# Integration of Polyhedral Outer Approximation Algorithms with MIP Solvers Through Callbacks And Lazy Constraints

Andreas Lundell[1,a)] and Jan Kronqvist[2]

[1]*Mathematics and Statistics, Faculty of Science and Engineering, Åbo Akademi University, Turku, Finland.*
[2]*Process Design and Systems Engineering, Faculty of Science and Engineering, Åbo Akademi University, Turku, Finland.*

[a)]Corresponding author: andreas.lundell@abo.fi

**Abstract.** In this paper, it is explained how algorithms for convex mixed-integer nonlinear programming (MINLP) based on polyhedral outer approximation (POA) can be integrated with mixed-integer programming (MIP) solvers through callbacks and lazy constraints. Through this integration, a new approach utilizing a single branching tree is obtained which reduces the overhead required when rebuilding the branching tree in the MIP solver due to the continuous addition of linear constraints approximating the nonlinear feasible region of the MINLP problem. The result is an efficient strategy for implementing a POA utilized by the Supporting Hyperplane Optimization Toolkit (SHOT) solver.

## INTRODUCTION

Currently, mainly two types of meta-algorithms are utilized for convex MINLP: branch and bound (BB), and PAO. The former category includes solver implementations such as Antigone [1], BARON [2], Couenne [3] and SCIP [4], while the latter category includes AlphaECP [5], Dicopt [6] and SHOT [7]. The classification of solvers into these categories is not completely straight-forward, as they both at one stage or another utilize branching, either directly or when solving subproblems, and most BB-based methods also utilize polyhedral approximation.

In this extended abstract, we will show how it is possible to integrate the PAO method utilized in SHOT more tightly with the MIP subsolver. This is accomplished by utilizing callback functionality provided by some MIP solvers (CPLEX and Gurobi in this case) in combination with so-called lazy constraints to include cuts generated by the PAO in the main branching tree of the MIP solver. The result is that instead of solving multiple MIP problems in sequence as in previous implementations, only one master MIP problem is solved. The preliminary results are promising as indicated by benchmarks shown in this paper.

## MULTI-TREE POLYHEDRAL OUTER APPROXIMATION

In the extended cutting plane (ECP) [8] or the extended supporting hyperplane (ESH) [9] algorithms, a sequence of MIP problems are solved to obtain the lower bound when considering a minimization problem. The lower bound is often referred to as a dual bound. The initial MIP model consists of all the linear and integer constraints in the original MINLP problem. The solution to the MIP problem in each iteration is removed with a linear cut if the point does not fulfill the nonlinear constraints in the MINLP problem, and after this the MIP problem is resolved. As this approximation is improved by adding more of these linear constraints, the dual bound will increase and finally converge to the optimal solution.

Note that, in general, it is not possible to know how close this dual bound is to the optimal solution of the MINLP problem. To remedy this shortcoming, heuristic methods can be utilized that find valid feasible solutions, and these in turn give an upper (primal) bound on the objective function value of the MINLP problem. Then it is possible to calculate absolute and relative objective gaps between the dual and primal bounds, and these gaps give a measure on the progress of the solution process and can be used as termination criteria.

As described in [9] and [10], the intermediate MIP problems need not be solved to optimality, but rather only the final one. For example, the solution limit parameter in the MIP solvers can be used to terminate the solution as soon as a specified number of integer-feasible MIP solutions to the problem in the current iteration have been found. Cuts are then added that remove these solution points if they do not fulfill the nonlinear constraints. If a point is feasible for the original MINLP problem, we cannot cut it away since it may be the optimal solution, so then we instead increase the solution limit and continue to solve the problem. Since the MIP model has not been modified when only increasing the solution limit, this allows the MIP solver to continue with the same branching tree without restarting. At some stage, *i.e.,* when the solution limit parameter is large enough, the MIP solver will start returning optimal solutions to the subproblems. Then, no more solution limit updates are required, and only additional linear cuts are added to remove solutions not feasible in the MINLP problem.

In the AlphaECP solver in GAMS [5], the solution limit is not directly available since not all MIP solvers supported it at the time of implementation. Therefore, a similar strategy utilizing the relative optimality gap tolerance for subproblem termination is employed. Initially, this parameter is set to the value one, and after fulfilling certain conditions it is reduced to finally reaching the value zero, and thus returning optimal solutions to the subproblem. If the solutions returned do not fulfill the nonlinear constraints, additional cutting planes are added until the optimal solution is found.

In PAO-based methods, adding linear constraints invalidates the current MIP branching tree, and the tree needs to be rebuilt whenever this occurs. For simple problems that are solved in just a few iterations, creating a new branching tree per iteration may not present a significant burden, and often in these cases no branching is needed at all, as the problem can be solved in preprocessing by the MIP solver. However, for more complex problems, it is possible to considerably reduce the work needed by considering a more integrated approach with the MIP solver as described in the next section.

## SINGLE-TREE POLYHEDRAL OUTER APPROXIMATION

A similar solution limit strategy as described above is implemented in the multi-tree strategy in SHOT. The default strategy is, however, to integrate further with the MIP solvers CPLEX and Gurobi, the motivation being to remove the additional burden of rebuilding the MIP tree. This single-tree strategy utilizes callback functionality provided by the MIP solver in combination with so-called lazy constraints. Lazy constraints are linear constraints without which the model would be incomplete, but which may not be actively part of the set of constraints utilized by the MIP solver when solving the integer-relaxed subproblems.

To give a rough idea of how the single-tree strategy works, assume that enough supporting hyperplanes have been added to the MIP model to give a PAO of the nonlinear feasible set, so that its solution is equal to the optimal dual solution for the MINLP problem. Then, this solution is also the primal solution to the MINLP problem. Of course, normally not all of the added constraints are binding, and may thus be redundant. Now, imagine it the other way around: The MIP solver has found an integer-feasible solution and we want to determine if it may be one to the original nonlinear problem as well. Then it is easy to check if this solution fulfills all nonlinear constraints. If it does, it gives a new primal solution candidate; if not, linear lazy constraints (either supporting hyperplanes or cutting planes) are added through a callback. The added constraints remove this solution from the MIP model, and the MIP solver acts as if these have always existed. Thus, only the linear constraints that are actually needed are included in the model.

Since all integer feasible solutions that violate one or more of the nonlinear constraints in the original MINLP problem are cut away, the MIP solver will have lower and upper bounds that are valid also for the MINLP problem, and thus when the absolute or relative termination criteria is fulfilled in the MIP solver, the optimal solution to the MINLP problem has been found to the given termination tolerances as well. When the MIP solver terminates with the optimal solution, we know that it is also the optimal solution to the MINLP problem since enough lazy constraints have been added to represent a PAO that is good enough.

In the single-tree strategy in SHOT, each time a new integer-feasible solution is found by the MIP solver, either obtained when branching or utilizing some heuristic method, control is returned to SHOT. In a way, each iteration in the multi-tree strategy is replaced with an activation of the callback. The solver can then check this solution: if it is feasible with respect to the nonlinear constraint, and better (lower when minimizing), it provides a bound on the primal solution. Otherwise, the point can be used for generating a cutting plane or supporting hyperplane. This cut is then added as a lazy constraint to the MIP model and control is returned to the MIP solver which continues to solve the problem using the same tree structure.

**Table 1.** The results of applying the single- and multi-tree solution strategies in SHOT to some test problems from MINLPLib. To be able to compare the strategies better, the number of threads in CPLEX has been set to one. This reduces performance significantly, but otherwise the results vary a lot between runs, making comparisons difficult. The columns are the solution time in seconds, the total number of hyperplane cuts added and the total number of MIP problems solved.

| **Instance** | **Multi-tree strategy** | | | **Single-tree strategy** | | |
|---|---|---|---|---|---|---|
| | Time (s) | Cuts | MIP problems* | Time (s) | Cuts | MIP problems |
| `batchs201210m` | 22.2 | 219 | 88 + 3= 91 | 6.9 | 306 | 1 |
| `clay0305h` | 11.6 | 157 | 128 + 1 = 129 | 3.1 | 372 | 1 |
| `cvxnonsep_normcon20` | 45.1 | 560 | 447 + 80 = 557 | 15.7 | 3479 | 1 |
| `flay05m` | 55.0 | 410 | 122 + 3 = 125 | 11.0 | 450 | 1 |
| `fo7` | 20.6 | 186 | 56 + 1 = 57 | 18.5 | 214 | 1 |
| `o7` | 411 | 155 | 55 + 1 = 56 | 364 | 189 | 1 |
| `rsyn0840m04m` | 8.9 | 247 | 33 + 2 = 35 | 4.8 | 940 | 1 |
| `sssd22-08` | 41.9 | 128 | 61 + 1 = 62 | 24.2 | 203 | 1 |
| `tls4` | 56.6 | 266 | 75 + 18 = 93 | 7.1 | 255 | 1 |

* The number of MIP problems in the multi-tree strategy indicates: "feasible + optimal = total".

Other tasks, such as primal heuristics are also performed inside the callbacks. Note that it is also possible to inject primal solutions found from external heuristics provided by SHOT (*e.g.,* obtained by solving an NLP problem) in the MIP model whenever a callback is activated. This can significantly increase the performance, as primal heuristics has been shown to be very important in MINLP [11].

## IMPLEMENTATION ISSUES

The way callbacks are implemented is different between MIP solvers. Also, the version of the solver impacts how callbacks are utilized. In CPLEX versions before 12.8, different callbacks are activated depending on what stage the solver is currently in, *e.g.,* there are callbacks activated whenever a new incumbent is found or whenever a new node is created. As of version 12.8 CPLEX also introduced a new type of callback, similar to the approach implemented in Gurobi: A general callback is activated at all these different steps in the optimization, and inside this callback, it is possible to deduce the reason why the callback was called. This reduces the complexity of the user code required to implement several callbacks. There is however still the restriction that some functionality is limited to certain scopes, *e.g.,* it is still only possible to add a lazy constraint to the MIP model whenever a new incumbent solution is found.

## NUMERICAL BENCHMARKS

The multi- and single-tree strategies in SHOT version 0.9.2 are tested on a selection of the convex MINLP problems in MINLPLib [12], and the results are presented in Table 1. All runs were performed on an Intel Xeon 3.6 GHz PC with 32 GB RAM running Linux. The absolute and relative objective termination criteria were both set to 0.001. CPLEX version 12.6.3 was selected as MIP solver, the reason why we use this older version is because it is more stable compared to later versions (12.7 and 12.8) when utilizing the callbacks. As can be seen from the results, the required solution times for the selected instances are in general much lower for the single-tree strategy.

A more comprehensive benchmark is presented in Fig. 1 where the different strategies are compared on all the 366 convex MINLP instances in MINLPLib [12]. Here, the single-tree strategy is clearly more efficient in the beginning, but the multi-tree strategy catches up at about 30 s. The reason for this may be that more time is spent on preprocessing in the multi-tree strategy, but the implementation is also otherwise more mature. The different strategies in SHOT are also compared to the state-of-the-art global MINLP solver BARON (in GAMS version 25.0 with CPLEX as MIP solver). These results indicate that the performance of SHOT surpasses that of BARON for convex problems. A more extensive benchmark is available in [13].

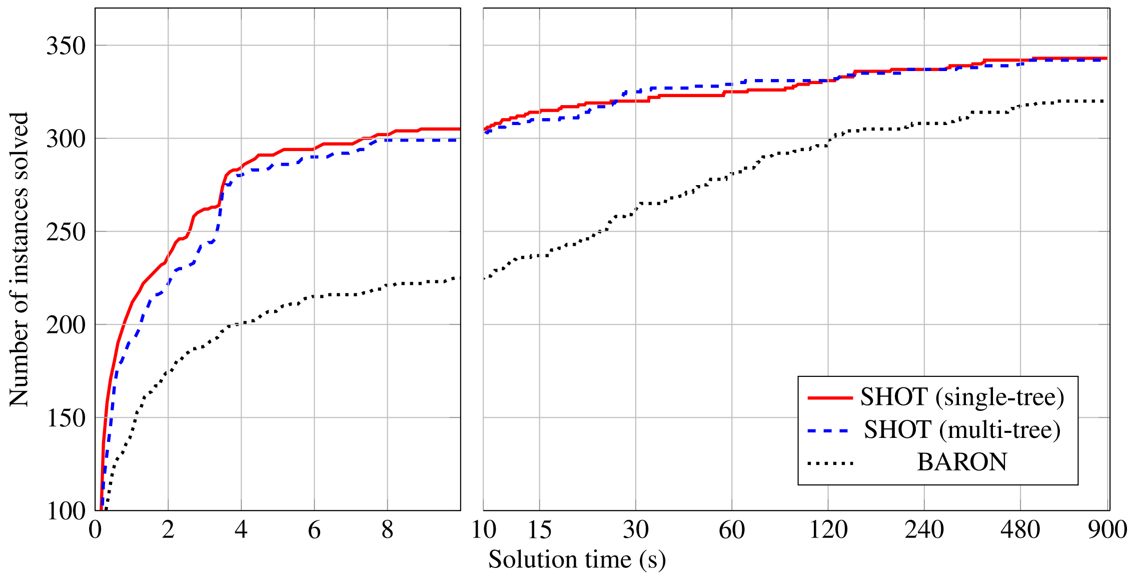

**Figure 1.** The solution profile indicates the number of solved convex MINLP instances in MINLPLib [12] as a function of time. A problem is regarded as solved if the absolute objective gap, as calculated by Paver [14], is ≤ 0.1%

## CONCLUSIONS

In this extended abstract, it was briefly explained how a single-tree strategy was implemented in the SHOT solver for convex MINLP. By utilizing callbacks in combination with lazy constraints, we were able to obtain a significant speedup for several difficult problems in MINLPLib. For a more general benchmark, performance was also promising.

## ACKNOWLEDGMENTS

A. Lundell acknowledges the support from the Magnus Ehrnrooth Foundation.

## References

[1] R. Misener and C. A. Floudas, Journal of Global Optimization **59**, 503–526 (2014).
[2] N. V. Sahinidis, Journal of Global Optimization **8**, 201–205 (1996).
[3] P. Belotti, "Couenne: a user's manual," Tech. Rep. (Lehigh University, 2009).
[4] S. Vigerske and A. Gleixner, "SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework," Tech. Rep. 16-24 (ZIB, 2016).
[5] T. Westerlund and T. Lastusilta, *AlphaECP user's manual*, GAMS Development Corp. (2008).
[6] I. Grossmann, J. Viswanathan, and A. Vecchiette, *DICOPT user's manual*, GAMS Development Corp. (2008).
[7] A. Lundell, J. Kronqvist, and T. Westerlund, Preprint, Optimization Online (2018).
[8] T. Westerlund and F. Pettersson, Computers & Chemical Engineering **19**, 131–136 (1995).
[9] J. Kronqvist, A. Lundell, and T. Westerlund, Journal of Global Optimization **64**, 249–272 (2015).
[10] T. Westerlund and R. Pörn, Optimization and Engineering **3**, 253–280 (2002).
[11] T. Berthold, Journal of Global Optimization **70**, 189–206 (2018).
[12] MINLPLib – a library of mixed-integer and continuous nonlinear programming instances, `http://www.minlplib.org` ( accessed 2018-04-26).
[13] J. Kronqvist, D. E. Bernal, A. Lundell, and I. E. Grossmann, Preprint, Optimization Online (2018).
[14] M. Bussieck, S. Dirkse, and S. Vigerske, Journal of Global Optimization **59**, 259–275 (2014).