UG Synthesizer

Synthesizing different types of parallel solvers on a supercomputer to solve now intractable optimization instances

***Thorsten Koch, Yuji Shinano, Ambros M. Gleixner**, Technische Universität Berlin/Zuse Institute Berlin*

In Short

- *UG Synthesizer (UGS)* is a new framework to flexibly realize any combinations of algorithm portfolios and racing to solve Mixed Integer Linear Programs (MILPs) on a distributed memory computing environment.
- It exploits and extends the seasoned *Ubiquity Generator (UG) framework* designed to port powerful single-solver MILP algorithms to HPC environments.
- In order to check its generality, at least one problem-specific solver should be parallelized.

In previous research we have developed distributed memory Mixed Integer Linear Programming (MILP) solvers ParaSCIP [1–3] and ParaXpress [4] based on *Ubiquity Generator (UG) framework* [5]. The solvers are successful in terms of solving previously unsolvable instances (open instances) on supercomputers. Figure 1 shows the number of open instances in MIPLIB2010 [6] solved during the last eight years. The instance set was published in 2011. In 2012, many instances were solved by standard MILP solvers, but these monotonously decreases by 2017. However, UG kept adding solved instance rather continuously. The increase in 2015 is due to the update of HLRN II to HLRN III. In 2016, ParaXpress was developed, and in 2017, two new instances were solved by ParaXpress, but it becomes clear that also on HPC new algorithmic approaches are needed to tackle hard instances.

The strategy of composing multiple heuristic algorithms within a single solver that chooses the best suited one for each input is called *algorithm portfolio*. In order to exploit performance variability [6] for MILP solving, a solver may solve an instance in parallel with several different configurations of parameters (including parameter for permutation of columns and rows of input data). This procedure is called *racing*. UGS is a general framework to realize any combinations of algorithm portfolio and racing on a distributed memory computing environment. Within this project, we develop UGS as an MPMD (Multiple-Program Multiple-Data) program to run stably with three or more different solver executable files, including at least one distributed memory SPMD (Single-Program Multiple-Data) type MPI program.

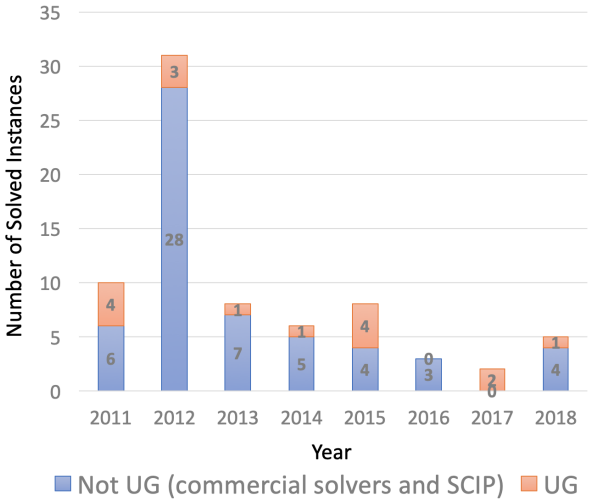

***Figure 1:** Number of open instances in MIPLIB2010 solved during last six years*

The idea to use an improved solution found by using some external heuristic MILP solver simply for restarting parallel B&B MILP solver execution is natural, easy to realize, and promising. With mathematically supercharged MILP solvers, however, many challenges arise. At first, finding the improved solution itself is extremely hard in general for hard MILP instances. It could be improved if the problem has special structure and the heuristic algorithm used is to specialized for the problem. In general MILP case, currently, only strong commercial MILP solvers could have a chance to improve the incumbent solution. Second, even if an improved solution was obtained, it often cannot be used directly in a different B&B search because current state-of-the-art MILP solvers reformulate problem structure in a preprocessing phase. Therefore, an automatic transformation procedure is necessary.

All things considered, we need a synthesized system that needs to satisfy below:

- Different types of algorithm implementations for solving MILP problem need to be run in parallel.
- All algorithms run in parallel must be state-of-the-art, since each of them is expected to contribute to improving the incumbent solution.
- Each algorithm implementation needs to be realized by a separated executable file, since which algorithms are used depends on problem to be solved and the parallel solver should be configured at run-time.
- An algorithm implementation could be a program which can run on a distributed memory

computing environment itself.

The goal of this project is to develop *UG Synthesizer (UGS)* that allows us to realize the synthesized system as an MPMD type MPI program. An instantiated distributed memory solver by using UG such as ParaSCIP and ParaXpress is a SPMD type MPI program. An MPMD MPI program realized by UGS can have two levels of distributed memory MPI programs, i.e., it could contain SPMD type MPI program inside of the MPMD MPI program. On top of that, we would like to run multiple SPMD type MPI programs instantiated by UG with different configurations of parameters in parallel.

Figure 2 shows the design structure of UGS. UGS is a software tool kit that contains scripts to generate run-time environment from a parallel processes configuration file. The run-time processes on a supercomputer composed of a special process `ugs` which mediates solution sharing and *ugs solvers*. The latter can be several different executable files. In order to make it possible to communicate between ugs solvers and `ugs`, a special MPI communicator is provided by UGS.

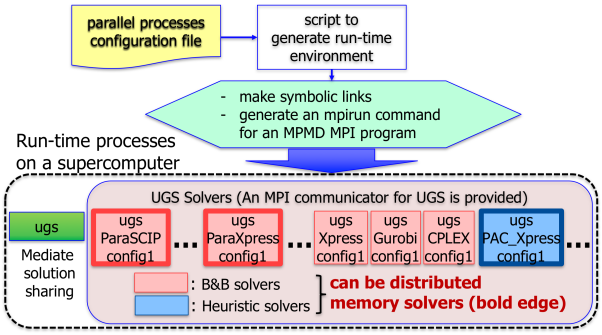

Figure 2: *Design structure of UG Synthesizer (UGS)*

Though our prime focus, UGS is not limited to MILP solvers only, but can be used for MINLP solvers and be extended to more general problem and algorithm classes. In future investigations, one valuable application of UGS could be block-structured MILPs, which appear in the context of energy systems modeling such as in our collaborative project BEAM-ME[1]. We already have ug[PIPS-SBB, MPI] [7], which is a branch-and-bound-based solver specialized for Stochastic MILPs (SMILPs). SMILPs need to solve extremely large Linear Programming (LP) relaxations with special structure called dual block-angular constraint matrix. Therefore, in ug[PIPS-SBB, MPI], the LPs are solved on distributed memory and branch and bound tree search is parallelized by UG. This means ug[PIPS-SBB, MPI] itself has two levels of MPI parallel program and it needs to be extended so that it can handle three levels when using UGS.

[1] `http://www.beam-me-projekt.de/`

WWW

`http://ug.zib.de/`

More Information

[1] Y. Shinano, T. Achterberg,T. Berthold, S. Heinz, T. Koch, ParaSCIP – a parallel extension of SCIP, Christian Bischof, Heinz-Gerd Hegering, Wolfgang E. Nagel, Gabriel Wittum, eds., *Competence in High Performance Computing 2010*. Springer, 135–148 (2012).

[2] Y. Shinano, T. Achterberg,T. Berthold, S. Heinz, T. Koch, M. Winkler, Solving Hard MIPLIB2003 Problems with ParaSCIP on Supercomputers: An Update, *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International* 1552 – 1561 (2014).

[3] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, T. Koch, M. Winkler, Solving Open MIP Instances with ParaSCIP on Supercomputers using up to 80,000 Cores, in 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE Computer Society,770–779,(2016).

[4] Y. Shinano, T. Berthold, and S. Heinz, *A First Implementation of ParaXpress: Combining Internal and External Parallelization to Solve MIPs on Supercomputers*, Springer International Publishing, Cham, 2016, pp. 308–316.

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

[6] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold,R.E. Bixby, E. Danna, G. Gamrath, A.M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T.K. Ralphs, D. Salvagnin, D.E. Steffy, K. Wolter, MIPLIB 2010 – Mixed Integer Programming Library version 5, *Mathematical Programming Computation*, **3** 103–163, (2011).

[7] L.-M. G. O. Munguía, D. Rajan, and Y. Shinano, *Parallel pips-sbb: Mulit-level parallelism for stochastic mixed-integer programs*, Tech. Rep. 17-58, ZIB, Takustr.7, 14195 Berlin, 2017.

Project Partners

Oak Ride National Laboratory, Lawrence Livermore National Laboratory, The Institute of Statistical Mathematics, Georgia Tech.

Funding

Forschungscampus Modal