of the chosen combinatorial optimization problem) and Related Work sections of the final report completed.
- By week 5: Parallel exact search program complete; true optimum solutions found for several small problem instances.
- By week 7: Massively parallel seesaw search program complete; solutions compared to true optimums for several small problem instances.
- By week 11: Investigation of massively parallel seesaw search program's results and running times scaling up to large problem instances complete.
- By week 13: First draft of complete report and poster.
- By week 14: Final report and poster.

Note: This project is available ***ONLY*** to students who can meet with Prof. Kaminsky ***IN PERSON*** every week.