POSTER: A Parallel Branch-and-Bound Algorithm with History-Based Domination

Taspon Gonggiatgul
California State University, Sacramento
Sacramento, California, USA
gonggiatgul@csus.edu

Ghassan Shobaki
California State University, Sacramento
Sacramento, California, USA
ghassan.shobaki@csus.edu

Pınar Muyan-Özçelik
California State University, Sacramento
Sacramento, California, USA
pmuyan@csus.edu

Abstract

In this paper, we describe a parallel Branch-and-Bound (B&B) algorithm with a history-based domination technique, and we apply it to the Sequential Ordering Problem (SOP). To the best of our knowledge, the proposed algorithm is the first parallel B&B algorithm that includes a history-based domination technique and is the first parallel B&B algorithm for solving the SOP using a pure B&B approach. The proposed algorithm takes a pool-based approach and employs a collection of novel techniques that we have developed to achieve effective parallel exploration of the solution space, including parallel history domination, history table memory management, and a thread restart technique. The proposed algorithm was experimentally evaluated using the SOPLIB and TSPLIB benchmarks. The results show that using ten threads with a time limit of one hour on the medium-difficulty instances, the proposed algorithm gives a geometric-mean speedup of 19.9 on SOPLIB and 10.23 on TSPLIB, with super-linear speedups up to 65x seen on 17 instances.

CCS Concepts: • **Computing methodologies → Parallel algorithms**; • **Theory of computation → Branch-and-bound**.

Keywords: parallel branch-and-bound, sequential ordering problem, combinatorial optimization, NP-complete problems

1 Summary

In this paper, we describe a parallel version of a Branch-and-Bound (B&B) algorithm that has a history-based domination technique. Although history-based domination substantially speeds up a B&B search, it makes it much more challenging to parallelize the algorithm. The proposed parallel B&B algorithm is applied to the Sequential Ordering Problem (SOP). To the best of our knowledge, the proposed algorithm is the first parallel B&B algorithm with a history-based domination technique and is the first parallel B&B algorithm for solving the SOP using a pure B&B approach.

The SOP is a generalization of the Traveling Salesman Problem (TSP), which is a well-known NP-hard problem. Given a weighted graph and a dependence graph representing precedence constraints among the vertices, the objective in the SOP is finding a minimum-cost Hamiltonian path in the weighted graph that satisfies the precedence constraints in the dependence graph. Precedence constraints make developing a parallel B&B algorithm more challenging, because they make it harder to estimate and balance thread loads.

The proposed parallel algorithm is a pool-based algorithm that consists of a collection of techniques that we have designed to effectively search the solution space using multiple threads. These techniques include global-pool assignment, load balancing, parallel history-based domination, history table memory management, and a thread restart technique.

The proposed parallel algorithm is based on the sequential B&B algorithm that was originally proposed by Shobaki and Jamal [5] and later enhanced by Jamal et al. [2] for solving the SOP. This sequential algorithm includes a history-based domination technique that stores information about previously explored sub-problems in a history table and uses that information to speedup the processing of similar sub-problems. That technique is a generalization of the history-based domination technique that was originally introduced by Shobaki and Wilken [6].

What distinguishes parallel search algorithms like B&B from other types of parallel computation is the possibility of achieving *super-linear speedup* relative to the sequential algorithm, that is, a speedup ratio that is greater than the number of parallel threads. Super-linear speedup is possible in search algorithms, because the performance is highly dependent on the search order, and the search order of the parallel algorithm can be better than that of the sequential algorithm. Performance is highly dependent on the search order, because the degree of pruning at a given point depends on the value of the best solution found at the point, and the best solution found is dependent on the search order.

A main contribution of this work is a dynamic thread restart technique that guides the search order to focus on the

PPoPP '22, April 2–6, 2022, Seoul, Republic of Korea

ACM ISBN 978-1-4503-9204-4/22/02.
https://doi.org/10.1145/3503221.3508415

most promising sub-spaces. The technique is inspired by the work of Anderson et. al [1]. While the algorithm of Anderson et. al is a sequential algorithm, our algorithm is a parallel algorithm that uses multiple threads to balance the exploitation of promisingness information and the exploration of new sub-spaces. Balancing exploitation and exploration is a unique feature that distinguishes our algorithm.

When history-based domination is used, super-linear speedup will be even more likely. Within the same time period, a parallel algorithm explores different sub-spaces simultaneously. This does not only increase the chances of finding a better best solution, but it also stores more diverse information in the history table, thus increasing the chances of applying history-based pruning.

Another important contribution of this work is a parallel version of history-based domination that we have developed to eliminate redundant concurrent exploration of similar sub-spaces. This is made possible with an enhanced history table that provides more information about the exploration state below a dominated history node. If the current thread is about to explore a dominant node, it can use that history information to stop the exploration of a dominated node and possibly reuse some of the pruning that has taken place during the exploration of the dominated node. This technique is called the thread stop/resume technique.

2 Experimental Results

The proposed algorithm was experimentally evaluated using the SOPLIB [3] and the TSPLIB [4] benchmark suites on two different machines with multi-core processors. The *main machine* that is owned by our group has a 10-core i9-9900x processor with 64GB of memory. In order to run the hardest instances with longer time limits within reasonable time, multiple external machines with identical configurations were rented. Each external machine has an AWS Graviton processor with 64 physical cores and 256 GB of memory.

Using 10 threads with a time limit of one hour on the medium-difficulty instances, the proposed algorithm gives a super-linear geometric-mean speedup of 19.9 on SOPLIB and 10.23 on TSPLIB as shown in Table 1. Super-linear speedups up to 65x are seen on 17 medium-difficulty instances.

On the hard instances, the proposed parallel algorithm solves seven SOPLIB instances and one TSPLIB instance that the sequential algorithm does not solve within an hour as shown in Table 2. The geometric-mean speedup lower bound on the hard instances is 5.78 on SOPLIB and 8.91 on TSPLIB. Super-linear speedups up to 220x are seen on two hard instances.

The instances that could not be solved within an hour on the main machine were processed by the proposed algorithm on the external machine with a time limit of five hours, and that solved seven more SOPLIB instances and two more TSPLIB instances that could not be solved within an hour on the main machine.

Table 1. Speedup of medium-difficulty instances

SOPLIB	4 Threads	8 Threads	10 Threads
Instances solved	19	19	19
Geo-mean speedup	9.73	17.78	19.9
Maximum speedup	36.05	64.92	65.37
Minimum speedup	2.92	4.43	4.77
Super-linear speedups	17	15	15
TSPLIB	**4 Threads**	**8 Threads**	**10 Threads**
Instances solved	6	6	6
Geo-mean speedup	4.04	8.09	10.23
Maximum speedup	8.3	23.29	29.74
Minimum speedup	2.15	3.62	4.58
Super-linear speedups	3	2	2

Table 2. Hard instances solved by the parallel algorithm

	SOPLIB	TSPLIB
Total hard instances	16	23
Solved by 4 threads in one hour	3	1
Solved by 8 threads in one hour	6	1
Solved by 10 threads in one hour	7	1
Solved by 64 threads in five hours	14	3

3 Conclusions and Future Work

In this paper, we propose a parallel B&B algorithm with history-based domination and apply it to the SOP. The proposed parallel algorithm greatly speeds up the search relative to the sequential algorithm, with super-linear speedup seen on many instances.

In future work, we will continue to work on enhancing the proposed algorithm, focusing on history-based domination and memory management.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1911235

References

[1] D. Anderson, G. Hendel, P. Le Bodic, and M. Viernickel. 2019. Clairvoyant restarts in branch-and-bound search using online tree-size estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1427–1434.

[2] J. Jamal, G. Shobaki, V. Papapanagiotou, L.M. Gambardella, and R. Montemanni. 2017. Solving the sequential ordering problem using branch and bound. In *IEEE Symposium Series on Computational Intelligence*. 1–9.

[3] R. Montemanni, D.H. Smith, and L.M. Gambardella. 2008. A heuristic manipulation technique for the sequential ordering problem. *Computers & Operations Research* 35, 12 (2008), 3931–3944.

[4] G. Reinelt. 1991. TSPLIB—A traveling salesman problem library. *INFORMS Journal on Computing* 3, 4 (1991), 376–384.

[5] G. Shobaki and J. Jamal. 2015. An exact algorithm for the sequential ordering problem and its application to switching energy minimization in compilers. *Comput. Optim. Appl.* 61, 2 (2015), 343–372.

[6] G. Shobaki and K. Wilken. 2004. Optimal Superblock Scheduling Using Enumeration. In *Proceedings of the 37th Annual IEEE/ACM International Symposium on Microarchitecture*. 283–293.