A Hybrid Massively Parallel Solver for MIPs

Hybrid Massively Parallel Mixed Integer Programming to Solve Challenging Optimization Instances for the First Time

***Y.Shinano, T. Koch**, Zuse Institute Berlin*

In Short

- Development of a hybrid massively parallel mixed integer programming problem(MIP) solver.
- Solve previously unsolved challenging MIP instances from MIPLIB.

The project deals with solving mixed integer programming (MIP) problems in parallel. Many optimization problems arising in practice are modeled as MIP, see, e.g., [5]. A MIP is given in the following general form:

$$\min\{c^\top x : Ax \leq b, l \leq x \leq u, x_j \in \boldsymbol{Z}, \forall j \in I\}, \quad (0.1)$$

with matrix $A \in \mathbb{R}^{m\times n}$, vectors $b \in \mathbb{R}^m$ and $c, l, u \in \mathbb{R}^n$, and a subset $I \subseteq \{1, \ldots, n\}$. The standard algorithm used to solve MIP is an LP-based branch-and-bound, which implicitly enumerates the whole solution space to find an optimal solution x^* that gives the minimum value of (0.1).

State-of-the-art MIP solvers are based on the branch-and-cut paradigm, which is a mathematically supercharged mixture of a branch-and-bound tree search combined with a cutting plane approach, employing a large number of sophisticated algorithms to keep the enumeration effort small. These additional algorithms include a large number of heuristic methods to devise primal feasible solutions, and many cutting plane separation algorithms to increase the lower bound value obtained by the Linear Programming (LP) relaxation, see, e.g., [5].

The FICO Xpress-Optimizer features software tools to solve linear, integer, quadratic, nonlinear, and robust optimization problems [3,4]. The solver uses varying techniques at different nodes of the branch-and-bound search for diversification. It learns from infeasible subproblems [15] and uses Machine Learning techniques to avoid numerical inaccuracies [1].

Tree search algorithms are generally considered easy to parallelize. However, to the best of our knowledge, there have been only two implementations of a large-scale parallelized MIP solver that succeeded in solving open benchmarking instances. One is GAMS/CPLEX/Condor by Bussieck et al. [2] who solved three models from MIPLIB2003 by a GRID computing approach. The other is ParaSCIP [8], one of our previous works. ParaSCIP is an external parallelization of the open-source solver SCIP [7] and has been developed using the *Ubiquity Generator (UG) framework* [12,14]. UG is a general software framework to parallelize MIP solvers externally. Both solvers use a state-of-the-art MIP solver as a black box to exploit the latest MIP solving technology; the tree-search-based solving process is parallelized externally. For a recent survey about parallel MIP solving, see [6].

This project aims to develop a hybrid, massively parallel solver for MIP based on the (Generalized) Ubiquity Generator (UG) framework. We had an experimental implementation to check how well Xpress integrates within UG; we called this project ParaXpress [**?**, **?**]. The result was encouraging but showed clear room for future improvements. One reason is that the same presolved instance, which is a reformulated problem with a strengthened feasible region, needs to be generated each time when a solver receives a sub-MIP. In a follow-on work, publication in preparation, we developed a layered presolving mechanism to overcome this shortage. Further, we extended our implementation by a racing ramp-up feature [9,12] . Even though the paper is not published yet, the optimal solutions for the following open instances from MIPLIB2010 generated by UG with Xpress as an underlying black-box MIP solver are published as on the MIPLIB page.

gmut-75-50 Timber harvest scheduling model. Solved by UG with Xpress in a 12288 core supercomputer run on HLRN III.

gmut-77-40 Timber harvest scheduling model. Solved by UG with Xpress in a 12288 core supercomputer run on HLRN III.

ger50_17_trans Multi-layer network design problem using a link-flow formulation over a path-flow formulation. A MIP start (initial solution) was given, which was generated during UG with Xpress development and tested in the first implementation.

In June 2021, Xpress version 8.12 was published, with many new features, which will allow us for even more robust integration of Xpress as a MIP solver component into the newly extended UG framework that we develop.

We made an experimental implementation for a non-branch-and-bound-based solver based on UG to solve Shortest Vector Problems (SVPs) [13]. The

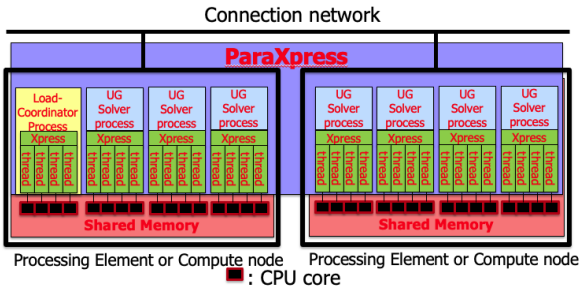

Figure 1: *Combining UG processes and MIP solver (here: Xpress) threads can handle more than a million CPU cores. Ex. 80,000 (MPI processes) × 24 (Xpress threads) = 1,920,000.*

implementation inspired us to build a *generalized UG* which can parallelize non-branch-and-bound solvers naturally as a single software framework. The parallelization of non-branch-and-bound solvers is the focus of project bem00052. However, it has an interesting side effect relevant for the proposed project bem00058. In the generalized UG, the controller process code of UG is abstracted, which allows UG user to fully customize the ramp-up mechanism, dynamic load balancing, and other UG routines. Consequently, in this project, we will by able to treat the underlying MIP solver completely as an interchangeable black-box while still exploiting its full functionality.

This project aims to (further) develop the parallel MIP-solver-based UG to handle over 1,000,000 CPU cores to solve other notoriously hard MIP instances to optimality. The key point addressed in this proposal is how to combine UG's external parallelization with the internal parallelization of modern MIP solvers (note that SCIP is only single-threaded). See Figure 1 for the principal design. We will implement several parallelization schemes for the algorithms and a new ramp-up mechanism. Those that are already implemented will be scientifically evaluated for the first time. After implementing and testing all of the proposed features, the results will be published in peer-reviewed international journals. We believe that this project will serve as a flagship for the efficient solution of MIP by using supercomputers.

WWW

https://www.zib.de/members/shinano,
https://www.zib.de/berthold,
https://ug.zib.de/

More Information

[1] T. Berthold and G. Hendel, *Learning to scale mixed-integer programs*, (2020). http://www.optimization-online.org/DB_HTML/2020/12/8166.html.

[2] M. R. Bussieck, M. C. Ferris, and A. Meeraus, *Grid-enabled optimization with GAMS*, IJoC, 21 (2009), pp. 349–362.

[3] *FICO Xpress-Optimizer*. http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx.

[4] R. Laundy, M. Perregaard, G. Tavares, H. Tipi, and A. Vazacopoulos, *Solving hard mixed-integer programming problems with Xpress-MP: a MIPLIB 2003 case study*, INFORMS Journal on Computing, 21 (2009), pp. 304–313.

[5] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*, Wiley, 1988.

[6] T. K. Ralphs, Y. Shinano, T. Berthold, and T. Koch, *Parallel solvers for mixed integer linear optimization*, in Handbook of Parallel Constraint Reasoning, Y. Hamadi and L. Sais, eds., Springer International Publishing, 2018, pp. 283–336.

[7] *SCIP: Solving Constraint Integer Programs*. https://scipopt.org.

[8] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, and T. Koch, *ParaSCIP – a parallel extension of SCIP*, in Competence in High Performance Computing 2010, C. Bischof, H.-G. Hegering, W. E. Nagel, and G. Wittum, eds., Springer, 2012, pp. 135–148.

[9] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, T. Koch, and M. Winkler, *Solving hard MIPLIB2003 problems with ParaSCIP on supercomputers: An update*, in Parallel Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International, May 2014, pp. 1552–1561.

[10] Y. Shinano, T. Berthold, and S. Heinz, *A first implementation of paraxpress: Combining internal and external parallelization to solve mips on supercomputers*, in Mathematical Software – ICMS 2016, G.-M. Greuel, T. Koch, P. Paule, and A. Sommese, eds., Cham, 2016, Springer International Publishing, pp. 308–316.

[11] Y. Shinano, T. Berthold, and S. Heinz, *Paraxpress: an experimental extension of the fico xpress-optimizer to solve hard mips on supercomputers*, Optimization Methods and Software, 33 (2018), pp. 530–539.

[12] Y. Shinano, S. Heinz, S. Vigerske, and M. Winkler, *FiberSCIP—a shared memory parallelization of SCIP*, INFORMS Journal on Computing, 30 (2018), pp. 11–30.

[13] N. Tateiwa, Y. Shinano, S. Nakamura, A. Yoshida, M. Yasuda, S. Kaji, and K. Fujisawa, *Massive parallelization for finding shortest lattice vectors based on ubiquity generator framework*, in 2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Los Alamitos, CA, USA, nov 2020, IEEE Computer Society, pp. 834–848.

[14] *UG: Ubiquity Generator framework*. http://ug.zib.de/.

[15] J. Witzig, T. Berthold, and S. Heinz, *Experiments with conflict analysis in mixed integer programming*, in International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Springer, 2017, pp. 211–220.

Project Partners

Dr. T.Berthold, Fair Isaac Germany GmbH

Funding

BMBF Research Campus MODAL