Polyhedral approximation strategies for nonconvex mixed-integer nonlinear programming in SHOT

Lundell, Andreas; Kronqvist, Jan

Published in:
Journal of Global Optimization

DOI:
10.1007/s10898-021-01006-1

Published: 20/03/2021

Document Version
Final published version

Document License
CC BY

Journal of Global Optimization (2022) 82:863–896
https://doi.org/10.1007/s10898-021-01006-1

Polyhedral approximation strategies for nonconvex mixed-integer nonlinear programming in SHOT

Andreas Lundell[1] · **Jan Kronqvist**[2]

Received: 16 March 2020 / Accepted: 27 February 2021 / Published online: 20 March 2021

Abstract
Different versions of polyhedral outer approximation are used by many algorithms for mixed-integer nonlinear programming (MINLP). While it has been demonstrated that such methods work well for convex MINLP, extending them to solve nonconvex problems has traditionally been challenging. The Supporting Hyperplane Optimization Toolkit (SHOT) is a solver based on polyhedral approximations of the nonlinear feasible set of MINLP problems. SHOT is an open source COIN-OR project, and is currently one of the most efficient global solvers for convex MINLP. In this paper, we discuss some extensions to SHOT that significantly extend its applicability to nonconvex problems. The functionality include utilizing convexity detection for selecting the nonlinearities to linearize, lifting reformulations for special classes of functions, feasibility relaxations for infeasible subproblems and adding objective cuts to force the search for better feasible solutions. This functionality is not unique to SHOT, but can be implemented in other similar methods as well. In addition to discussing the new nonconvex functionality of SHOT, an extensive benchmark of deterministic solvers for nonconvex MINLP is performed that provides a snapshot of the current state of nonconvex MINLP.

Keywords Nonconvex MINLP · Supporting Hyperplane Optimization Toolkit (SHOT) · Polyhedral outer approximation · Reformulation techniques · Local and global MINLP techniques · Feasibility relaxation

1 Introduction

Mixed-integer nonlinear programming (MINLP) is one of the most versatile optimization paradigms with many applications across engineering, manufacturing and the natural sci-

✉ Andreas Lundell
andreas.lundell@abo.fi

Jan Kronqvist
j.kronqvist@imperial.ac.uk

[1] Department of Information Technologies / Department of Mathematics, Faculty Science and Engineering, Åbo Akademi University, Turku, Finland

[2] Department of Computing, Imperial College London, London, United Kingdom

ences [7,21,27,42,67]. MINLP combines the modeling capabilities of mixed-integer linear programming (MILP) and nonlinear programming (NLP), while at the same time inheriting computational challenges from both fields. The combinatorial features of mixed-integer programming (MIP) in combination with nonlinearities creates a difficult class of mathematical optimization problems. While the practical limits of MINLP are constantly pushed forward through the means of computational and algorithmic improvements, there are still MINLP problems with only a few variables that are difficult to solve. Most of the difficult cases are nonconvex problems, *i.e.*, MINLP problems with either a nonconvex objective function or one or more nonconvex constraints, *e.g.*, a nonlinear equality constraint.

Convex MINLP is a subclass of MINLP, where the nonlinear functions may have desirable properties that can be utilized in algorithms especially tailored for this problem class. There are several convex MINLP algorithms, such as branch and bound [12,30], center-cut [38], decomposition based outer approximation [61], extended cutting plane (ECP) [74], extended supporting hyperplane (ESH) [37], generalized Benders decomposition [24] and outer approximation (OA) [16]. Today, convex MINLP can almost be considered a technology and there are a variety of efficient solvers available [36]. Globally optimizing nonconvex MINLP is, however, still very challenging. Global solvers for nonconvex MINLP include Alpine [62], Antigone [59], BARON [77], Couenne [2], LINDOGlobal [46] and SCIP [25,71]. Gurobi also recently introduced functionality to globally optimize nonconvex mixed-integer quadratically constrained quadratic programming (MIQCQP) problems [31]. These global solvers mainly rely on spatial branch and bound, where convex understimators and concave overestimators are refined in nodes of a branching tree. There are also reformulation techniques that can transform special cases of nonconvex problems, *e.g.*, signomial [53] or general twice-differentiable [50,53], into convex MINLP problems that can then be solved with convex solvers. A decomposition technique to divide large sparse MINLP problems into smaller more tractable MINLP subproblems is presented in [63]. More details on algorithms and solvers for MINLP are given in [5,9,14,68,70].

Due to the computational difficulties of optimizing nonconvex MINLP problems, it may not be possible to obtain a guaranteed optimal solution, or even reasonable bounds on the best possible solution, within a limited amount of time. However, it is not always a necessity to obtain a guaranteed globally optimal solution or tight bounds. Sometimes optimization software users are mainly interested in finding a good-enough feasible solution to the optimization problem within a reasonable computation time. In such situations a local MINLP solver, or a heuristic MINLP technique [13,45], might be the best option. The definition of a local MINLP solver is not completely straightforward, but in this paper we consider a solver without any nonheuristic handling of nonconvexities in optimization problems, such as employing convexifying reformulations or concave/convex over- and underestimating techniques, to be a local solver. For example, a solver based on an algorithm with only guaranteed convergence for convex problems is considered a local one. For nonconvex MINLP problems, a local solver is not guaranteed to find an optimal solution or any feasible solution at all. However, local solvers are often significantly faster than global solvers, and in many cases they manage to return the global solution, or a good approximation of it, even for nonconvex problems. Local MINLP solvers include AlphaECP [41], BONMIN [6], DICOPT [28], Juniper [40], Minotaur [54], Muriqui [55], SBB [23] and SHOT [49].

SHOT is a new solver initially developed for solving convex MINLP problems with guaranteed globality, and extensive benchmarks have shown that the solver is one of the most efficient solvers for this problem class [36]. SHOT is based on a polyhedral outer approximation (POA) approach like many other local solvers, and is therefore tightly inte-

grated with its underlying MILP (mixed-integer linear programming), MIQP (mixed-integer quadratic programming) or MIQCQP (mixed-integer quadratically constrained quadratic programming) subsolver. Solvers based on a POA technique solve a sequence of linear relaxations of the MINLP problem, where cutting or supporting hyperplanes form an outer approximation of the nonlinear feasible set. When a local POA solver is applied to a nonconvex problem, the solver is no longer guaranteed to generate an outer approximation of the feasible set and the cuts may exclude parts of the feasible set. Therefore, a local POA solver may converge to a locally nonoptimal solution or even fail to find any feasible solution.

In this paper, we present some heuristic techniques that have recently been added to SHOT to improve its performance on nonconvex MINLP problems; some of these improvements were briefly mentioned in the conference paper [48]. Although mainly intended to improve the solver's ability to find good feasible solutions, these techniques even enable the solver to find and verify the global optimum in some cases. The techniques discussed in this paper are:

1. Automatic convexity detection:
   - Enables specialized handling of nonconvexities in the problem.
2. Feasibility relaxations through repair techniques of the linear subproblems:
   - Can continue from an infeasible subproblem by expanding the search space and mitigate the impact of bad cuts generated for nonconvex expressions.
3. Objective cutoff constraints:
   - Reduce the chance of the solver terminating at a nonoptimal solution by forcing it to search for better solutions.
4. Return valid bounds also for nonconvex problems:
   - Can verify global optimality for some nonconvex problems.
5. Integer cuts:
   - Exclude specific integer assignments from the search space and increase the speed of convergence.
6. Automatic reformulations:
   - Enable some nonconvexities to be transformed into convex form. Reformulations are also performed for nonlinear equality constraints and absolute value expressions.

These improvements are not unique to SHOT and can also be added to other solvers based on a POA algorithm such as ECP, ESH, or OA. The techniques are tested on a set of 326 nonconvex MINLP problems, and the results show a significant improvement in SHOT's capabilities to find good primal solutions to nonconvex problems. In addition, the new functionality enables SHOT to verify globality for a significant number of problems in the test set, something not previously possible in any local MINLP solver. Finally, this benchmark also provides an overview of the computational efficiency of some of the MINLP solvers available by applying them to the same benchmark set.

2 Background

In this paper, we consider general MINLP problems with the structure

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{x}, \\ \text{subject to} \quad & \mathrm{A}\mathbf{x} \leq \mathbf{a},\ \mathrm{B}\mathbf{x} = \mathbf{b}, \\ & g_k(\mathbf{x}) \leq 0 && \forall k \in K_I, \\ & h_k(\mathbf{x}) = 0 && \forall k \in K_E, \\ & \underline{x}_i \leq x_i \leq \overline{x}_i && \forall i \in I = \{1, 2, \ldots, n\}, \\ & x_i \in \mathbf{R},\ x_j \in \mathbf{Z} && \forall i \in I \setminus I_Z, \forall j \in I_Z, \end{aligned} \tag{1}$$

where I_Z contains the indices of all integer variables. Throughout the paper, we assume that the nonlinear functions g and h are differentiable, but we set no restriction on the convexity of the functions. To simplify the notation by keeping all nonlinearities in the constraints, we will assume that the MINLP problem has a linear objective function. The assumption is not restrictive since a nonlinear objective can always be treated as a linear objective through an epigraph reformulation. However, do note that a nonlinear objective is normally treated separately within the SHOT solver [49] regardless of the conventions used in this paper.

MINLP problems are by definition nonconvex, but they are commonly divided into convex and nonconvex classes based on their continuous relaxation. Therefore, problem (1) is regarded as convex if all the nonlinear functions g_k are convex and $K_E = \emptyset$, *i.e.*, the problem does not have any nonlinear equality constraints [5]. If any of the nonlinear functions in problem (1) are nonconvex, or there are nonlinear equality constraints, the problem is regarded as a nonconvex MINLP. In this paper, we focus on nonconvex MINLP problems so we assume that the problems contain some form of nonconvexity.

In this paper, and often in MINLP terminology in general, the words dual and primal, have special meanings:

Definition 1 The *dual bound* to problem (1) is a valid lower bound on the optimal objective value. A dual bound may be given by the optimal solution point $\mathbf{x}$ of the POA, *i.e.*, $\mathbf{c}^T\mathbf{x}$, or it can just be a lower bound provided by, *e.g.*, the MIP solver. In SHOT, there are two main algorithms, ESH and ECP, for obtaining dual bounds and we refer to these as *dual strategies*. Any feasible solution to the MINLP problem is considered a *primal solution*, and the objective value of the best known feasible solution is called the *primal bound*.

2.1 Polyhedral approximation

In a convex setting, algorithms based on either ECP, ESH or OA can commonly be regarded as *polyhedral outer approximation* (POA) type algorithms. However, since we focus on nonconvex problems, for which these algorithms do not necessarily generate *outer* approximations, we will refer to them as *polyhedral approximation* (PA) based algorithms. We begin by briefly describing PA in a convex setting, before we move on the challenges of using this approach for nonconvex problems.

The main concept behind PA-type algorithms is to construct a POA of the nonlinear feasible set, and use the approximation to form a linear relaxation of the MINLP problem. The POA is defined by a finite set of linear inequality constraints, often referred to as cutting planes or supporting hyperplanes. The cutting and supporting hyperplanes are obtained by linearizing the nonlinear constraints, *i.e.*, a first order Taylor series expansion is used in the

right-hand side of the constraint. If all the constraints in problem (1) are convex, then a POA of the nonlinear feasible set is simply given by

$$g_k(\mathbf{x}^i) + \nabla g(\mathbf{x}^i)^T(\mathbf{x} - \mathbf{x}^i) \leq 0 \quad \forall k \in K_I, \forall i \in 1, \ldots, K, \tag{2}$$

where $\{\mathbf{x}^i\}_{i=1}^K$ is a sequence of points. The POA can be used to generate a linear relaxation of problem (1), which forms the problem

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{x}, \\ \text{subject to} \quad & \mathrm{A}\mathbf{x} \leq \mathbf{a}, \ \mathrm{B}\mathbf{x} = \mathbf{b}, \\ & g_k(\mathbf{x}^i) + \nabla g(\mathbf{x}^i)^T(\mathbf{x} - \mathbf{x}^i) \leq 0 \quad && \forall k \in K_I, \forall i \in 1, \ldots, K, \\ & \underline{x}_i \leq x_i \leq \overline{x}_i && \forall i \in I = \{1, 2, \ldots, n\}, \\ & x_i \in \mathbf{R}, \ x_j \in \mathbf{Z} && \forall i \in I \setminus I_Z, \forall j \in I_Z. \end{aligned} \tag{3}$$

In the convex case, the optimum of problem (3) gives a lower bound on the optimal objective value of the MINLP problem, *i.e.*, a *dual bound* using the terminology in Def. 1.

The main difference between the ECP and ESH algorithms is how the sequence of points $\{\mathbf{x}^i\}_{i=1}^K$ is chosen. With the ECP algorithm, a new linearization point $\mathbf{x}^i$ is directly chosen as the minimizer of problem (3), resulting in a so-called cutting plane [74]. With the ESH algorithm, the linearization points are obtained by approximately projecting the minimizer of problem (3) onto the integer-relaxed feasible set of the MINLP problem, resulting in a supporting hyperplane to the feasible set [37]. The OA algorithm also uses a similar approach of iteratively solving problem (3) and generating new linearization points by solving an NLP subproblem [16,20]. Techniques for utilizing quadratic approximations within an OA framework have also been presented in [35,65]. If a polyhedral approximation technique is combined with convexification procedures and spatial branch and bound, then it can also be employed as a deterministic global optimization technique [69].

If the MINLP problem is nonconvex, the linearized constraints in Eq. (2) will not necessarily outer approximate the feasible set of the problem. Feasible solutions of problem (1) may then also be excluded from problem (3), which may then no longer provide a valid lower bound. Problem (3) may even become infeasible even if the original MINLP problem is feasible. Therefore, directly applying a PA algorithm, such as ECP, ESH or OA, to a nonconvex MINLP problem can result in solutions very far from the global optimum, or even failure to find any feasible solution. Thus, for a local solver to be efficient for nonconvex problems, there is a strong need for some additional (heuristic) techniques to deal with the nonconvexities. The AlphaECP solver in GAMS, which as the name suggests is based on the ECP algorithm, uses several heuristic techniques that have greatly improved its performance for nonconvex problems [41]. For example, AlphaECP uses a strategy of only considering subsets of the cutting planes generated as well as a so-called alpha updating strategy which effectively relaxes the cuts [41,75]. The OA-based DICOPT solver uses the methods of equality relaxation and augmented penalty to improve its performance for nonconvex problems [28,33,72]. A good summary of the additional nonconvex strategies in DICOPT is given in [3].

Extensive benchmarking have shown that SHOT is one of the most efficient solvers for convex problems [36]. However, the tight cuts giving SHOT an advantage for convex problems can actually make the solver perform worse for nonconvex problems. Without any further remedies, SHOT often ends up with an empty search space, *i.e.*, problem (3) being infeasible, without finding any feasible solutions. Therefore, we will present several tech-

niques in Sect. 3 to deal with the challenges of applying a local PA-based solver to nonconvex MINLP problems.

2.2 The SHOT solver

SHOT is an open source solver for MINLP problems[1] [49]. It can be used standalone, or be integrated into modeling systems such as AMPL [22], GAMS [23], JuMP [15] or Pyomo [32]. Like most solvers based on PA, such as DICOPT [28] and BONMIN-OA [6], SHOT only guarantees to find the global solution to convex MINLP instances. These types of solvers utilize a primal-dual strategy, where a lower bound is given by a PA of the nonlinear feasible set and primal solutions are provided by heuristics. The main difference between SHOT and the two other solvers mentioned is how the linear approximation is generated: DICOPT and BONMIN-OA are based on OA, while SHOT utilizes the ECP and ESH algorithms.

SHOT uses two internal representations of the optimization problem, one copy of the original problem on which the primal solutions are verified, and one reformulated version on which the MIP subproblems solved in SHOT's dual strategies are based. Having a separate reformulated version, makes it possible to use lifting-reformulations to, *e.g.*, partition nonlinear expressions in objectives and constraints. As shown in [39], this can have a large impact on the performance when solving certain types of MINLP problems with separable constraints, and this is used extensively in SHOT to rewrite the problem in a format that is more suitable for an PA-based method.

Feasibility-based bound tightening (FBBT) based on interval arithmetic, is an important part of global optimization [56,64]. Bound tightening on the constraints in the linear (and quadratic, if supported) part of the problem is automatically performed by the MIP solver, however, since it is not aware of the nonlinear constraints, all bounding information is not normally carried over to the MIP subproblem. Bound tightening in SHOT is separately performed on both the original problem, which is needed since too loose bounds can disqualify certain automatic reformulations, and on the resulting reformulated problem, on which the MIP subproblems in turn are based.

The main features of SHOT relevant to this paper are summarized in the following sections. For more details, we refer to [49].

2.2.1 Utilizing subsolvers for the MIP subproblems

The PA strategy in SHOT is tightly integrated with the underlying MIP solver, which performs most of the computational work. This means that SHOT's performance is highly dependent on the efficiency of its subsolver. If the MIP solver supports it, SHOT provides a single-tree strategy, where so-called lazy constraint callbacks are used to iteratively add hyperplane cuts without needing to restart the MIP solver. There is also a multi-tree strategy, where the cuts are added after each iteration to form a new MIP subproblem, and where at least in principle the MIP solver starts from the beginning in each iteration. However, as some information, *e.g.*, the currently best solution, is saved between such iterations even in a multi-tree strategy, there is in practice not that much difference in efficiency between the single- and multi-tree strategies if implemented correctly [49]. The subproblems are of the MILP, MIQP or MIQCQP types, depending on the expressions present in the original MINLP problem and what types of expressions the MIP solver supports. If Cbc is used, only MILP subproblems are allowed,

[1] SHOT is a COIN-OR project and is available at github.com/coin-or/shot. More information about the solver is available at www.shotsolver.dev.

and all quadratic constraints need to be considered as general nonlinear and handled by the ECP or ESH algorithm. CPLEX and Gurobi can both handle convex and nonconvex quadratic objective functions, as well as convex quadratic constraints, and then convex quadratic terms does not need to be linearized by adding supporting hyperplanes or cutting planes. As of Gurobi version 9, general nonconvex quadratic constraints are also supported, so in this case nonconvex MIQCQP subproblems are allowed in SHOT; more on this subject in Sect. 5. In general, handling the quadratic expressions in the MIP subsolver is more efficient than utilizing either ECP or ESH methods, especially if the quadratic expressions are nonconvex.

2.2.2 Primal heuristics

The primal strategy in SHOT is (as of version 1.0) based on the following three heuristics for obtaining integer-feasible solutions to the MINLP problem:

- *MIP solution pool*: The MIP solvers often find several valid, but nonoptimal, solutions during a normal run, and these are stored in the so-called solution pool. SHOT checks all of these solutions in case they might fulfill also the nonlinear part of the MINLP problem, in which case they are a valid primal solution.
- *Fixed integer NLP relaxation*: When a solution candidate has been found, *e.g.*, from the MIP solution pool, the integer variables can be fixed to their corresponding values and the resulting NLP problem solved to obtain a new candidate for a primal solution. If the NLP problem is infeasible, an integer cut can be added, *cf.*, Sect. 3.4.
- *Root searches with fixed discrete variables*: When having obtained an integer-valid solution not fulfilling the nonlinear constraints, a root search can be performed that possibly obtain a solution fulfilling the nonlinear constraints as well.

In the future, additional methods such as the center-cut algorithm [34], rounding heuristics [4] or feasibility pumps [1,3,18] are also planned. The NLP relaxations are solved either by interfacing with the NLP solvers in GAMS (if available) or IPOPT [73].

2.2.3 Automatic convexity detection

In SHOT, all nonlinear expressions are by default considered to be nonconvex, and are only regarded as convex if they fulfill some predefined rule. However, most convexity detection is done term-wise, so currently not all convexities are discovered. The convexity of the individual objective function and constraints are more valuable to SHOT than whether the entire problem is convex, since knowing whether a specific constraint is convex or not affects whether a cut generated for this constraint will be globally valid or not.

The convexity detection in SHOT is mostly internal, however for quadratic functions, the Eigen library [29] is used to determine whether their Hessian matrix is positive semidefinite. Monomials are always nonconvex, but the convexity of signomial terms depend on whether the term is positive or negative and on the powers of the variables in the term. However, as there are clear rules, *e.g.*, described in [47], automatic determination of convexity for signomial terms is trivial. For general nonlinear terms, convexity is determined by recursively considering the nodes of the expression tree; inspiration for the used convexity detection rules is taken from [11]. An important part of determining convexity is to have tight bounds, since the available convexity rules for a node in the expression tree are often dependent on what values the underlying expression can attain. Therefore, the convexity detection functionality heavily depends on the bound tightening step described earlier.

3 Strategies for finding and improving local solutions to nonconvex MINLP problems

As mentioned earlier, PA-based methods are normally not able to guarantee optimality of a solution for a nonconvex MINLP problem, but rather they work as a heuristic method that might be able to find a good, perhaps even an optimal, solution. The problem is that the separating hyperplane theorem, which these methods generally rely on, only guarantees that it is possible to find a separating hyperplane between two convex sets. Due to the violation of the separation theorem, the separation techniques commonly used in PA-based methods may not be valid. Thus, whenever cutting planes or supporting hyperplanes are generated to remove a previous solution point from the PA expressed in the MIP problem, we run the risk of cutting away feasible solutions of the original nonconvex problem. Therefore, while the primal bound provided by known integer-feasible solutions are still valid, the dual bound provided by the MIP solver is not as soon as a cut has been generated for a nonconvex constraint. Here, bound tightening is especially important since it may exclude problematic nonconvex parts of the feasible region early on.

As long as no cutting planes or supporting hyperplanes have been added to nonconvex constraints the lower bounds provided by the MIP solver are valid lower bounds also for the nonconvex MINLP problem. These global bounds are stored in SHOT, and can be used for termination on gap tolerance, *e.g.*, if the MIP lower bound is equal or close to the upper bound provided by primal heuristics. Because SHOT can automatically detect convexity of nonlinear constraints, it is possible in some cases to avoid adding cuts for nonconvex constraints; to the best knowledge of the authors, SHOT is the only local solver available today that returns valid lower bounds also for nonconvex problems. Also, even if termination cannot be achieved by closing the objective gap, the lower bound may provide a good indication of the quality of the primal solution. This is one of the reasons that the default nonconvex strategy in SHOT tries to avoid creating cuts for nonconvex constraints as long as possible, *i.e.*, as long as cuts can be added to convex constraints, none are created for nonconvex ones.

By only adding the minimal number of cuts for nonconvex constraints required, the probability that the subproblem (3) becomes infeasible is reduced, and the longer the iterative dual-primal solution process is allowed to continue, the greater is the probability of finding better solutions with the primal heuristics. This is of course a generalization, and there are naturally problem instances where adding several cuts early on and reducing the number of subproblems solved improves the performance. However, as can be seen in the benchmarks later in this paper, AlphaECP and DICOPT are quite efficient at quickly finding a feasible solution, but they struggle at improving these initial solutions, and often have to terminate before finding the optimal one.

In the next example, we will illustrate how the ESH algorithm fails to solve a simple nonconvex MINLP problem. The same example will be used throughout this section to exemplify the nonconvex improvements in SHOT.

Example 1 We will now consider a simple nonconvex MINLP problem with one continuous variable x_1 and one integer variable x_2. The first nonlinear constraint g_1 is nonconvex and g_2 is convex.

$$\begin{aligned}
\text{minimize} \quad & f(x_1, x_2) := 4x_1 - 15x_2 \\
\text{subject to} \quad & l_1(x_1, x_2) := -x_1 - 10x_2 \leq -6, \quad l_2(x_1, x_2) := x_1 - 10x_2 \leq 4, \\
& g_1(x_1, x_2) := 8.8x_1 - x_1^2 + 7x_2 + x_1x_2 - x_2^2 \leq 23.5, \\
& g_2(x_1, x_2) := -10x_1 + x_1^2 - 10x_2 + 2x_2^2 \leq -25.1, \\
& 2 \leq x_1 \leq 8, \quad x_2 \in \{0, 1, 2\}.
\end{aligned} \tag{4}$$

In the first iteration of the ESH algorithm, only the linear constraints are considered, *i.e.*, constraints g_1 and g_2 are ignored. The solution point to the MILP problem will be $x_1^* = (2, 2)$. Assuming that we have found the interior point $(5.99, 0.35)$ by minimizing the function

$$G(x_1, x_2) := \max\{g_1(x_1, x_2),\ g_2(x_1, x_2)\} \leq 0, \tag{5}$$

we can then perform a root search for a point on the boundary on the integer-relaxed nonlinear feasible set, *i.e.*, where $G(x_1, x_2) = 0$, to obtain the point $(5.69, 0.48)$. By generating a supporting hyperplane at this point for the constraint g_1, since g_2 is satisfied, we will get the following supporting hyperplane

$$\text{CUT}_1(x_1, x_2) := -2.10x_1 + 11.74x_2 + 6.39 \leq 0. \tag{6}$$

As can be seen from Fig. 1, adding this hyperplane, which is based on the nonconvex constraint, causes the MILP problem to immediately become infeasible as all integer solutions are cut off. Thus, the standard ESH algorithm could not find a primal solution even to this simple problem, as it cannot recover from an infeasible MILP subproblem.

SHOT includes much more functionality than the pure ESH algorithm, so it is possible that it will still find a valid integer-solution to problem (4). For example, the MIP solver can return more than one feasible solution in its so-called solution pool, and checking these candidates on the original MINLP problem may give an integer feasible solution. Also other primal heuristic strategies such as fixing integer variables to specific values and solving an NLP problem could work. In general, however, SHOT without the supplementary strategies discussed in this paper will not work well for nonconvex problems.

To reduce the probability of cutting away parts of the nonconvex feasible region, or more drastically creating an infeasible subproblem, we may want to generate as few cuts for nonconvex constraints as possible. It may also be a good idea to make the cuts less tight while still cutting away the previous solution point. Utilizing the ECP algorithm instead of the ESH algorithm, *i.e.*, generating cutting planes instead of supporting hyperplanes is a strategy to make the cuts less tight, thus reducing the probability of cutting away parts of the nonconvex feasible region. Since it is also problematic to find an interior point needed for the root search, ECP can in many cases be a better choice or the only option. However, since the ESH algorithm normally generates fewer and better cuts, it is very problem specific which of the algorithms to use for optimal performance.

As previously mentioned, SHOT first only adds cuts for the convex constraints. For most problems, it will however be required to eventually add cuts for nonconvex constraints as well. After this we cannot be sure that the lower bound obtained from the MIP solver is valid anymore for the nonconvex MINLP problem. Another issue is that we can end up with infeasible subproblems, even though the original MINLP problem is feasible. To handle this, SHOT will try to repair infeasible MIP problems by relaxing the cuts added, as described in Sect. 3.1. Also, if a primal solution has been found, SHOT will introduce an objective cut that forces the next solution to be better than the currently best known solution. If this causes the MIP problem to become infeasible, the same feasibility relaxation can be attempted. The objective cut is described in Sect. 3.2.

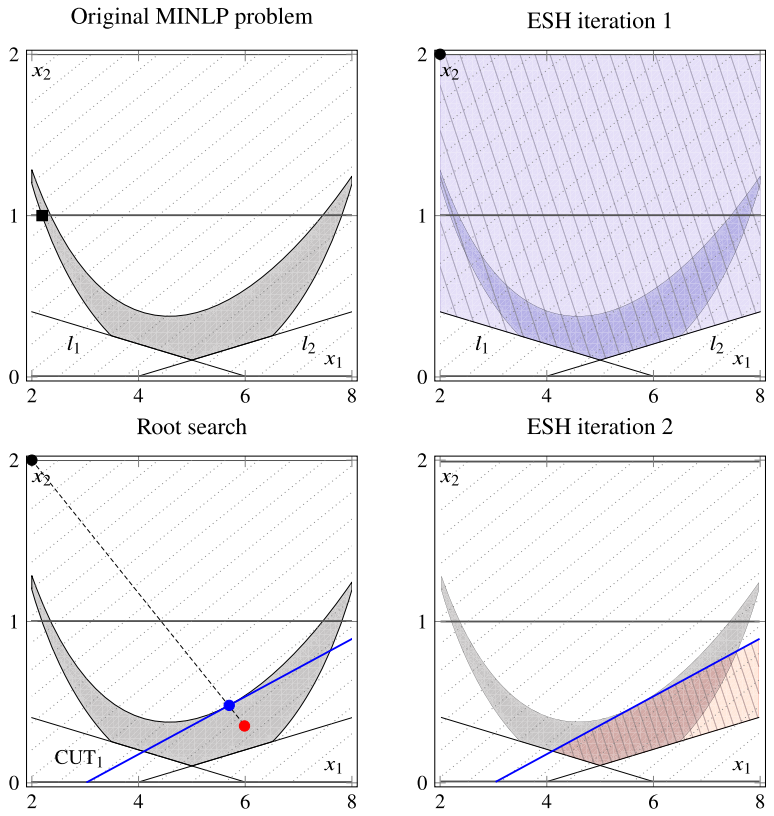

Fig. 1 The grey shaded area indicate the integer-relaxed feasible region of the MINLP problem. As can be seen, the only feasible solutions lie in two disjunct feasible regions on the line $x_2 = 1$, with the optimal solution being in (2.19, 1) (black square). When solving the first ESH iteration (the integer-relaxed feasible region is the blue area), we will get the solution point (2, 2) (black circle). When performing the root search between this point and the interior point (5.99, 0.35) (red circle) the point (5.69, 0.48) (blue circle) on the boundary is obtained. A supporting hyperplane CUT_1 (blue line) is then generated and added to the MILP problem in the second ESH iteration. Now, the MILP problem in iteration 2 is now no longer integer-feasible (the integer-relaxed feasible region is the red area) since all the feasible solutions have been cut off

3.1 Repairing infeasibilities in the dual strategies

The main issue with solving nonconvex problems with a PA strategy, is that feasible solutions are often sooner or later cut off when adding cuts to nonconvex constraints. The cuts might also make the linearized problem infeasible. Normally it is not possible to continue in this case, and we would need to terminate with the currently best known solution (if any).

A PA strategy can, however, be made more robust by performing an infeasibility relaxation, where the cuts added are relaxed to restore feasibility, after an infeasible subproblem has been detected. Assuming we have created an polyhedral approximation expressed using the constraints $\mathbf{Cx} + \mathbf{d} \leq \mathbf{0}$, we can easily solve the following MILP problem to find a feasibility relaxation:

$$\begin{aligned} \text{minimize} \quad & \mathbf{v}^T \mathbf{r} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{a}, \ \mathbf{B}\mathbf{x} = \mathbf{b}, \\ & \mathbf{C}\mathbf{x} + \mathbf{d} \leq \mathbf{r}, \\ & \underline{x}_i \leq x_i \leq \overline{x}_i \qquad \forall i \in I = \{1, 2, \ldots, n\}, \\ & x_i \in \mathbf{R}, \ x_j \in \mathbf{Z} \qquad \forall i, j \in I, \ i \neq j, \\ & \mathbf{r} \geq \mathbf{0}. \end{aligned} \tag{7}$$

Note that if the MIP solver supports quadratic terms, these can be included in the repair problem as well. Here, the solution vector $\mathbf{r}$ will contain the values required for restoring feasibility for the corresponding constraints. If r_k is fixed to be zero, k-th cut will not be allowed to be modified, *e.g.*, for cuts generated for convex constraints, which we know are valid. Penalizing the relaxation of individual constraints is done by assigning high values to the corresponding element of the positive vector of scalars $\mathbf{v}$. In SHOT, the strategy is to penalize the constraints added later more than those added earlier, by assigning the weight k, where k is an increasing counter for generated cuts. By favoring the modification of early added cuts, the risk of cycling, *i.e.*, when a cut recently added is directly relaxed, can be reduced. After the feasibility relaxation has been found, the constraints in the MIP problem are modified according to:

$$\mathbf{C}\mathbf{x} + \mathbf{d} \leq \mathbf{0} \quad \longrightarrow \quad \mathbf{C}\mathbf{x} + \mathbf{d} \leq \tau \mathbf{r}, \tag{8}$$

where $\tau \geq 1$ is a parameter to relax the model further. The MIP problem can now be solved, and additional cuts added to the linearization. If it was not possible to repair feasibility, SHOT will have to terminate with the currently best solution, as it is not possible to continue. This can, *e.g.*, happen if integer cuts have been added and the user has restricted SHOT to not try to relax these in the repair process. However, since the NLP solvers utilized are not global, we cannot guarantee that their returned solution is global and therefore, the integer cut may exclude a solution we have not found yet, *cf.*, Sect. 3.4. Another case when the repair step might fail is when a cutoff value below the best possible solution has been added, *cf.*, Sect. 3.2.

CPLEX and Gurobi have built in feasibility-repair functionality (utilizing the functions `feasopt` and `feasRelax` respectively), while Cbc currently lacks this functionality. Thus, if Cbc is used, SHOT restores feasibility by solving problem (7) directly.

This type of infeasibility relaxation has similarities to the strategy used in DICOPT, where the repair functionality is integrated into the MILP subproblem and considered in each iteration instead of a separate repair step taken when the subproblem becomes infeasible [72].

Example 2 We will now apply the repair functionality to the infeasible subproblem obtained in Ex. 1. The different steps are illustrated in Fig. 2. The infeasible problem (with added supporting hyperplane CUT_1) was

$$\begin{aligned} \text{minimize} \quad & x_1 - 10x_2 \\ \text{subject to} \quad & -x_1 - 10x_2 \leq -6, \quad x_1 - 10x_2 \leq 4, \\ & -2.10x_1 + 11.74x_2 \leq -6.39, \\ & 2 \leq x_1 \leq 8, \quad x_2 \in \{0, 1, 2\}. \end{aligned} \tag{9}$$

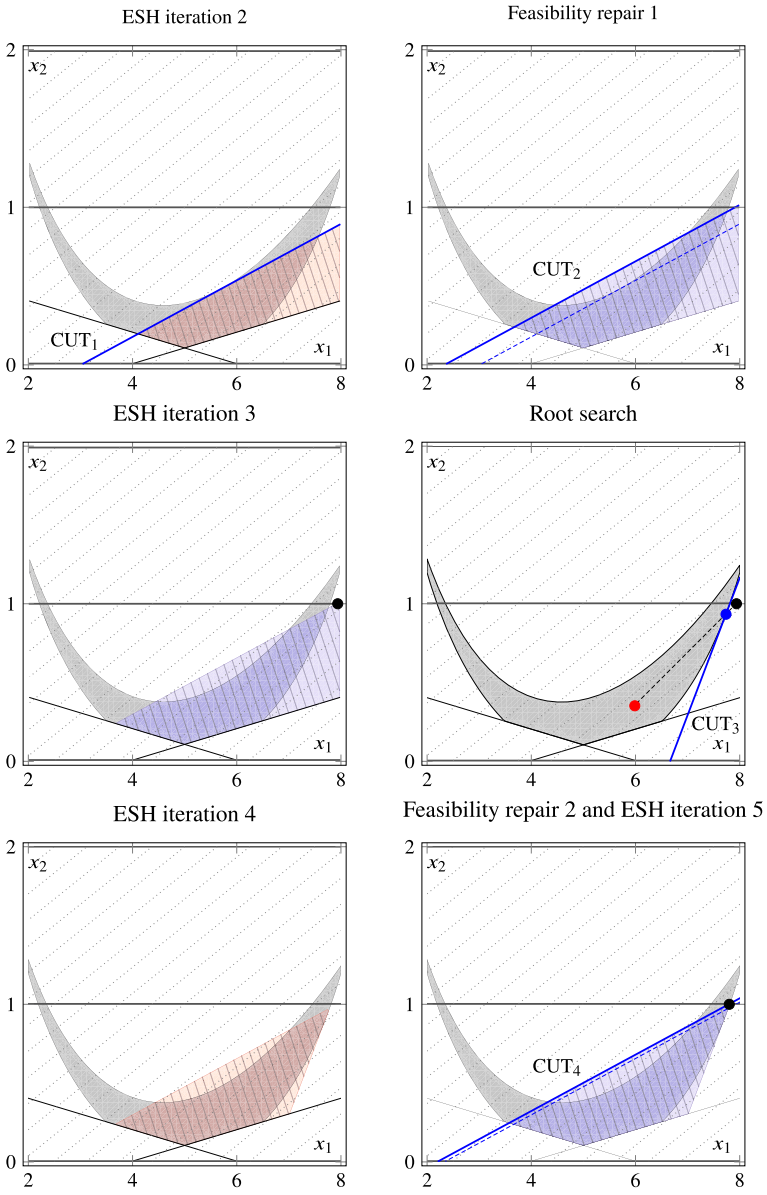

Fig. 2 The infeasibility repair functionality is utilized twice on the problem in Ex. 1 as described in Ex. 2

Now, we formulate and solve problem (7) with $v_1 = 1$

$$\begin{aligned} \text{minimize} \quad & r_1 \\ \text{subject to} \quad & -x_1 - 10x_2 \leq -6, \quad x_1 - 10x_2 \leq 4, \\ & -2.10x_1 + 11.74x_2 + 6.39 \leq r_1, \\ & 2 \leq x_1 \leq 8, \quad x_2 \in \{0, 1, 2\}, \quad r_1 \geq 0, \end{aligned} \tag{10}$$

which gives the solution $(x_1, x_2, r_1) = (8, 1, 1.30)$. The supporting hyperplane in problem (9) is relaxed by adding 1.43 to the RHS (here, we assume that the factor $\tau = 1.1$). Then the new cutting plane replacing CUT_1 will be

$$\text{CUT}_2(x_1, x_2) := -2.10x_1 + 11.74x_2 + 4.96 \leq 0. \tag{11}$$

When the updated MILP problem is solved, the solution is $(x_1, x_2) = (7.94, 1)$, with objective value 16.75. Note that this point is not a valid solution to problem (1), since the constraint g_2 is not fulfilled. However, we have a valid point outside the feasible region of the (integer-relaxed) MINLP problem, so we can perform a root search and generate a new constraint CUT_3:

$$\text{CUT}_3(x_1, x_2) := 5.47x1 - 6.27x2 - 36.50 \leq 0. \tag{12}$$

As can be seen from the figure, this supporting hyperplane cut is generated for the convex constraint g_2. Thus, the new cut does not cut away any feasible solutions. Adding this cut will however again give an integer-infeasible MILP problem, and we will need to restore feasibility by solving the following relaxation (with $v_1 = v_2 = 1$)

$$\begin{aligned} \text{minimize} \quad & r_1 + r_2 \\ \text{subject to} \quad & -x_1 - 10x_2 \leq -6, \quad x_1 - 10x_2 \leq 4, \\ & -2.10x_1 + 11.74x_2 + 6.39 \leq r_1, \\ & 5.47x_1 - 6.27x_2 - 36.50 \leq r_2, \\ & 2 \leq x_1 \leq 8, \quad x_2 \in \{0, 1, 2\}, \quad r_1 \geq 0, \; r_2 = 0, \end{aligned} \tag{13}$$

where the variable r_2 has been fixed to zero since the corresponding constraint was generated for a convex constraint. Now, we obtain the value $r_1 = 0.29$, and thus we replace the constraint CUT_2 with

$$\text{CUT}_4(x_1, x_2) := -2.10x_1 + 11.74x_2 + 4.67 \leq 0. \tag{14}$$

The resulting MILP problem is then

$$\begin{aligned} \text{minimize} \quad & x_1 - 10x_2 \\ \text{subject to} \quad & -x_1 - 10x_2 \leq -6, \quad x_1 - 10x_2 \leq 4, \\ & -2.10x_1 + 11.74x_2 + 4.67 \leq 0, \\ & 5.47x_1 - 6.27x_2 - 36.50 \leq 0, \\ & 2 \leq x_1 \leq 8, \quad x_2 \in \{0, 1, 2\}, \end{aligned} \tag{15}$$

which gives the solution 16.20 at the point $(x_1, x_2) = (7.80, 1)$. This is also a feasible solution to the original MINLP problem (1). By performing these simple repair steps, we have thus found a feasible solution to a problem we could not have solved otherwise with the ESH method. Note however, that this is still not the globally optimal solution mentioned in Ex. 1.

3.2 Utilizing a cutoff constraint to force new solutions and reduce the objective gap

Solving problem (1) as described in Exs. 1 and 2, shows that PA strategies in general, and the ESH algorithm specifically, may get stuck in suboptimal solutions for nonconvex problems. Normally, the PA methods then terminate with this suboptimal solution. However, as briefly described in [48], it is possible to try to force a better solution from the MIP problem when no progress can otherwise be made by introducing a so-called primal objective cut and then

resolving the MIP problem. This cut is of the form

$$\mathbf{c}^T \mathbf{x} \leq \gamma \cdot \text{PB}, \tag{16}$$

where γ must be selected so that $\gamma \cdot \text{PB} < \text{PB}$. Note that this cannot normally be accomplished by using the cutoff functionality in the MIP solvers, since the infeasibilities normally need to be explicitly present in the model as constraints for the solvers' built in infeasibility relaxation functionality to work.

In practice, whenever SHOT reaches an optimality gap of zero with cuts also created for nonconvex constraints, which would for a convex problem mean the global solution is found, it creates or modifies the objective cut in Eq. (16), so that its right-hand-side is less than the current primal bound. The problem is then resolved with the MIP solver. The problem will then either be infeasible (in which case the repair functionality discussed in Sect. 3.1 will try to repair the infeasibility), or a new solution with better objective value will be found. Note however, that this solution does not need to be a new primal solution to the MINLP problem, since it is not required to fulfill the nonlinear constraints, only their linearizations through hyperplane cuts that have been included in the MIP problem. This whole procedure is then repeated a user-defined number of times.

In the next example, and as illustrated in Fig. 3, the primal objective cut procedure in combination with the repair functionality is applied to the problem considered in Exs. 1 and 2.

Example 3 In Ex. 2, we were able to repair the MILP problem to get a feasible solution in $(x_1, x_2) = (7.80, 1)$ with the objective value 16.20. However, we know that this is not the global solution so we will try to find a better one by adding a primal objective cut that forces the objective to have a better (lower) value. We do this by introducing a cut

$$\text{CUT}_5(x_1, x_2) := x_1 - 10x_2 \leq 0.3 \cdot 16.20 = 4.86. \tag{17}$$

Note that the value 0.3 has been chosen here to reduce the numbers of iterations, and normally a γ-value less than but close to one should be used. As can be seen in Fig. 3, this makes the MILP problem infeasible again, and the constraint CUT_4 needs to be relaxed. The required feasibility relaxation can now be obtained by again formulating and solving problem (7). Note however, that the primal objective cut CUT_5 should not be relaxed, so its corresponding r-variable should be fixed to zero. By replacing constraint CUT_4 with the repaired constraint (with $\tau = 1.1$)

$$\text{CUT}_6(x_1, x_2) := -2.10x_1 + 11.74x_2 - 7.51 \leq 0, \tag{18}$$

the MIP problem again have a solution in the point $(2.0, 1)$. This point is, however, not feasible in the original MINLP problem, so we need to remove the point by adding a cut to the MILP problem in the next iteration. Now, the interior point is no longer feasible in the PA, so we instead add the cutting plane

$$\text{CUT}_7(x_1, x_2) := -5.98x_1 + 6.00x_2 + 19.06 \leq 0, \tag{19}$$

based on the convex constraint g_1. In iteration 7, the optimal solution $(x_1, x_2) = (2.18, 1)$ has now been found to a constraint tolerance of 0.03 and with an objective value of -6.25.

3.3 Verifying lower bounds for nonconvex problems

If feasibility can not be restored by modifying the supporting hyperplanes or cutting planes generated for the nonconvex constraints, the primal bound cannot be less the value $\gamma \cdot \text{PB}$

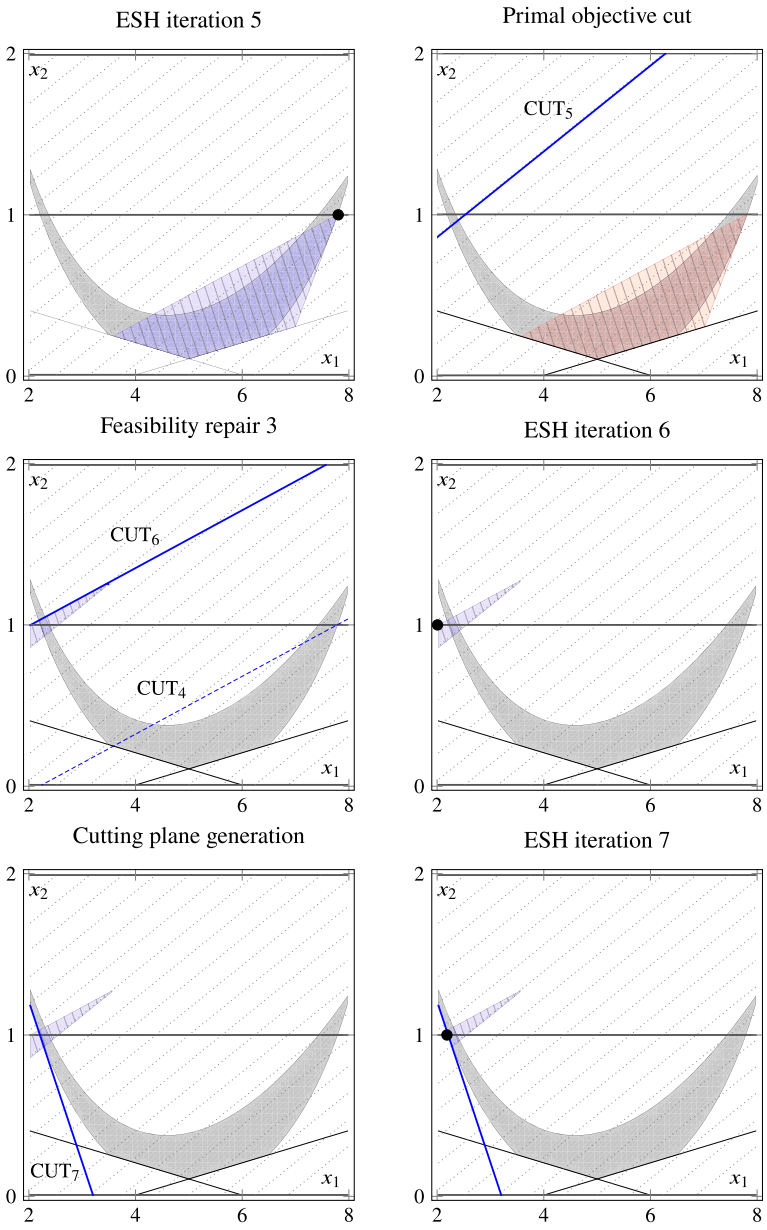

Fig. 3 A primal cut CUT_5, which makes the problem infeasible, is now introduced to the MILP problem in the first figure. The previously generated cut CUT_4 is therefore relaxed by utilizing the technique in Sect. 3.1 and replace with CUT_6 to allow the updated MILP problem to have a solution (2.0, 1). After adding a cutting plane CUT_7, a new primal solution, which is better than the previous primal solution (7.8, 1), is found in (2.18, 1)! After this, we can continue to generate more supporting hyperplanes to try to find an even better primal solution

and this value is thus a valid lower bound for the objective value for the nonconvex problem. What this means in practice is that the POA of the convex constraints has no solution giving a lower objective value than $\gamma \cdot \text{PB}$, and since all solutions to the nonconvex problems are contained in this polyhedral feasible set, no solution to the entire nonconvex problem can have a lower value than this value either.

Thus, the techniques in Sects. 3.1 and 3.2 can be combined to create a method for verifying a lower bound for problem (1). Assuming that we have generated cuts CUT_l with indices $l \in L_{\text{NC}}$ for nonconvex constraints out of all generated constraints indices in L. Then we generate the following MILP problem:

$$\begin{aligned}
\text{minimize} \quad & \mathbf{c}^T\mathbf{x} + \sum_{l \in L} r_l, \\
\text{subject to} \quad & \text{A}\mathbf{x} \leq \mathbf{a}, \ \text{B}\mathbf{x} = \mathbf{b}, \\
& \text{CUT}_l(\mathbf{x}) \leq r_l \qquad \forall l \in L, \\
& \mathbf{c}^T\mathbf{x} \leq \gamma \cdot \text{PB}, \\
& \underline{x}_i \leq x_i \leq \overline{x}_i \qquad \forall i \in I = \{1, 2, \ldots, n\}, \\
& r_l \geq 0 \qquad \forall l \in L_{\text{NC}}, \\
& r_l = 0 \qquad \forall l \in L \setminus L_{\text{NC}}, \\
& x_i \in \mathbf{R}, \ x_j \in \mathbf{Z}, \qquad \forall i, j \in I, \ i \neq j.
\end{aligned} \tag{20}$$

If this problem is infeasible, then we know that the nonconvex problem (1), where each nonlinear equality constraint $h(\mathbf{x}) = 0$ has been rewritten as the two constraints $-h(\mathbf{x}) \leq 0$ and $h(\mathbf{x}) \leq 0$, does not have a solution with lower objective value than $\tau \cdot \text{PB}$.

3.4 Adding integer cuts

In algorithms based on PA, integer cuts are often used to exclude a specific combination of integer or binary variable solutions. For example, in POA-based convex MINLP, this can be used to speed up the solution process since a specific integer combination will not be revisited in later iterations. In nonconvex PA-based methods integer cuts may be needed to force the MIP solver to visit other integer combinations. An integer cut is a constraint of the form

$$\|\mathbf{y} - \mathbf{y}^k\|_1 \geq 1, \tag{21}$$

where $\mathbf{y}$ corresponds to the elements of the vector $\mathbf{x}$ that are integer or binary variables. The constraint in Eq. (21) will then exclude the specific integer combination $\mathbf{y}^k$. In the case where all discrete variables are binaries, this expression simplifies to

$$\sum_{y_j^k = 0} y_j - \sum_{y_j^k = 1} (1 - y_j) \geq 1. \tag{22}$$

It is also possible to write the constraint in Eq. (21) in linear form in the more general case when one or more of the discrete variables are nonbinary; this is discussed further in [3].

For nonconvex MINLP, generating integer cuts in points provided by local NLP solvers is problematic due to the fact that we cannot be sure that the solution we have received when solving a fixed NLP problem for a specific integer combination is globally optimal unless a global solver has been used. The integer cut can, therefore, exclude the optimal integer assignment even if the optimal solution has not been obtained. Therefore, there is a setting

in SHOT that also allows us to relax added integer cuts when doing the feasibility relaxation in Sect. 3.1 in case the MIP subproblem becomes infeasible after adding integer cuts.

4 Utilizing reformulations in nonconvex MINLP

Reformulations may allow us to transform an optimization problem into a form that is more suitable for a specific method. This is normally accomplished by utilizing so-called lifting reformulations [43], where additional variables and constraints are introduced. The reformulations can be either exact or approximative. In the former case, the solution for the original problem can be easily obtained (*e.g.*, using a direct linear or nonlinear correspondence between variables) from the solution of the reformulated problem. In the latter case, the solution to the reformulated problem might not directly provide a valid solution in the original problem, but only *e.g.*, a valid lower bound; an example being piecewise linear approximations of nonlinear functions. As of version 1, SHOT only utilizes exact reformulations, but we plan to implement the αSGO algorithm [53] for lower bounding of bilinear [10,52], signomial [51] and general twice-differentiable [50] functions in a coming release.

The reformulations discussed in this section are introduced into the problem on which the MIP subproblems are based and hyperplane cuts generated for; the original problem will remain as is, and the validity of all primal solution candidates are verified on the original problem.

4.1 Handling nonlinear equality constraints

A special type of nonconvexity are nonlinear equality constraints. As long as the function h in a general nonconvex constraint $h(\mathbf{x}) = 0$ is not linear, *i.e.*, independently of whether it is convex, concave or nonconvex, the constraint will always (unless it is infeasible or its feasible region is a single point) give rise to a nonconvex feasible region. However, since nonlinear equality constraints are quite common, these need to be handled in some way by SHOT.

A possibility is to replace the constraint $h(\mathbf{x}) = 0$ with two separate constraints $h(\mathbf{x}) \leq 0$ and $-h(\mathbf{x}) \leq 0$. Then, if it is possible to deduce the convexity of $h(x)$ (in which case the first constraint is convex) or convexity of $-h(x)$ (in which case the second constraint is convex), hyperplane cuts can be added to the convex one without the risk of cutting away the optimal solution. Depending on whether the equality constraint is binding in one or both directions, it might however not be possible to tighten the objective gap without adding cuts to also the nonconvex constraint.

4.2 Reformulations for special terms

SHOT automatically performs reformulations for nonconvex terms, such as bilinear terms of at least one integer variable or monomials of binary variables, into linear form. Such reformulations can be written in many different ways, some options are discussed in [43,44]. If all nonconvex nonlinearities (that cannot be handled by the MIP solver) are removed utilizing this strategy, SHOT will manage to find the global solution to a nonconvex MINLP problem. Currently SHOT implements reformulations for the following terms:

– bilinear terms with at least one binary variable,

– bilinear terms with two integer variables, and
– monomials of binary variables.

The reformulations employed in SHOT are not new, for more information see, *e.g.*, for bilinear terms [17], [76] or [26], and for general multilinear terms, [58,66].

4.2.1 Reformulating bilinear terms with at least one binary variable

Bilinear terms with one or more binary variables, *i.e.*, a product $x_i x_j$ of a binary variable x_i and a continuous or discrete variable x_j, where $0 \leq \underline{\mathbf{x}}_j \leq x_j \leq \overline{\mathbf{x}}_j$, $x_i x_j$, can easily be reformulated to linear form by replacing the term with the auxiliary variable w_{ij} and introducing the linear constraints

$$\underline{\mathbf{x}}_j x_i \leq w_{ij} \leq \overline{\mathbf{x}}_j x_i, \quad w_{ij} \leq x_j + \overline{\mathbf{x}}_j (1 - x_i) \quad \text{and} \quad w_{ij} \geq x_j - \overline{\mathbf{x}}_j (1 - x_i).$$

Especially, if both variables are binaries, these expressions simplify to:

$$0 \leq w_{ij} \leq x_i, \quad w_{ij} \leq x_j - x_i + 1 \quad \text{and} \quad w_{ij} \geq x_i + x_j - 1.$$

Note that the variable w_{ij} will be reused in all terms where the bilinear term $x_i x_j$ occurs.

4.2.2 Reformulating bilinear terms of at least one discrete variable

A bilinear term $x_i x_j$ of one or more discrete variables with bounds $\underline{\mathbf{x}}_i \leq x_i \leq \overline{\mathbf{x}}_i$ and $\underline{\mathbf{x}}_j \leq x_j \leq \overline{\mathbf{x}}_j$ can be be exactly represented in linear form by replacing the term with an auxiliary variable w_{ij}. Note that the term is not allowed to change signs. If there is only one discrete variable in the product, we assume that it is x_i, if there are two discrete variables, we assume that x_i is the one with smaller domain, *i.e.*, $|\overline{\mathbf{x}}_i - \underline{\mathbf{x}}_i| \leq |\overline{\mathbf{x}}_j - \underline{\mathbf{x}}_j|$. Now binary variables b_k for the variable x_i is introduced, and constrained by

$$\sum_{k=\underline{\mathbf{x}}_i}^{\overline{\mathbf{x}}_i} b_k = 1, \qquad \text{and} \qquad x_i = \sum_{k=\underline{\mathbf{x}}_i}^{\overline{\mathbf{x}}_i} k \cdot b_k.$$

Using the additional variables, the value of w is then given as

$$k \cdot x_j - M(1 - b_k) \leq w \leq k \cdot x_j + M(1 - b_k), \qquad \forall k \in \{\underline{\mathbf{x}}_i, \ldots, \overline{\mathbf{x}}_i\},$$

where the values of $M = 2 \max\{|\underline{x}_i|, |\overline{x}_i|\} \max\{|\underline{x}_j|, |\overline{x}_j|\}$.

4.2.3 Reformulating monomials of binary variables

A monomial term of binary variables $b_1 \cdots b_N$ is either one (if all variables are one) or zero (otherwise). This nonlinear and nonconvex term can be replaced with an auxiliary variable w, where the relationship between w and b_i's is expressed as:

$$N \cdot w \leq \sum_{i=1}^{N} b_i \leq w + N - 1.$$

5 Utilizing nonconvex MIQCQP solver functionality

One of the main philosophies of SHOT has always been to fully utilize its subsolvers to extend the functionality and enhance the performance of the solver. SHOT utilized solving MIQP and MIQCQP subproblems iteratively as the first of the POA-based solvers. This enhanced the performance significantly when solving MINLP problems with a quadratic objective function. SHOT has also previously supported passing on convex quadratic constraints to its subsolvers instead of linearizing them with cutting planes or supporting hyperplanes. Currently CPLEX and Gurobi support nonconvex quadratic objective functions, and as of Gurobi version 9, there is support for nonconvex quadratic functions (*i.e.*, bilinear terms and concave quadratic terms) in the constraints as well. Naturally, SHOT can then automatically pass on these types of quadratic expressions to the subsolver if supported. For example, when considering Ex. 1, the convex quadratic constraint could directly be passed on to CPLEX and Gurobi, and not handled with the ESH linearization strategy. If Gurobi 9 was used as a subsolver, also the nonconvex quadratic constraint could be passed on directly. SHOT also has the option to extract nonconvex quadratic expressions from nonlinear expressions into quadratic equality constraints; this is beneficial since the nonlinear expressions might then become convex and the nonconvex equality can be directly handled by Gurobi. SHOT will then form a natural extension to the nonconvex functionality implemented in Gurobi, as it does not currently support general nonlinear constraints or objective functions. SHOT also exploits other types of reformulations and strategies, including NLP calls, that are not available in Gurobi.

Gurobi's strategy for solving nonconvex MIQCQP problems is mainly based on bounding the nonconvex terms with their McCormick envelopes in the subnodes of the branching tree. This is something that can be accomplished also by utilizing lazy constraint callback functionality of solvers not supporting nonconvex MIQCQP problems [19]. This has not been implemented yet in SHOT (as of version 1.0), but might be added to future releases.

6 Nonconvex MINLP benchmark

Results from benchmarking SHOT's new nonconvex functionality with different subsolvers, and comparing the results to those of other MINLP solvers, are presented in this section. The main benchmark set consists of problems taken from MINLPLib [57] that fulfill the following conditions:

- are classified as nonconvex,
- have other nonlinearities than a quadratic objective function (*i.e.*, are not pure MIQP problems),
- have at least one discrete, *i.e.*, binary or integer, variable, and
- have primal and dual bounds with an objective gap less than 0.1 in MINLPLib.

When applying these filters to the total problem library consisting of 1704 problems the result is 326 problems. A full list of these are given in Appendix A. Note that a large part (182) of these problems are actually MIQCQP instances. All the solver logs as well as the reports generated by PAVER [8], which was used for performing the benchmark analysis, are available online on the web site https://andreaslundell.github.io/minlpbenchmarks.

SHOT 1.0 was used in the comparisons. All the available MIP solvers in SHOT were considered separately, namely Cbc 2.10 (with IPOPT as NLP solver), as well as CPLEX 12.10 and Gurobi 9 (with CONOPT as NLP solver). Hence, both a completely free version,

as well as options where commercial subsolvers are used, are included in the benchmarks. Note that both CPLEX and Gurobi offer free academic licenses which can be used with SHOT as well.

Both local (AlphaECP, DICOPT, SBB, BONMIN) and global (Antigone, BARON, Couenne, SCIP, LINDOGlobal) MINLP solvers, as well as one global MIQCQP solver (Gurobi) have been tested. It should be mentioned that comparing the local solvers with the global solvers is not completely fair since the global solvers have more functionality for handling nonconvex problems, and the local solvers are more tailored towards convex problems. However, we have included both local and global solvers to give a better overview of the current state of nonconvex MINLP. Also, since SHOT share similarities with both local and global solvers, it is interesting to see how SHOT compares.

The solvers were called through GAMS 31.2. Most subsolvers rely on LP, MILP and NLP subsolvers, with the defaults selected as CPLEX and CONOPT. However, for BONMIN and Couenne the recommended solvers Cbc and IPOPT were used. Also, SCIP only supports IPOPT as its NLP solver in GAMS. For BONMIN, the recommended nonconvex BB strategy was used. In these comparisons, we have tried to use the default values for solver settings unless there is a specific reason to do otherwise, such as the solver clearly terminating prematurely due to a low iteration limit. Nondefault settings are listed in Appendix B. The absolute and relative gap termination limits used were both set to 0.1%. However, since BARON uses a different measure of the relative gap than the others, a slightly larger relative value was used in PAVER (0.102%) to make up for this fact. The time limit for all solvers were selected to be 900 s, and if the solver did not exit within 920 seconds, the run was considered as failed in PAVER. In the solution profiles shown in this section, we have also included the so-called virtual best and worst solvers, where the virtual best solver selects the most efficient solver for each problem instance based on the results, and the virtual worst the least efficient one. This gives a good overview of the difficulty of the test set.

The local solvers AlphaECP, DICOPT, SBB and BONMIN only guarantee to find the global solutions for convex instances, so for nonconvex problems they can only provide a primal solution; note however, that in GAMS, these will still return a lower bound for the objective value, but this bound is not in general valid. As SHOT can detect convexity on the function level, both the lower and upper bounds on the objective are valid (unless its convex strategy is forced), and thus it acts more like a global (nonconvex) solver, but with no theoretical guarantee of finding the global optimal solution as the other global solvers considered here.

The set of benchmark problems described above is by no means balanced, as we have not excluded any instances and there are several problems with similar structure. Therefore, we have also considered the instances in the Mittelmann MINLP benchmark problem collection [60] fulfilling the criteria mentioned above. If there were instances with similar names, we chose either the larger instance as indicated by the name or the last one alphabetically. The results from the 16 nonconvex problems for SHOT (convex and nonconvex strategies) and the global solvers BARON and SCIP are available in in Table 1. Note that the comparisons on the smaller benchmark set was with a time limit of 1800 seconds.

6.1 Comparing the convex and nonconvex strategies in SHOT

The impact of the new nonconvex strategy in SHOT, based on the ideas presented in this paper, is discussed in this section. Table 1 clearly shows that the improvements increase SHOT's ability to solve nonconvex problems, and the nonconvex strategy obtained a primal

Table 1 The table shows relative dual (DG) and primal (PG) gaps (in %) as well as the solution times (in seconds) for the global solvers BARON and SCIP, as well as SHOT with the default nonconvex strategy. Also included is SHOT with its default convex strategy that disables most nonconvex functionality mentioned in this paper, *i.e.*, it assumes the problem is convex. SHOT (convex) does return a dual bound, but it cannot be trusted for nonconvex problems and therefore the DG is not included. Since SHOT (nonconvex) is able to reformulate some of the instances into convex form using the reformulations in Sect. 4 and records the last valid dual bound, the dual bounds are valid and SHOT could solve three of the problems to global optimality. An '∞' in the GAP columns indicates that the respective bound was not found. An '–' in the time column means that no result were returned, indicating, *e.g.*, a solver crash, and TL means that the given time limit of 1800 s was reached. The relative objective gap termination criteria used was 0.1%. The results were analyzed with PAVER [8]

	BARON			SCIP			SHOT (nonconvex)			SHOT (convex)	
Instance	DG	PG	Time	DG	PG	Time	DG	PG	Time	PG	Time
autocorr_bern25-13	0	0	1532	43	0	TL	143	0	TL	11	11
cecil_13	0	0	40	0	0	1275	250	6.7	44	7.9	2.6
crudeoil_lee3_10	∞	∞	–	0	0	246	2.3	0.3	33	∞	–
genpooling_meyer04	42.0	12	TL	78	106	TL	957	127	0.5	∞	–
multiplants_mtg1a	∞	0	576	0	0	887	∞	409	0.8	∞	0.8
oil	35.0	0	TL	23	0	TL	247	9.5	4.1	∞	2.8
pooling_epa2	1.8	0	TL	88	7.1	TL	∞	∞	–	∞	0.6
radar-3000-10-a-8_lat_7	> 1000	172	TL	167	1.7	TL	> 1000	> 1000	119	∞	9.9
sfacloc2_4_80	0	0	103	0	0	143	∞	4.8	12	35	2.2
smallinvSNPr3b050-055	0	0	5	0	0	456	0	0	8.1	0	7.7
sonet20v6	0	0	178	0	0	785	0	0	344	0	56
sporttournament20	0	0	24	0	0	17	0	0	4.1	0	3.4
squfl030-150persp	8.4	0.3	TL	0.2	0	TL	> 1000	0	TL	∞	9.4
sssd20-04persp	72.0	0.2	TL	16	0	TL	> 1000	9.5	1047	∞	–
tln7	0	0	1170	390	2	TL	∞	0	TL	∞	–
wastepaper4	11	19	91	15	12	126	∞	189	4.7	∞	0.2

Table 2 Details on the impact of some of the nonconvex enhancements utilized by SHOT's nonconvex strategy in Table 1. The problem type before and after reformulation is indicated, as well as the number of times the feasibility repair and objective cut steps were invoked and were successful. The maximum number of objective cut steps performed is by default five in SHOT, and for these instances, none of them helped find a better primal solution

	Problem type		Feasibility repair step		Primal cut step (%)	
Instance	before ref.	after ref.	performed	successful	performed	successful
autocorr_bern25-13	MINLP	MILP				
cecil_13	MINLP	MINLP	4	1	5	0
crudeoil_lee3_10	MIQCQP	MINLP	4			
genpooling_meyer04	MIQCQP	MINLP	4	1	5	0
multiplants_mtg1a	MINLP	MINLP	2		5	0
oil	MINLP	MINLP	3	1	5	0
pooling_epa2	MINLP	MINLP				
radar-3000-10-a-8_lat_7	MIQCQP	MINLP			5	0
sfacloc2_4_80	MINLP	MINLP	5	2	5	0
smallinvSNPr3b050-055	MIQCQP	MIQCQP				
sonet20v6	MIQCQP	MILP				
sporttournament20	MIQCQP	MILP				
squfl030-150persp	MIQCQP	MINLP			5	0
sssd20-04persp	MIQCQP	MINLP	82	80	5	0
tln7	MIQCQP	MILP				
wastepaper4	MINLP	MINLP	3	1	5	0

Table 3 In the upper half of the table, the impact of the reformulation step is indicated by showing the distribution of the resulting problem types after reformulation. This indicates the differences between the subsolver with regard to what type of subproblem they support. For example Cbc only handles linear subproblems so the reformulated problems are either MILP or MINLP. CPLEX and Gurobi both support nonconvex quadratic objectives. CPLEX supports only convex quadratic constraints, while Gurobi also allows nonconvex constraints. Thus, for CPLEX the reformulation step rewrites all nonconvex quadratic constraints as linear if possible, or otherwise general nonlinear. In the lower half of the table, the success rates of the infeasibility repair and objective cuts are shown. A success is registered if a better primal solution has been found for a specific problem after the step was performed

	Number of problem instances		
	SHOT+Cbc	SHOT+CPLEX	SHOT+Gurobi
Problem type after reformulation			
MILP	63	64	49
Convex MIQCQP	0	29	29
Nonconvex MIQCQP	0	0	116
Convex MINLP	29	0	0
Nonconvex MINLP	232	233	139
Crash before ref.	3	1	3
Successful infeasibility repairs	48	67	19
Successful primal cuts	21	8	1

solution in 15 of the 16 problems compared to six for the convex strategy. The nonconvex strategy also allowed SHOT to solve three of the problems to global optimality, which is not possible in the convex strategy. In Table 2, more information about the impact of some of the nonconvex enhancements in SHOT is detailed for this smaller benchmark set. Out of the 16 problems, SHOT was able to transform four into MILP problems. The number of times per problem the feasibility repair and objective cut strategies successfully enabled the search for a better primal solution when considering the smaller benchmark set is also shown in Table 2. In this comparison CPLEX was used as MIP solver, and convex quadratic constraints were passed on to the MIP solver.

The impact of the reformulations on the larger benchmark set is shown in Table 3. Since the resulting reformulations depend on what type of subproblem the MIP solver allows, statistics are shown for each of the three MIP solvers supported. It is clear that Gurobi (which supports nonconvex quadratic constraints) can handle a large number of the problems directly without any reformulations. For Cbc and CPLEX these problems must either be reformulated into MILP form or handled as nonconvex nonlinear constraints or objectives. There are also statistics on the number of problems where the infeasibility repair and primal cut strategies have had an impact in Table 3. While the repair step seems to be useful in many of the problem instances, the impact of the primal cut is not as clear. There are, however, certain aspects of the repair and objective cut strategies that could still be improved; for example, a smarter way of prioritizing which constraints to modify during the repair step, and how to select the reduction parameter γ in Eq. (16).

To further illustrate the impact of the nonconvex enhancements, the repair step in Sec. 3.1 and the primal cut in Sec. 3.2 were disabled, as were the reformulations in Sec. 4. The number of solved instances with relative primal gaps (deviation from known optimal value) and relative objective gaps (difference between upper and lower bound on the objective) of 0.1%,1% and 10% are shown in Table 4. This is a further indication that the strategies

Table 4 The impact of disabling the nonconvex functionality has on SHOT when considering the larger benchmark set of 326 problems. The time limit was set at 900 seconds, and CPLEX was used as MIP solver in SHOT

Enabled functionality		Solved instances with relative objective gap (%)			Solved instances with relative primal gap (%)		
Reformulations	red.cut + repair	≤ 0.1	≤ 1	≤ 10	≤ 0.1	≤ 1	≤ 10
✓	✓	92	92	119	128	136	144
✓	×	90	90	113	116	123	132
×	✓	48	48	73	86	94	114
×	×	45	45	66	73	80	98

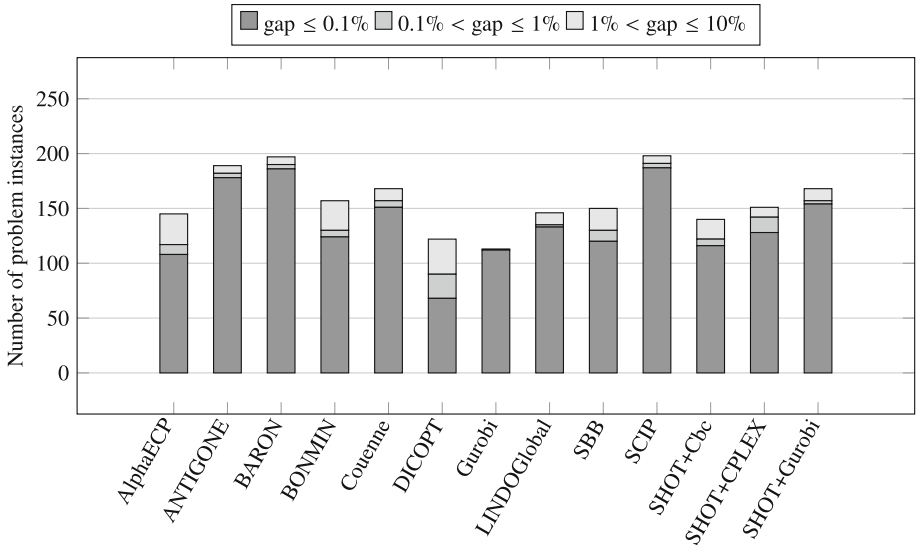

Fig. 4 The number of instances in the benchmark where the solvers returned a primal solution within 0.1%, 1% and 10% (as calculated by PAVER) of the best known objective value, and within a time limit of 900 s. Note that Gurobi can only solve MIQCQP problems, in total 182 out of the 326 problems

presented in this paper has had a significant impact on the number of problems that could be solved by SHOT. As can be expected, the reformulations have the largest impact when considering the objective gap. This is natural since the repair and reduction cut is not used before a linearization of a nonconvex constraint has been introduced, which in the process excludes further dual bound improvements. When considering the primal gap, the reduction cut and repair steps have more impact, but the reformulations are still important here; one reason is that the primal heuristic of solving a fixed-integer MINLP, *i.e.*, an NLP problem, does not currently utilize the reformulated problem, but rather its original form.

6.2 Efficiency of finding primal solutions

In Fig. 4, it is shown how many of the problems are solved to relative primal gap of 1% and 10% in addition to the termination gap of 0.1%. For some of the solvers the differences

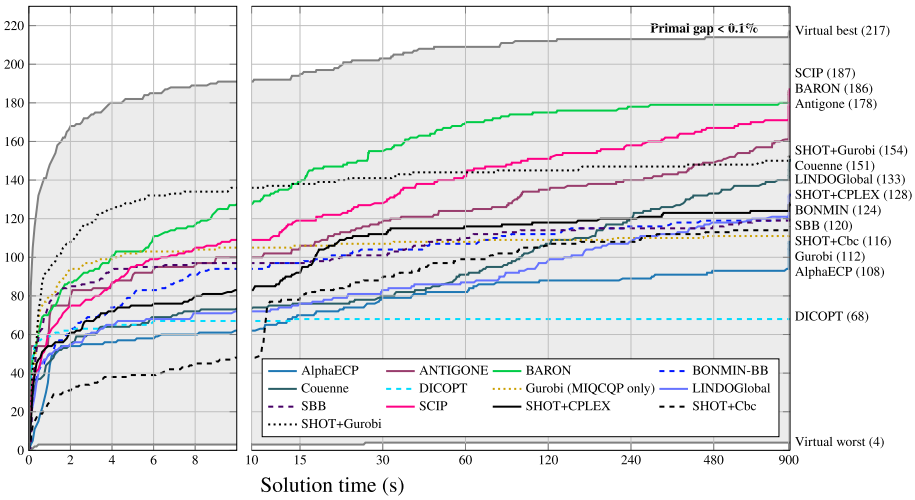

Fig. 5 The solution profile indicates the number of solved MINLP instances as a function of time. A problem is regarded as solved if the primal gap, as calculated by PAVER [8], is $\leq 0.1\%$. For example, at 10 second AlphaECP reaches 61 problems, which means that the solver is able to solve 61 of the problems by spending less than 10 seconds on each problem. The solution times used are the ones where the solver has terminated, not necessarily when the primal solution has been found. This explains the jump at 900 seconds for many of the solvers, where they return the current best known solution. Note that the time axis up until 10 s is linear and after that logarithmic. The grey shaded region indicates the difference between the virtual best (top) and worst (bottom) solver

are small, *i.e.*, either the solvers manages to find a good solution or they do not find one within a 10% gap. However, especially for the local solvers there is a significant difference, and for some of the problems they are struggling to find a close to optimal solution. The number of instances each solver found a primal solution to (within a relative gap of 0.1% of the best known solution) as a function of time is shown as a solution profile in Fig. 5. Note that this does not indicate at which exact time each solver has actually found the solution, only when it has terminated with said solution, a fact considered in an older benchmark in [41]. From the figure, it can be seen that the polyhedral approximation based local solvers AlphaECP, DICOPT and SHOT are quite good at quickly finding primal solutions, as are the BB-based local solvers BONMIN and SBB. It is clear, however, that the global solvers Antigone, BARON, Couenne and SCIP are the most efficient at finding the correct primal solution when regarding the total time limit. We can also assume that their progress will continue, albeit at a slower rate, if the time limit was increased, which may not be the case of the local solvers. Gurobi also is very efficient when considering that it only supports a little over half of the total number of problems!

When considering the performance of SHOT, we can conclude that the performance with Cbc as MIP solver is the least efficient, but this can be expected as the same is true for convex problems as well [49]. However, even with Cbc and IPOPT as subsolvers, SHOT clearly finds more primal solutions than the other PA-based solvers, DICOPT and AlphaECP, even if they have more efficient subsolvers (CPLEX and CONOPT in this case). This clearly indicate that the additional techniques for nonconvex problems described in this paper are beneficial. When considering SHOT with CPLEX and CONOPT, the performance is clearly better. This has not only to do with CPLEX in general being more efficient than Cbc, but also due to the fact that CPLEX can handle convex or nonconvex quadratic objective functions, and convex

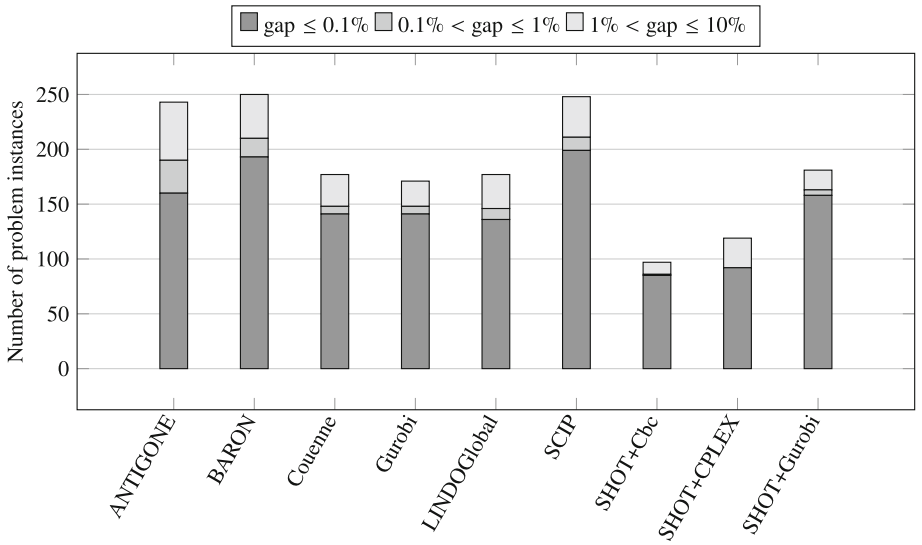

Fig. 6 The number of instances in the benchmark where the solvers managed to reach an objective gap of 0.1%, 1% and 10%. Only the solvers that give a valid lower bound on the optimal solution, *i.e.*, the global solvers and SHOT, are included. Note that Gurobi can only solve MIQCQP problems, in total 182 out of the 326 problems

quadratic constraints. Thus, the subproblems solved in SHOT may also be of the MIQP- or MIQCQP-types in CPLEX. Since more than half of the benchmark problems are actually nonconvex MIQCQP problems, the performance boost of utilizing Gurobi as a subsolver in SHOT is significant, and this has more to do with Gurobi than with SHOT. However, when comparing directly with Gurobi, SHOT of course solves more problems since it supports significantly more of the problems in the benchmark set.

6.3 Efficiency in proving optimality

The global MINLP solvers Antigone, BARON, Couenne, LINDOGlobal, and SCIP also provide a valid lower bound on the optimal solution to nonconvex problems; this is also true for Gurobi for the problems it supports, *i.e.*, MIQCQP problems. SHOT also provides a valid lower bound as described in Sect. 3, however in contrast to the global solvers, there is no theoretical guarantee that the gap can be reduced to the globally optimal solution for nonconvex problems.

In Fig. 6, the number of instances solved to objective gaps of 0.1%, 1% and 10% are shown. From this figure it can be deduced that many of the unsolved problems for a solver are far from being solved within 900 s since the gap is larger than 1% in most of the remaining ones.

A solution profile of the number of problems solved to an relative gap of 0.1% is shown in Fig.7. It is clear that SCIP and BARON are in a league of their own, and manage to find the global optimal solution to 195 and 193 of the problems respectively. Antigone and SHOT with Gurobi manage to solve 160 and 158 respectively. Gurobi solves 141 of the 182 MIQCQP instances, which means that SHOT can verify the global solution of about an additional 17 MINLP problems. The performance of SHOT with CPLEX and Cbc are significantly worse due to the fact that they do not support nonconvex quadratic constraints

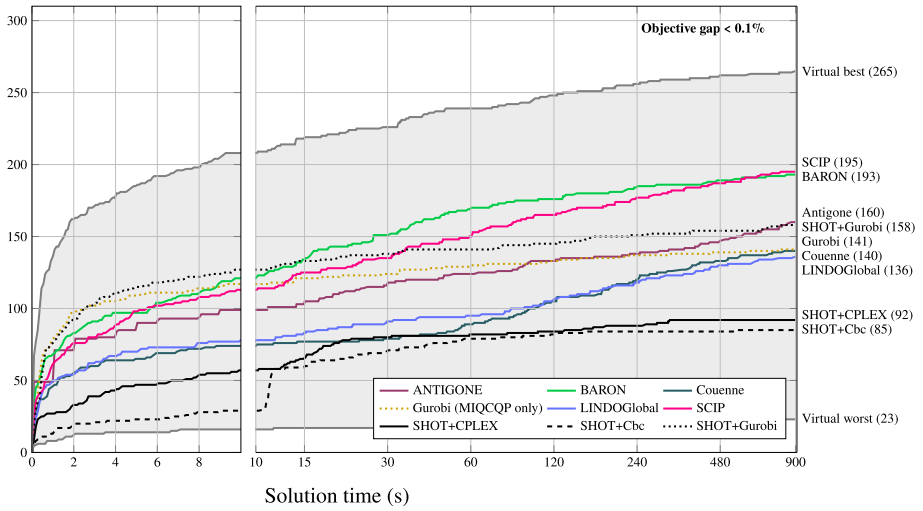

Fig. 7 The solution profile indicates the number of solved MINLP instances as a function of time. A problem is regarded as solved if the relative objective gap, as calculated by PAVER [8], is $\leq 0.1\%$. Note that the time axis up until 10 s is linear and after that logarithmic. The grey shaded region indicates the difference between the virtual best (top) and worst (bottom) solver

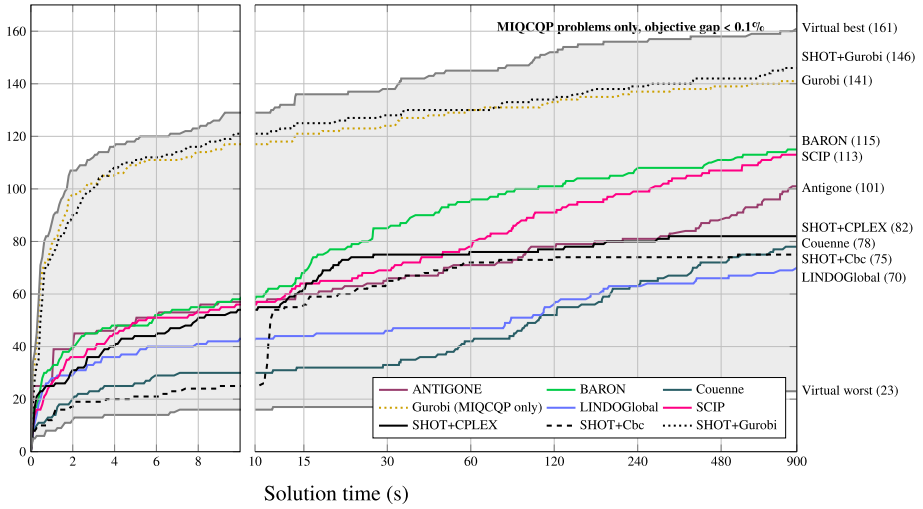

Fig. 8 The solution profile indicates the number of solved instances out of the total 182 MIQCQP problems as a function of time. A problem is regarded as solved if the relative objective gap, as calculated by PAVER [8], is $\leq 0.1\%$. Note that the time axis up until 10 s is linear and after that logarithmic. The grey shaded region indicates the difference between the virtual best (top) and worst (bottom) solver

(CPLEX) or any quadratic terms at all (Cbc), respectively. Finally in Fig. 8 only the pure nonconvex MIQCQP instances are considered. For these instances, it is clear that Gurobi is currently the most efficient solver for this problem type. The difference between SHOT with Gurobi and Gurobi might be due to the fact that SHOT also solves NLP problems in its primal strategy which might help to tighten the objective gap. It also modifies a few of Gurobi's default settings.

7 Conclusions

In this paper, we have described some new features added to the SHOT solver. As shown in this paper, these features significantly increase SHOT's capability of solving nonconvex MINLP problems. The main issues with utilizing PA-based techniques for nonconvex MINLP are *(i)* that valid solutions are likely to be cut off by the constraints generated as the approximation of the nonlinearities in the problem is tightened or that the solution procedure is terminated too early, and *(ii)* that the lower bounds given by the best possible solution in the outer approximation can no longer be trusted. In this paper, the former has been addressed by introducing a repair step for infeasible MIP subproblems and a primal objective cut that forces the search for better primal solutions. The latter issue is partly handled by utilizing lifting reformulations to convexify certain nonconvex functions and by exploiting convexity detection on the nonlinear functions in the problem. The convexity detection is, for example, used to determine if and when a cut has been added that invalidates the lower bound, in which case the last valid bound is returned when terminating the solver. This approach does not generally guarantee to completely close the objective gap, but may in some cases actually allow us to reduce the objective gap to zero. It does however often provide some bounds on the objective other local solvers can not. The type of MIP subsolver used has a large impact on how well SHOT can solve nonconvex problems; especially in combination with Gurobi version 9 and later, SHOT becomes a much more general and powerful tool for solving general MINLP problems, since it can utilize Gurobi's nonconvex MIQCQP functionality. Even if SHOT's performance is somewhat reduced with Cbc as a subsolver, together with IPOPT the combination forms a completely free open-source source solver with a performance that is on pair with the commercial local solvers. In this paper, SHOT is compared to other local and global solvers on a benchmark set of 326 MINLP problems, and the results are promising. It is clear that the new features greatly improve SHOT's ability to solve the nonconvex test problems, and overall the solver performs well compared to the other local solvers. The improvements discussed in this paper are, however, only the first steps in SHOT's nonconvex development, and we plan to introduce lifting reformulations for signomials [51] and general twice-differentiable [50] functions in coming versions; these will significantly extend SHOT's nonconvex capabilities further.

Acknowledgements AL acknowledges support from the Magnus Ehrnrooth Foundation. JK acknowledges support from the Newton International Fellowship by the Royal Society (NIF\R1\182194) and the Swedish Cultural Foundation in Finland

Funding Open access funding provided by Abo Akademi University (ABO).

Appendix A

The problems considered in the benchmark were selected using the conditions specified in Section 6. The result is the 326 problems listed here.

The following 144 problems are general MINLP problems, *i.e.*, they have at least one nonlinearity that is not quadratic:

4stufen, autocorr_bern20-05, autocorr_bern20-10, autocorr_bern20-15, autocorr_bern25-06, autocorr_bern25-13, autocorr_bern30-04, autocorr_bern30-08, autocorr_bern35-04, autocorr_bern40-05, batch0812_nc, batch_nc, bchoco05, bchoco06, bchoco07, casctanks, cecil_13, chp_shorttermplan1a, chp_shorttermplan1b, chp_shorttermplan2a, chp_shorttermplan2b, chp_shorttermplan2d, csched1a, csched1, eniplac, ethanolh, ethanolm, ex1221, ex1222, ex1224, ex1225, ex1226, ex1233, ex1243, ex1244, ex1252, ex1252a, ex3pb, fin2bb, gasnet_al1, gasnet_al2, gasnet_al3, gasnet_al4, gasnet_al5, gastrans, gastrans040, gastrans135, gastrans582_cold13, gastrans582_cold13_95, gastrans582_cold17, gastrans582_cold17_95, gastrans582_cool12, gastrans582_cool12_95, gastrans582_cool14, gastrans582_cool14_95, gastrans582_freezing30, gastrans582_freezing30_95, gastrans582_mild10, gastrans582_mild10_95, gastrans582_mild11, gastrans582_mild11_95, gastrans582_warm15, gastrans582_warm15_95, gastrans582_warm31, gastrans582_warm31_95, gear4, ghg_1veh, ghg_2veh, gkocis, hadamard_4, hadamard_5, hda, heatexch_gen2, heatexch_spec1, heatexch_spec2, heatexch_spec3, heatexch_trigen, hmittelman, hybriddynamic_var, johnall, kport20, lip, milinfract, minlphix, multiplants_mtg1a, multiplants_mtg2, multiplants_mtg5, multiplants_stg5, nvs01, nvs05, nvs16, nvs21, nvs22, oaer, oil, oil2, ortez, parallel, pooling_epa1, pooling_epa2, pooling_epa3, procsel, sepasequ_convent, sfacloc2_2_80, sfacloc2_2_90, sfacloc2_2_95, sfacloc2_3_80, sfacloc2_3_90, sfacloc2_3_95, sfacloc2_4_80, sfacloc2_4_90, sfacloc2_4_95, spring, st_e15, st_e29, st_e32, st_e35, st_e36, st_e38, st_e40, super1, supplychainp1_020306, supplychainp1_022020, supplychainp1_030510, supplychainr1_020306, supplychainr1_022020, supplychainr1_030510, supplychainr1_053050, synheat, tanksize, tspn05, tspn08, tspn15, unitcommit2, wager, wastepaper3, wastepaper4, waternd1, waternd2, waterno2_01, waterno2_02, waterno2_03, watertreatnd_conc, watertreatnd_flow.

The following 182 problems can in general be classified as MIQCQP problems:

autocorr_bern20-03, autocorr_bern25-03, blend029, blend146, blend480, blend531, blend721, blend852, carton7, crudeoil_lee1_05, crudeoil_lee1_06, crudeoil_lee1_07, crudeoil_lee1_08, crudeoil_lee1_09, crudeoil_lee1_10, crudeoil_lee2_05, crudeoil_lee2_06, crudeoil_lee2_07, crudeoil_lee2_08, crudeoil_lee2_09, crudeoil_lee2_10, crudeoil_lee3_05, crudeoil_lee3_06, crudeoil_lee3_07, crudeoil_lee3_08, crudeoil_lee3_09, crudeoil_lee3_10, crudeoil_lee4_05, crudeoil_lee4_06, crudeoil_lee4_07, crudeoil_lee4_08, crudeoil_lee4_09, crudeoil_lee4_10, crudeoil_li01, crudeoil_li02, crudeoil_li03, crudeoil_li05, crudeoil_li06, crudeoil_li11, crudeoil_li21, crudeoil_pooling_ct2, crudeoil_pooling_ct4, crudeoil_pooling_dt1, crudeoil_pooling_dt4, edgecross10-010, edgecross10-020, edgecross10-030, edgecross10-040, edgecross10-050, edgecross10-060, edgecross10-070, edgecross10-080, edgecross10-090, edgecross14-019, edgecross14-039, edgecross14-058, edgecross14-078, edgecross14-137, edgecross14-156, edgecross14-176, edgecross20-040, edgecross22-048, elf, ex1263, ex1263a, ex1264, ex1264a, ex1265, ex1265a, ex1266, ex1266a, feedtray2, gabriel01, gabriel04, gasprod_sarawak01, gasprod_sarawak16, gasprod_sarawak81, genpooling_lee1,

genpooling_lee2, genpooling_meyer04, hydroenergy1, hydroenergy2, hydroenergy3, portfol_robust050_34, portfol_robust100_09, portfol_robust200_03, portfol_shortfall050_68, portfol_shortfall100_04, portfol_shortfall200_05, prob02, prob03, product, product2, radar-3000-10-a-8_lat_7, ringpack_10_1, ringpack_10_2, sep1, sjup2, smallinvSNPr1b020-022, smallinvSNPr1b100-110, smallinvSNPr1b150-165, smallinvSNPr1b200-220, smallinvSNPr2b010-011, smallinvSNPr2b020-022, smallinvSNPr2b050-055, smallinvSNPr2b100-110, smallinvSNPr2b150-165, smallinvSNPr2b200-220, smallinvSNPr3b010-011, smallinvSNPr3b020-022, smallinvSNPr3b050-055, smallinvSNPr3b100-110, smallinvSNPr3b150-165, smallinvSNPr3b200-220, smallinvSNPr4b010-011, smallinvSNPr4b020-022, smallinvSNPr4b050-055, smallinvSNPr4b100-110, smallinvSNPr4b150-165, smallinvSNPr4b200-220, smallinvSNPr5b010-011, smallinvSNPr5b020-022, smallinvSNPr5b050-055, smallinvSNPr5b100-110, smallinvSNPr5b150-165, smallinvSNPr5b200-220, sonet17v4, sonet18v6, sonet20v6, sonet21v6, sonet22v4, sonet24v2, sonet25v5, spectra2, sporttournament06, sporttournament08, sporttournament10, sporttournament12, sporttournament14, sporttournament16, sporttournament18, sporttournament20, sporttournament22, sporttournament24, sporttournament26, sporttournament28, squfl010-025persp, squfl010-040persp, squfl010-080persp, squfl015-060persp, squfl015-080persp, squfl020-040persp, squfl020-050persp, squfl020-150persp, squfl025-025persp, squfl025-030persp, squfl025-040persp, squfl030-100persp, squfl030-150persp, squfl040-080persp, sssd08-04persp, sssd12-05persp, sssd15-04persp, sssd20-04persp, sssd25-04persp, st_e13, st_e31, supplychain, telecomsp_njlata, telecomsp_pacbell, tln2, tln4, tln5, tln6, tln7, tln12, tloss, tltr,

unitcommit_200_0_5_mod_7, unitcommit_200_100_2_mod_7, util, waste.

Appendix B

The options provided to the solvers (and subsolvers) in the benchmark are listed below. Otherwise default values have been used for the solvers.

Name	Value	Description
General GAMS		
MIP	CPLEX	Uses CPLEX as MIP solver
NLP	CONOPT	Uses CONOPT as NLP solver
threads	7	Max amount of threads
optCR	0.001	Relative termination tolerance
optCA	0.001	Absolute termination tolerance
nodLim	10^8	To avoid premature termination
domLim	10^8	To avoid premature termination
iterLim	10^8	To avoid premature termination
resLim	900	Time limit
BONMIN		
bonmin.algorithm	B-BB	Selects the main strategy
bonmin.time_limit	900	Sets the time limit
DICOPT		
maxcycles	10^8	Iteration limit
SBB		
memnodes	$5 \cdot 10^7$	To avoid premature termination, but not too large, since memory is preallocated
nodlim	10^{10}	
rootsolver	CONOPT.1	To use the CONOPT options below
SHOT		
Dual.MIP.NumberOfThreads	7	Max number of threads
Dual.MIP.Solver	0–2	Depending on MIP solver
Primal.FixedInteger.Solver	1	To use GAMS NLP solvers
Subsolver.GAMS.NLP.Solver	conopt	use CONOPT as GAMS NLP solver
Termination.ObjectiveGap.Absolute	0.001	Absolute termination tolerance
Termination.ObjectiveGap.Relative	0.001	Relative termination tolerance
Termination.TimeLimit	900	time limit
CONOPT (GAMS)		
RTMAXV	10^{30}	To avoid problems with unbounded variables in DICOPT and SBB

References

1. Belotti, P., Berthold, T.: Three ideas for a feasibility pump for nonconvex MINLP. Optim. Lett. **11**(1), 3–15 (2017)
2. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optim. Methods Softw. **24**, 597–634 (2009)

3. Bernal, D.E., Vigerske, S., Trespalacios, F., Grossmann, I.E.: Improving the performance of DICOPT in convex MINLP problems using a feasibility pump. Optim. Methods Softw. **35**(1), 171–190 (2020)
4. Berthold, T.: Heuristic algorithms in global MINLP solvers. Ph.D. thesis, Technische Universität Berlin (2014)
5. Bonami, P., Kilinç, M., Linderoth, J.: Algorithms and software for convex mixed integer nonlinear programs. In: Mixed integer nonlinear programming, pp. 1–39. Springer (2012)
6. Bonami, P., Lee, J.: BONMIN user's manual. Numer. Math. **4**, 1–32 (2007)
7. Boukouvala, F., Misener, R., Floudas, C.A.: Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. Eur. J. Op. Res. **252**(3), 701–727 (2016)
8. Bussieck, M.R., Dirkse, S.P., Vigerske, S.: PAVER 2.0: an open source environment for automated performance analysis of benchmarking data. J. Global Optim. **59**(2), 259–275 (2014)
9. Bussieck, M.R., Vigerske, S.: MINLP solver software. In: Wiley encyclopedia of operations research and management science. Wiley Online Library (2010)
10. Castro, P.M.: Tightening piecewise mccormick relaxations for bilinear problems. Comput. Chem. Eng. **72**, 300–311 (2015)
11. Ceccon, F., Siirola, J.D., Misener, R.: SUSPECT: MINLP special structure detector for pyomo. Optimization Letters **14**, 801–814 (2019)
12. Dakin, R.J.: A tree-search algorithm for mixed integer programming problems. Comput. J. **8**(3), 250–255 (1965)
13. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: A storm of feasibility pumps for nonconvex MINLP. Math. Program. **136**(2), 375–402 (2012)
14. D'Ambrosio, C., Lodi, A.: Mixed integer nonlinear programming tools: an updated practical overview. Ann. Op. Res. **204**(1), 301–320 (2013)
15. Dunning, I., Huchette, J., Lubin, M.: JuMP: a modeling language for mathematical optimization. SIAM Rev. **59**(2), 295–320 (2017)
16. Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. **36**(3), 307–339 (1986)
17. D'Ambrosio, C., Lodi, A., Martello, S.: Piecewise linear approximation of functions of two variables in MILP models. Op. Res. Lett. **38**(1), 39–46 (2010)
18. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Math. Program. **104**(1), 91–104 (2005)
19. Fischetti, M., Monaci, M.: A branch-and-cut algorithm for mixed-integer bilinear programming. Eur. J. Op. Res. **282**(2), 506–514 (2020)
20. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Math. Program. **66**(1), 327–349 (1994)
21. Floudas, C.A.: Deterministic global optimization, vol. 37 of nonconvex optimization and its applications (2000)
22. Fourer, R., Gay, D., Kernighan, B.: AMPL. Boyd & Fraser Danvers, MA (1993)
23. GAMS: Solver manuals (2018). https://www.gams.com/latest/docs/S_MAIN.html
24. Geoffrion, A.M.: Generalized benders decomposition. J. Optim. Theory Appl. **10**(4), 237–260 (1972)
25. Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R.L., Hendel, G., Hojny, C., Koch, T., Lübbecke, M.E., Maher, S.J., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J.M., Walter, M., Wegscheider, F., Witt, J.T., Witzig, J.: The SCIP Optimization Suite 6.0. Technical report, Optimization Online (2018)
26. Gounaris, C.E., Misener, R., Floudas, C.A.: Computational comparison of piecewise- linear relaxations for pooling problems. Ind. Eng. Chem. Res. **48**(12), 5742–5766 (2009)
27. Grossmann, I.E., Kravanja, Z.: Mixed-integer nonlinear programming: A survey of algorithms and applications. In: L.T. Biegler, T.F. Coleman, A.R. Conn, F.N. Santosa (eds.) Large-scale optimization with applications, pp. 73–100. Springer (1997)
28. Grossmann, I.E., Viswanathan, J., Vecchietti, A., Raman, R., Kalvelagen, E., et al.: GAMS/DICOPT: A discrete continuous optimization package. GAMS Corporation Inc (2002)
29. Guennebaud, G., Jacob, B., et al.: Eigen v3 (2010). http://eigen.tuxfamily.org
30. Gupta, O.K., Ravindran, A.: Branch and bound experiments in convex nonlinear integer programming. Manag. Sci. **31**(12), 1533–1546 (1985)
31. Gurobi Optimization: Gurobi optimizer reference manual (2020). https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.0/refman.pdf
32. Hart, W.E., Laird, C.D., Watson, J.P., Woodruff, D.L., Hackebeil, G.A., Nicholson, B.L., Siirola, J.D.: Pyomo-optimization modeling in Python, vol. 67. Springer, Berlin (2012)
33. Kocis, G.R., Grossmann, I.E.: Computational experience with DICOPT solving MINLP problems in process systems engineering. Comp. Chem. Eng. **13**(3), 307–315 (1989)

34. Kronqvist, J., Bernal, D., Lundell, A., Westerlund, T.: A center-cut algorithm for quickly obtaining feasible solutions and solving convex MINLP problems. Comp. Chem. Eng. (2018)
35. Kronqvist, J., Bernal, D.E., Grossmann, I.E.: Using regularization and second order information in outer approximation for convex MINLP. Mathematical Programming p. 285-310 (2020)
36. Kronqvist, J., Bernal, D.E., Lundell, A., Grossmann, I.E.: A review and comparison of solvers for convex MINLP. Optimization and Engineering pp. 1–59 (2018)
37. Kronqvist, J., Lundell, A., Westerlund, T.: The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. J. Global Optim. **64**(2), 249–272 (2016)
38. Kronqvist, J., Lundell, A., Westerlund, T.: A center-cut algorithm for solving convex mixed-integer nonlinear programming problems. In: Computer Aided Chemical Engineering, vol. 40, pp. 2131–2136. Elsevier (2017)
39. Kronqvist, J., Lundell, A., Westerlund, T.: Reformulations for utilizing separability when solving convex minlp problems. J. Global Optim. **71**(3), 571–592 (2018)
40. Kröger, O., Coffrin, C., Hijazi, H., Nagarajan, H.: Juniper: An open-source nonlinear branch-and-bound solver in julia. In: Integration of Constraint Programming, Artificial Intelligence, and Operations Research, pp. 377–386. Springer International Publishing (2018)
41. Lastusilta, T.: GAMS MINLP solver comparisons and some improvements to the AlphaECP algorithm. PhD thesis, Åbo Akademi University (2011)
42. Leyffer, S., Linderoth, J., Luedtke, J., Miller, A., Munson, T.: Applications and algorithms for mixed integer nonlinear programming. In: Journal of Physics: Conference Series, vol. 180, p. 012014. IOP Publishing (2009)
43. Liberti, L.: Reformulation techniques in mathematical programming. HDR thesis (2009)
44. Liberti, L., Cafieri, S., Tarissan, F.: Reformulations in mathematical programming: a computational approach. In: Abraham, A., Hassanien, A.E., Siarry, P., Engelbrecht, A. (eds.) Foundations of Computational Intelligence Volume 3: Global Optimization, pp. 153–234. Springer, Berlin Heidelberg, Berlin, Heidelberg (2009)
45. Liberti, L., Nannicini, G., Mladenović, N.: A good recipe for solving MINLPs. In: Matheuristics, pp. 231–244. Springer (2009)
46. Lin, Y., Schrage, L.: The global solver in the LINDO API. Optim. Methods Softw. **24**(4–5), 657–668 (2009)
47. Lundell, A.: Transformation techniques for signomial functions in global optimization. Ph.D. thesis, Åbo Akademi University (2009)
48. Lundell, A., Kronqvist, J.: On solving nonconvex MINLP problems with SHOT. In: World Congress on Global Optimization, pp. 448–457. Springer, Cham (2019)
49. Lundell, A., Kronqvist, J., Westerlund, T.: The Supporting Hyperplane Optimization Toolkit. Preprint, Optimization Online (2020)
50. Lundell, A., Skjäl, A., Westerlund, T.: A reformulation framework for global optimization. J. Global Optim. **57**(1), 115–141 (2013)
51. Lundell, A., Westerlund, J., Westerlund, T.: Some transformation techniques with applications in global optimization. J. Global Optim. **43**(2–3), 391–405 (2009)
52. Lundell, A., Westerlund, T.: Representation of the convex envelope of bilinear terms in a reformulation framework for global optimization. In: 12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering, pp. 833–838. Elsevier (2015)
53. Lundell, A., Westerlund, T.: Solving global optimization problems using reformulations and signomial transformations. Comp. Chem. Eng. **116**, 122–134 (2018)
54. Mahajan, A., Leyffer, S., Linderoth, J., Luedtke, J., Munson, T.: Minotaur: A Mixed-Integer Nonlinear Optimization Toolkit. Preprint, Optimization Online (2017)
55. Melo, W., Fampa, M., Raupp, F.: An overview of MINLP algorithms and their implementation in Muriqui Optimizer. Annals of Operations Research pp. 1–25 (2018)
56. Messine, F.: Deterministic global optimization using interval constraint propagation techniques. RAIRO Op. Res. **38**(4), 277–293 (2004)
57. MINLPLib: Mixed-integer nonlinear programming library (2020). http://www.minlplib.org/. [Downloaded January 6th 2020]
58. Misener, R., Floudas, C.A.: Piecewise-linear approximations of multidimensional functions. J. Optim. Theory Appl. **145**, 120–147 (2010)
59. Misener, R., Floudas, C.A.: ANTIGONE: Algorithms for continuous/integer global optimization of nonlinear equations. J. Global Optim. **59**(2–3), 503–526 (2014)
60. Mittelmann, H.: Benchmarks for optimization software (2018). http://plato.asu.edu/bench.html. [Accessed 28-Jan-2020]

61. Muts, P., Nowak, I., Hendrix, E.M.: The decomposition-based outer approximation algorithm for convex mixed-integer nonlinear programming. Journal of Global Optimization pp. 1–22 (2020)
62. Nagarajan, H., Lu, M., Wang, S., Bent, R., Sundar, K.: An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. J. Global Optim. **74**, 639–675 (2019)
63. Nowak, I., Breitfeld, N., Hendrix, E.M., Njacheun-Njanzoua, G.: Decomposition-based inner-and outer-refinement algorithms for global optimization. Journal of Global Optimization pp. 1–17 (2018)
64. Schichl, H., Neumaier, A.: Interval analysis on directed acyclic graphs for global optimization. J. Global Optim. **33**(4), 541–562 (2005)
65. Su, L., Tang, L., Bernal, D.E., Grossmann, I.E.: Improved quadratic cuts for convex mixed-integer nonlinear programs. Comput. Chem. Eng. **109**, 77–95 (2018)
66. Sundar, K., Nagarajan, H., Wang, S., Linderoth, J., Bent, R.: Piecewise polyhedral formulations for a multilinear term (2020)
67. Tawarmalani, M., Sahinidis, N.V.: Convexification and global optimization in continuous and mixed-integer nonlinear programming: Theory, algorithms, software, and applications, vol. 65. Springer Science & Business Media (2002)
68. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: a theoretical and computational study. Math. Program. **99**(3), 563–591 (2004)
69. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**, 225–249 (2005)
70. Trespalacios, F., Grossmann, I.E.: Review of mixed-integer nonlinear and generalized disjunctive programming methods. Chem. Ingenieur Technik **86**(7), 991–1012 (2014)
71. Vigerske, S., Gleixner, A.: SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. Optim. Methods Softw. **33**(3), 563–593 (2018)
72. Viswanathan, J., Grossmann, I.E.: A combined penalty function and outer-approximation method for MINLP optimization. Comput. Chem. Eng. **14**(7), 769–782 (1990)
73. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
74. Westerlund, T., Petterson, F.: An extended cutting plane method for solving convex MINLP problems. Comput. Chem. Eng. **19**, 131–136 (1995)
75. Westerlund, T., Pörn, R.: Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. Optim. Eng. **3**(3), 253–280 (2002)
76. Wicaksono, D.S., Karimi, I.A.: Piecewise milp under- and overestimators for global optimization of bilinear programs. AIChE J. **54**(4), 991–1008 (2008)
77. Zhou, K., Kılınç, M.R., Chen, X., Sahinidis, N.V.: An efficient strategy for the activation of MIP relaxations in a multicore global MINLP solver. J. Global Optim. **70**(3), 497–516 (2018)