Optimizing Determinism and Parallelism in Branch-and-Bound Trees for MIP Problems

Background

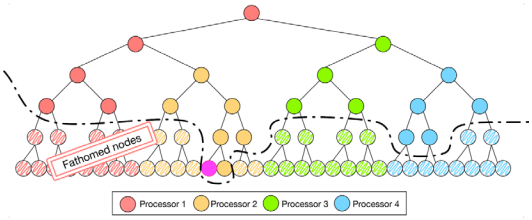

Figure 1 Mapping between subnodes and processors (threads) [1]

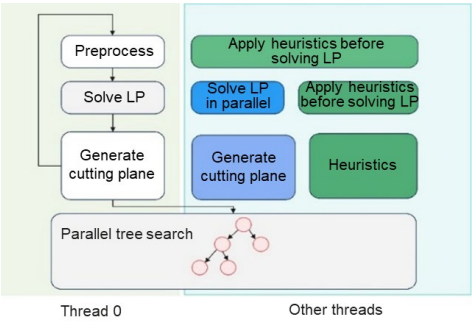

Figure 2 Parallelism of a MIP solver

- Mixed integer programming (MIP) is an extension of linear programming and may include continuous and integral variables. It is widely used in domains such as production planning, logistics scheduling, and resource allocation.
- MIP problems are typically NP-hard as they cannot be solved within polynomial time. Therefore, MIP solvers use advanced optimization algorithms (such as the branch-and-bound, cutting plane, interior-point, and heuristic algorithms) to accelerate the solving process.
- Enhanced multi-core computing capabilities of computers enable MIP solvers to execute these advanced optimization algorithms in parallel, significantly reducing computing time. As illustrated in Figure 1, the branch-and-bound algorithm maps subnodes to separate processors for parallel solving. This enables the simultaneous search for a larger number of nodes within the same timeframe.
- Achieving efficient load balancing among processors requires accurate evaluation and even distribution of subtask workloads. Additionally, optimizing computing efficiency demands timely synchronization of information during the solving process.
- While cutting plane and heuristic algorithms can speed up problem-solving, they often exacerbate load imbalance. Implementing parallelism for these algorithms further complicates solver development, as illustrated in Figure 2.

Technical Challenges

- **Deterministic searching path:** To ensure consistent results and processes, multiple runs must produce identical sequences. A common approach is to design deterministic, parallel schemes with built-in wait mechanisms. However, improper synchronization timing can lead to prolonged thread idling, resulting in inefficient use of computing resources.
- **Efficient memory use:** To prevent data conflicts, parallel solvers often replicate models and intermediate data for each thread, isolating the data to ensure data privacy. The exponential growth of nodes during MIP solving substantially increases the memory overhead. As the degree of

parallelism (DOP) increases, memory consumption can reach system limits, constraining the efficient use of multi-core CPU resources.

- **Effective DOP enhancement:** Increasing DOP in tree search generally improves search efficiency, but the parallel speed-up ratios of MIP solving algorithms do not scale linearly with DOP. Excessively increasing the number of nodes without careful management can lead to low parallel speed-up ratios, wasting computing resources.

Current Scenario & Achievements

- A deterministic, parallel tree search strategy has been developed using two deterministic matrices: the number of nodes and the number of sub-algorithm iterations. While this strategy ensures determinism, it leads to inconsistent CPU utilization. Even with eight threads, it only achieves a maximum speed-up ratio of **1.8**.
- The memory of individual threads is underutilized, and the memory overhead **increases nonlinearly** with DOP.
- Although parallel tree search significantly increases the number of nodes processed per unit time, it fails to improve search efficiency for certain problems.

Technical Requirements

- Design **deterministic metrics** that simulate the wall clock timer more precisely. Apply these metrics to MIP solving algorithms to maximize CPU efficiency (> **90%**) and minimize the total wait time (< **10%**), as validated by **Benchmark 2017** test results.
- Design a data sharing and node storage scheme for MIP problems, ensuring that the memory overhead increases **linearly** with the number (≤ 32) of threads.
- Design a deterministic, parallel branch-and-cut algorithm for MIP problems. The algorithm must achieve a minimum speedup of **2.2x** when utilizing 8 threads compared to single-thread execution.

References

[1] Munguía L M, Oxberry G, Rajan D, et al. Parallel PIPS-SBB: multi-level parallelism for stochastic mixed-integer programs[J]. Computational Optimization and Applications, 2019, 73: 575-601.

[2] Niu Y S, You Y, Liu W Z. Parallel dc cutting plane algorithms for mixed binary linear program[C]//World Congress on Global Optimization. Cham: Springer International Publishing, 2019: 330-340.

[3] Schryen G. Parallel computational optimization in operations research: A new integrative framework, literature review and research directions[J]. European Journal of Operational Research, 2020, 287(1): 1-18.