Parallel Branch and Bound: Applying an Asynchronous Multi-Pool Approach to Cyclic Best First Search

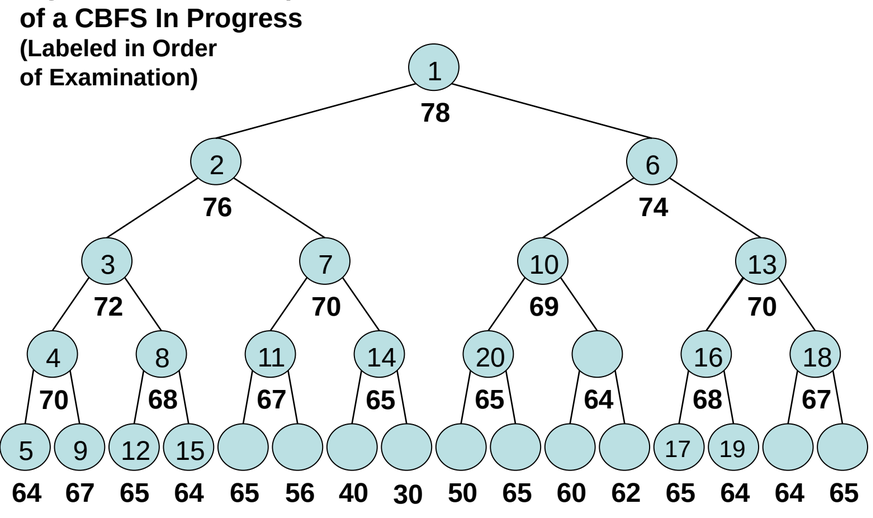
Joshua Gluck, Nartezya Dykes

Swarthmore College, Spelman College, and The University of Illinois at Urbana-Champaign

I. Introduction

Branch and Bound (B&B) is a general algorithm for finding optimal solutions to a range of combinatorial optimization problems, with applications from managing employee shifts at hospitals to optimizing profits for shipping corporations. B&B generates results trees which must be searched. Cyclic Best First Search (CBFS) is a new search algorithm for optimal processing of B&B trees [1] [2]. We investigate the feasibility of implementing CBFS in parallel, quantify the resulting speedup and identify issues and future directions.

Figure 1: First 20 Steps of a CBFS In Progress (Labeled in Order of Examination)

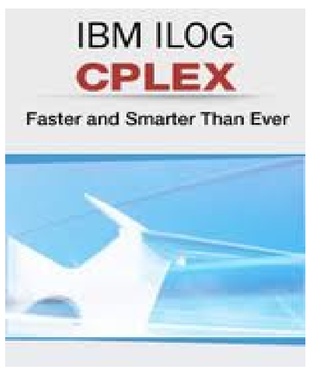


II. Motivation

Our purpose was to examine the feasibility and potential benefit of CBFS implemented in parallel. We hoped to achieve reduced runtimes while minimizing parallel overhead. Our investigative starting point was the application of parallel methods from other predecessor algorithms [5] to CBFS. From the possible methods, we determined that an asynchronous multi-pool approach showed the greatest potential increase in throughput.

III. Materials

Our parallel Implementation of CBFS was programmed using the POSIX Threads library in C++. Tests were run on an Intel Core i7 2.8 GHz Processor with four hyperthreaded cores (supporting 8 parallel threads) and 12 GB of RAM, running CentOS 6.2. The Knapsack B&B test problem was chosen for its extremely common use [3][4] as well as its relative ease of implementation. The individual Knapsack sub problems were solved using IBM ILOG CPLEX Optimization Studio.



IV. Results

Our 8-way parallel implementation was tested on Knapsack problem instances with 95 to 120 variables. **The implementation generated a significant reduction in wall time (Figure 2) amounting to at least a factor of 10 speed up for all problems.** Given an 8-thread implementation, our achieving greater than an 8x speed gain may seem surprising; it is accomplished via the reduction in insertion times into priority queues enabled by the multi-pool approach.

Figure 2: Serial vs. Parallel Wall Time

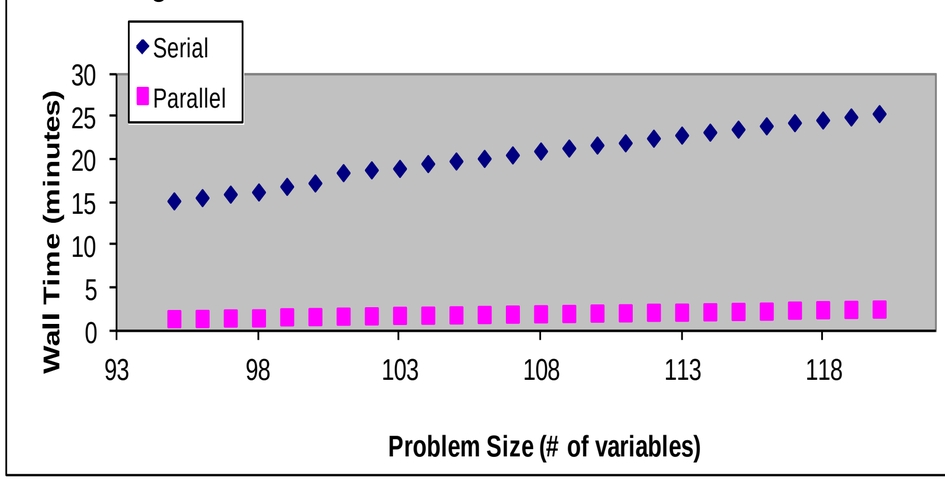
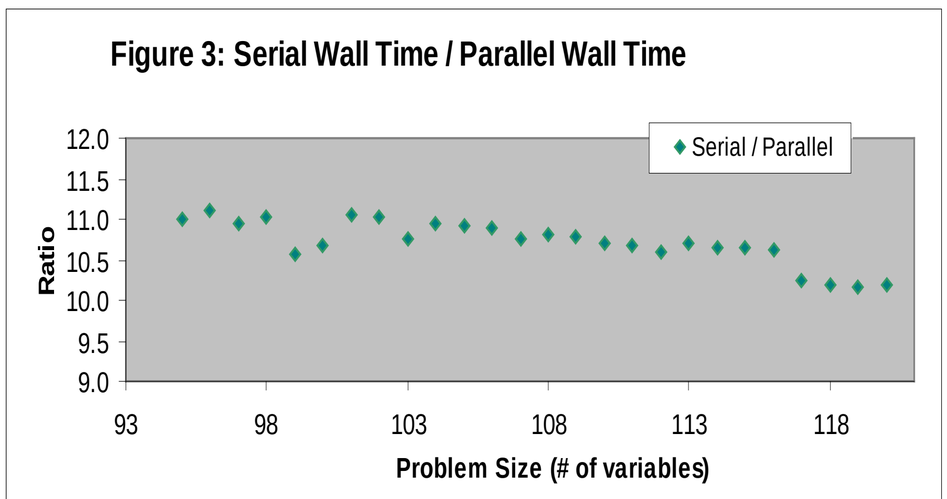


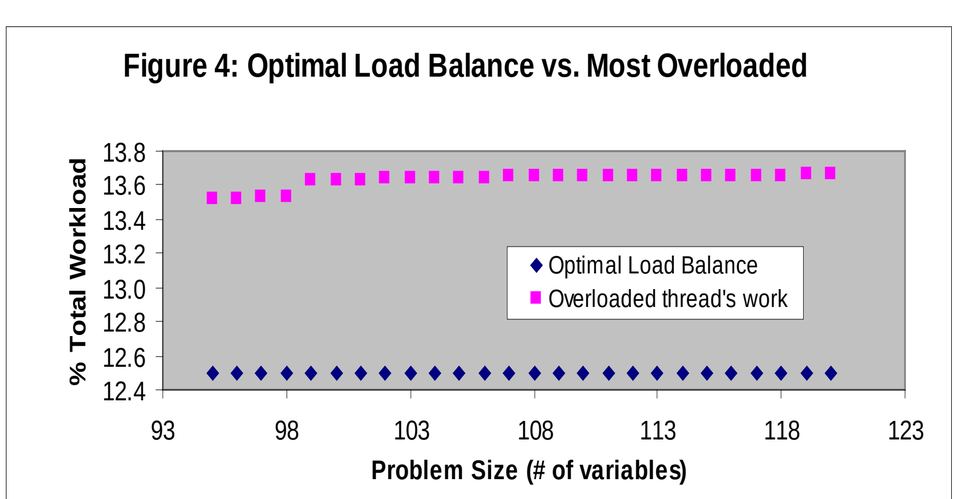
Figure 2 shows a significant reduction in wall time; which suggests that CBFS is a good candidate for parallelization. However, Figure 3 shows diminishing returns when problem size is increased.

Figure 3: Serial Wall Time / Parallel Wall Time



To determine the root cause of this trend, we examined the % of the total problems being computed by the thread with the most work. The results of this examination are shown in Figure 4.

Figure 4: Optimal Load Balance vs. Most Overloaded



As Figure 4 illustrates, the more overloaded thread carries far more work that it should for optimal load balance. Because this imbalance in work grows steadily with the size of the problems, we attribute the diminishing returns in Figure 3 to this phenomenon.

V. Conclusions

Our results show the tremendous benefits of applying parallelism to CBFS. We achieved speedups of 10-11x, and with an increased number of processors as well as refined load balancing, our work suggests even faster possible implementations. B&B problems which are currently time-infeasible may become solvable.

Future work might delve into heuristic best stopping times for threads to communicate, as well as into methods to use when shared memory is not an option.

Further refinement of a load balancing framework we have implemented but not yet tested, as well as testing scalability to orders of magnitude larger problems are also good areas for future research.

VII. Literature Cited

[1]. Zhang, W., and Korf, R. E. "Depth-First vs. Best-First Search: New Results." (1993): n. pag. Association for the Advancement of Artificial Intelligence. 21 July 2012.

[2]. A BB&R algorithm for minimizing total tardiness on a single machine with sequence dependent setup times. Journal of Global Optimization. Sewell, E.C., Sauppe, J.J., Morrison, D.R., Jacobson, S.H., and Kao, G. To appear, 2012.

[3]. S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementation, John Wiley and Sons, 1990

[4]. Horowitz, E.; Sahni, S. (1974), "Computing partitions with applications to the knapsack problem", *Journal of the Association for Computing Machinery* **21**: 277–292

[5]. Gendron, B., and Crainic, T. G. "Parallel Branch and Bound Algorithm Survey and Synthesis." *Operations Research* 42 (1994): 1042-066. *INFORMS*. 25 June 2012.

VIII. Acknowledgements

Special Thanks to Dr. Sheldon Jacobson, Dr. Craig Zilles, David Morrison, Jason Sauppe, and Jill Peckham, as well as the University of Illinois at Urbana-Champaign, Swarthmore College Spelman College, and the National Science Foundation for funding our research.

Contact Us:

Joshua Gluck:
jgluck2@swarthmore.edu
Nartezya Dykes:
ndykes@scmail.spelman.edu
Dr. Sheldon Jacobson
shj@illinois.edu

Passionate on Parallel
2012 REU Program

parallel COMPUTING INSTITUTE

This REU is co-funded by the ASSURE program of the Department of Defense in partnership with the National Science Foundation REU Site program under Award No. 1004311.