A disjunctive cut strengthening technique for convex MINLP

Jan Kronqvist   Ruth Misener

Department of Computing,
Imperial College London
{j.kronqvist, r.misener}@imperial.ac.uk

August 8, 2020

Abstract

Generating polyhedral outer approximations and solving mixed-integer linear relaxations remains one of the main approaches for solving convex mixed-integer nonlinear programming (MINLP) problems. There are several algorithms based on this concept, and the efficiency is greatly affected by the tightness of the outer approximation. In this paper, we present a new framework for strengthening cutting planes of nonlinear convex constraints, to obtain tighter outer approximations. The strengthened cuts can give a tighter continuous relaxation and an overall tighter representation of the nonlinear constraints. The cuts are strengthened by analyzing disjunctive structures in the MINLP problem, and we present two types of strengthened cuts. The first type of cut is obtained by reducing the right-hand side value of the original cut, such that it forms the tightest generally valid inequality for a chosen disjunction. The second type of cut effectively uses individual right-hand side values for each term of the disjunction. We prove that both types of cuts are valid and that the second type of cut can dominate both the first type and the original cut. We use the cut strengthening in conjunction with the extended supporting hyperplane algorithm, and numerical results show that the strengthening can significantly reduce both the number of iterations and the time needed to solve convex MINLP problems.

Keywords— Disjunctive cuts, Convex MINLP, Cut strengthening, Extended supporting hyperplane algorithm.

1 Introduction

Mixed-integer nonlinear optimization (MINLP) arises in many applications across engineering, manufacturing, and the natural sciences [16]. An important MINLP subclass features exclusively convex nonlinearities, i.e. the nonconvexity of the MINLP comes only from the discrete variables [43]. Convex MINLP is highly relevant in diverse fields including process synthesis [21, 24], portfolio optimization [9, 12, 28], and constrained layout [17, 63]. For MINLP with nonconvex nonlinearities, e.g. heat integration of chemical processes [23] and pooling problems [55], optimization algorithms assuming convex nonlinearities may generate excellent primal heuristics to the original optimization problem [13, 20, 22].

Convex MINLP represents a highly successful subclass of optimization problems, e.g. algorithm developers often develop convex approximations of nonconvex engineering relationships [29] or decompose their optimization problems into a series of convex MINLP problems [48, 59]. A wide range of efficient solver software is developed specifically for convex MINLP [7, 13, 32, 38, 44, 50, 52, 53]. The success of convex MINLP derives from the seminal work of [22] in developing the outer approximation (OA) algorithm. The work by [22] became pivotal in solving convex MINLP problems because of the algorithm's strong convergence properties for a wide range of problem classes [27, 60] and its speed in solving practical problems [13]. In a recent benchmark by [43] it was shown that several of the most efficient convex MINLP solvers are based on the OA algorithm.

The concept of using an outer approximation of the nonlinear constraints for MINLP problems, developed by [22, 30], forms the core of several other convex MINLP algorithms, *e.g.,* extended cutting plane (ECP) [78, 79], feasibility pump [11], extended supporting hyperplane (ESH) [39], and the center-cut algorithm [40]. Further developments of the OA algorithm, incorporating quadratic approximations and regularization, has been presented by [71] and [41]. These algorithms could commonly be referred to as outer approximation type algorithms, although this classification is seldom used.

This paper focuses on deriving strong cutting planes for convex MINLP problems, resulting in tight outer approximations, by exploiting disjunctive structures in the problem. We use cuts obtained by the ESH algorithm as a basis, and we develop a framework for strengthening the cuts by considering the integer restrictions. The cut strengthening technique is not unique to the ESH algorithm and could also be used with an OA, ECP or generalized Benders decomposition [30] framework. The main motivation behind using the ESH algorithm is that the algorithm tends to generate a single strong cut per iteration. The ESH cuts are actually as tight as possible with regards to the nonlinear constraints [39], but they do not in general form supporting hyperplanes to the convex hull of all integer feasible solutions. Here we develop a framework for strengthening the ESH cuts, which results in two new types of cuts that are always as tight or tighter than the ESH cut. The new cuts can give both a tighter representation of the nonlinear constraints as well as a tighter continuous relaxation. By obtaining a tighter outer approximation of the nonlinear constraints, we can reduce both the number of iterations and the time needed to solve problems.

Cutting planes that strengthen the continuous relaxation are nowadays an essential part of an efficient mixed-integer linear programming (MILP) solver [1, 46], and there is an active interest in developing similar cuts for convex MINLP. Disjunctive cutting planes for convex MINLP originate from the fundamental contributions of [18] and [69], and further developments are presented in [74]. Lift-and-project cuts were first introduced in MILP by [5], and this technique has later been adopted within convex MINLP. By linearizing the constraints, a polyhedral outer approximation can be used to derive lift-and-project cuts through a cut generating LP [10, 37, 64, 80]. An alternative approach is presented by [47], where they obtain cuts directly by solving cut generating conic programs. Other types of cuts used within MINLP includes different types of mixed-integer rounding cuts [2, 31], reformulation linearization technique (RLT) based cuts [56, 66], and split cuts [57].

The cut strengthening techniques presented here can be viewed as an alternative approach to the previously mentioned lift-and-project and disjunctive cuts. However, our cut strengthening procedure is more focused on obtaining a tight MILP relaxation, than getting the best improvement for the continuous relaxation. The cuts are generated by selecting a disjunction of the MINLP problem and strengthening an ESH cut over the convex hull of the selected disjunction. [74] use a somewhat similar idea, where they derive a supporting hyperplane for a nonlinear disjunction by solving a separation problem. Instead of solving a separation problem, we strengthen the ESH cut by deriving the smallest possible right-hand side values to the ESH cut that are still valid for each term of the disjunction. This enables us to effectively use individual right-hand side values for each term of the disjunction, making the cut tight for each disjunct. A similar approach is used by [73] to construct tighter big-M reformulations of generalized disjunctive programs. We determine right-hand side values of the cuts by solving independent convex NLP problems in the original variable space and do not rely on the convex hull formulation of the disjunctions. By doing so, numerical difficulties associated with the perspective function are avoided and instead of solving a larger (lifted) problem, we solve several smaller independent (parallelizable) problems. This approach also enables us to identify some infeasible integer assignments and to handle numerical tolerances in a straightforward fashion. To the authors' best knowledge, this is a novel cut strengthening technique for convex MINLP.

The paper is organized as follows. Section 2 gives a short description of the ESH algorithm, along with the necessary assumptions on the MINLP problems. Section 3 presents the theory and techniques used for the cut strengthening, and a cut strengthening algorithm is presented in Section 4. Section 5 presents an algorithm for solving convex MINLP problems that combines the ESH algorithm with the cut strengthening techniques. Finally, some numerical results are presented in Section 6.

2 Background

First, we define the class of problems considered within the paper and state the assumptions needed to guarantee convergence of the ESH algorithm. The disjunctive structure that the cut strengthening technique builds upon is also presented in this section. The second part of this section briefly describes the ESH algorithm, which is later used to generate cuts and forms the basis of the convex MINLP algorithm in section 5.

2.1 Problem statement

The most commonly used, and most practical, definition of a convex MINLP problem, is that all of the nonlinear constraints and objective are given by convex functions [15, 33, 60, 78]. Throughout the paper, we use this definition of convexity. Without loss of generality, we only consider convex MINLP problems with the following structure

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{B}\mathbf{x} = \mathbf{d}, \\ & g_j(\mathbf{x}) \leq 0, \quad \forall j = 1, 2, \ldots, l, \\ & \mathbf{x} \in \mathbb{R}^n, \\ & x_i \in \mathbb{Z}, \quad \forall i \in I_{\mathbb{Z}}, \end{aligned} \tag{MINLP}$$

where $g_j : \mathbb{R}^n \to \mathbb{R}$ are convex continuously differentiable functions. Here, $I_{\mathbb{Z}}$ is a set containing the indices of all the integer variables. To clarify the notation, x_i referrers to the i-th element of the variable vector $\mathbf{x}$. The feasible set defined by the nonlinear constraints will be referred to as the nonlinear feasible set, and it is given by

$$N = \{\mathbf{x} \in \mathbb{R}^n \mid g_j(\mathbf{x}) \leq 0 \quad \forall j = 1, \ldots l\}. \tag{1}$$

To simplify the notation, we will also introduce a set L defined by the linear constraints and a set Y given by the variable domains

$$\begin{aligned} L &= \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b},\ \mathbf{B}\mathbf{x} = \mathbf{d}\}, \\ Y &= \{\mathbf{x} \in \mathbb{R}^n \mid \ x_i \in \mathbb{Z} \quad \forall i \in I_{\mathbb{Z}}\}. \end{aligned}$$

To ensure convergence of the ESH algorithm, we need to make the following assumptions of problem (MINLP).

Assumption 1 *The linear constraints form a compact set.*

Assumption 2 *The continuous relaxation of problem* (MINLP) *satisfies Slater's condition [68].*

For the cut strengthening procedure, we make the following assumption on the problem structure.

Assumption 3 *The MINLP problem contains at least one exclusive selection constraint of binary variables,* i.e., $\exists\ I_D \subset I_{\mathbb{Z}} :\ x_i \in \{0, 1\} \quad \forall i \in I_D$*, and either one of the constraints*

$$\sum_{i \in I_d} x_i = 1, \tag{2}$$

$$\sum_{i \in I_d} x_i \leq 1, \tag{3}$$

appears in the problem.

For the sake of simplicity and clarity, we will throughout the paper only focus on the exclusive selection constraint (2). The second type of exclusive selection constraint (3), can trivially be converted into the first type by introducing a slack binary variable and can be handled by the same approach.

The exclusive selection constraints arise, for example, from the representation of disjunctive constraints through the so-called *big-M* or *convex hull* formulation [3, 61, 72]. Note that we do not restrict all of the integer variables to be binary variables, nor do we assume the problems to have disjunctive constraints of a specific type. The cut strengthening simply requires that the problem contains at least one exclusive selection constraint, which is used for strengthening the cut. However, the cut strengthening is most powerful in case the problem contains the big-M constraints, resulting in a weak continuous relaxation. Therefore, we focus on problems containing big-M constraints.

For the cut strengthening to be computationally efficient, the number of elements in I_D should be less than the elements in $I_{\mathbb{Z}}$. Throughout the paper, we also assume that the main challenges in solving problem (MINLP) arise from the integer restrictions. Consequently, we assume that a continuous relaxation of the problem is significantly easier to solve than the MILP relaxations used by OA, ECP, and ESH. This is often the case for convex MINLP problems which is, for example, shown by the numerical result in [58, 70].

2.2 The extended supporting hyperplane algorithm

The ESH algorithm was presented by [39] as a method for solving convex MINLP problems, and it builds upon ideas presented by [76]. It was proven by [26] that the ESH algorithm is directly applicable to nonsmooth MINLP problems with constraints given by pseudoconvex functions. Properties of the ESH algorithm have also been further analyzed by [65].

The ESH algorithm constructs a tight polyhedral outer approximation of the nonlinear feasible set N, by generating supporting hyperplanes to the set. The polyhedral outer approximation at iteration k is given by

$$\hat{N_k} = \left\{ \nabla g_j \left(\bar{\mathbf{x}}^i\right)^\top \left(\mathbf{x} - \bar{\mathbf{x}}^i\right) \leq 0 \quad \forall i = 1, 2 \ldots k,\ j \in A_i \right\}, \tag{4}$$

where $\bar{\mathbf{x}}^i$ are points on the boundary of N and A_i contains the indices of all constrains active at $\bar{\mathbf{x}}^i$. From convexity it directly follows that $N \subseteq \hat{N_k}$, and $\hat{N_k}$ is commonly referred to as an outer approximation of N.

A new trial solution $\mathbf{x}^{k+1}$ is obtained by solving the following MILP relaxation

$$\begin{aligned} \mathbf{x}^{k+1} \in \ \underset{\mathbf{x}}{\arg\min} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in L \cap \hat{N_k} \cap Y. \end{aligned} \tag{MILP-r}$$

A lower bound on the optimal objective value of problem (MINLP) is given by $\mathbf{c}^\top \mathbf{x}^{k+1}$, where $\mathbf{x}^{k+1}$ is an optimal solution to the MILP relaxation.

The trial solutions obtained by solving problem (MILP-r) will all be outside of the nonlinear feasible set N, before the very last iteration. Therefore, linearizing the nonlinear constraints at the trial solutions $\mathbf{x}^k$ would, in general, not form supporting hyperplanes to N and would result in weaker cuts. To obtain supporting hyperplanes, ESH performs an approximative projection of the trial solution $\mathbf{x}^k$ onto $N \cap L$. A point in the interior of $N \cap L$ is needed for the projection, and such a point is obtained by solving the convex continuous problem

$$\begin{aligned} \mathbf{x}_{\text{int}}, \mu \in \ \underset{\mathbf{x},\mu}{\arg\min} \quad & \mu \\ \text{s.t.} \quad & g_j(\mathbf{x}) \leq \mu, \quad \forall j = 1, 2, \ldots, l \\ & \mathbf{x} \in L, \\ & \mu \in \mathbb{R}. \end{aligned} \tag{NLP-IP}$$

For the approximative projection of $\mathbf{x}^k$, we define the one-dimensional function

$$F(\lambda) = \max_j \left\{ g_j\left(\lambda \mathbf{x}_{\text{int}} + (1-\lambda)\mathbf{x}^k\right)\right\}, \tag{5}$$

for $\lambda \in [0,1]$. Using a simple root-search algorithm we can obtain a λ^k such that $F\left(\lambda^k\right) = 0$. The approximative projection of $\mathbf{x}^k$ onto $N \cap L$ is then given by

$$\bar{\mathbf{x}}^k = \lambda^k \mathbf{x}_{\text{int}} + (1-\lambda^k)\mathbf{x}^k. \tag{6}$$

Now, if the active constraints are linearized at $\bar{\mathbf{x}}^k$ we obtain the following cuts

$$\nabla g_j\left(\bar{\mathbf{x}}^k\right)^\top \left(\mathbf{x} - \bar{\mathbf{x}}^k\right) \le 0 \quad \forall j \in A_i, \tag{7}$$

which forms supporting hyperplanes to $N \cap L$. The supporting hyperplanes are then added to the current polyhedral outer approximation to form $\hat{N}_{k+1}$, which ensures that $\bar{\mathbf{x}}^k \notin \hat{N}_{k+1}$.

The ESH algorithm repeats the procedure of solving (MILP-r) and improving the outer approximation by generating supporting hyperplanes. To improve the computational performance, the algorithm starts by further relaxing (MILP-r) and solving LP relaxations to quickly generate an outer approximation. For more details and computational enhancements on the ESH algorithm see [51].

The cuts generated by the ESH algorithm are as tight as possible with regards to $N \cap L$. However, there is no guarantee that the algorithm generates supporting hyperplanes to the convex hull of $N \cap L \cap Y$. Therefore, it can be possible to further strengthen the cuts by considering the integrality restrictions. To illustrate the possible strengthening of the cuts, consider the following example

$$\begin{aligned}
\min_{\mathbf{x}} \quad & -x_1 - x_2 \\
\text{s.t.} \quad & (x_1-1)^2 + (x_2-2)^2 \le 1 + 29.944(1-x_3), \\
& (x_1-2)^2 + (x_2-5)^2 \le 1 + 29.944(1-x_4), \\
& (x_1-4)^2 + (x_2-1)^2 \le 1 + 29.944(1-x_5), \\
& x_3 + x_4 + x_5 = 1, \\
& 0 \le x_1 \le 8,\ 0 \le x_2 \le 8, \\
& x_1, x_2 \in \mathbb{R},\ x_3, x_4, x_5 \in \{0,1\}.
\end{aligned} \tag{EX1}$$

The example contains the disjunctive constraint that the (x_1, x_2)-variables must be within one of three circles, which is represented by the big-M formulation. The value 29.944 is, in this case, the tightest common value for the big-M coefficients. A stronger problem formulation could simply be obtained by using individual M values for each constraint, which can easily be determined as described in the Appendix. We only use the weaker formulation in order to better highlight differences between the cuts. Figure 1 shows the feasible set of problem (EX1) along with the continuously relaxed feasible set projected down onto the (x_1, x_2)-space.

In the first iteration, the ESH algorithm will generate the following cut

$$5.920x_1 + 4.536x_2 + 29.944x_3 \le 59.249, \tag{8}$$

which forms a supporting hyperplane to $N \cap L$ but not a supporting hyperplane to convex hull of $N \cap L \cap Y$. From Figure 1, it is clear that the cut given by eq. (8) is not as tight as possible when considering the integer properties. In the next section, we present a technique to further tighten the cut by utilizing the disjunctive structures of the MINLP problem.

3 Cut strengthening

From the example in the previous section, it can be observed that the ESH cut can be tightened by simply reducing the right-hand side and still remain valid for the integer feasible set, *i.e.,*

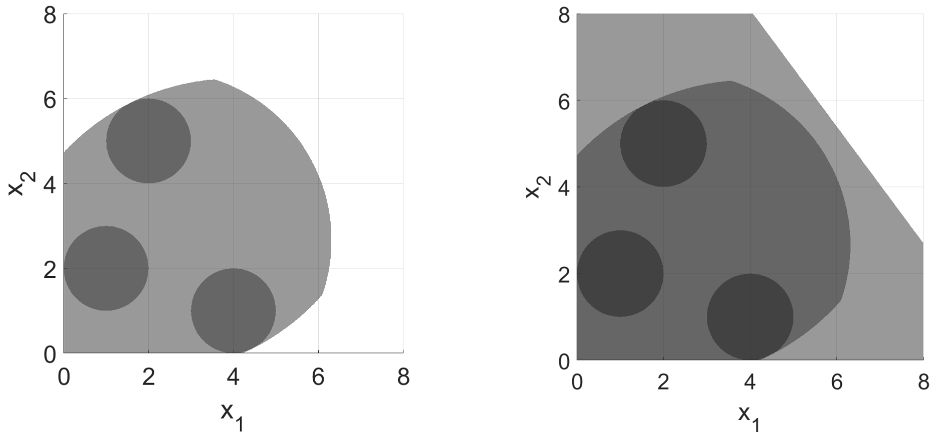


Figure 1: The dark circles show the feasible set of problem (EX1) projected onto the (x_1, x_2)-space. The light gray area in the left figure shows the feasible set of the continuous relaxation. The right figure also shows the projection of the outer approximation obtained by the first iteration of the ESH algorithm. Note that, a supporting hyperplane to $N \cap L$ does not necessarily form a supporting hyperplane in a projected space, as shown in the figure.

$N \cap L \cap Y$. To reduce the right-hand side, we will consider an exclusive selection constraint, see assumption 3, and determine the smallest right-hand side values for each selection. This enables us to strengthen the cut by reducing the right-hand side alone or to further strengthen the cut by assigning individual right-hand side values for each assignment of the exclusive selection constraint.

First, we select an index set I_D ($I_D \subset I_{\mathbb{Z}}$) that contains the indices of all the binary variables included in an exclusive selection constraint of the MINLP problem. By using the ESH algorithm we obtain the cut

$$\alpha^\top \mathbf{x} \leq \beta, \tag{9}$$

which forms a tight valid inequality for $N \cap L$. To tighten cut (9), consider the following disjunctive programming (DP) problem

$$\begin{aligned} z^* = \max_{\mathbf{x}} \quad & \alpha^\top \mathbf{x} \\ \text{s.t.} \quad & \bigvee_{i \in I_D} \begin{bmatrix} \mathbf{x} \in N \cap L \\ x_i = 1 \\ x_j = 0 \ \ \forall j \in I_D \setminus i \end{bmatrix}. \end{aligned} \tag{10}$$

This DP problem can be solved as a convex NLP through the convex hull formulation [18, 45, 69]. Formulating problem (10) as a convex NLP through a convex hull formulation can cause numerical difficulties, such as division by zero and non-smoothness [63], and the problem will contain $|I_D|$ copies of the variables. Instead of solving (10) as a single large problem we solve it as smaller individual convex problems, by considering the following alternative formulation of problem (10)

$$\begin{aligned} z^* = \max_{i \in I_D} \quad b_i = \max_{\mathbf{x}} \, & \alpha^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in N \cap L, \\ & x_i = 1, \\ & x_j = 0, \quad \forall j \in I_D \setminus i. \end{aligned} \tag{11}$$

By solving each inner problem of (11) separately we can determine z^* as the largest b_i. This approach requires $|I_D|$ independent convex NLP problems to be solved, but computationally it can be more efficient than solving a single problem with $|I_D|$ copies of the variables. Using z^* as the new right-hand side value of cut (9), we form the tightened cut

$$\alpha^\top \mathbf{x} \leq z^*. \tag{12}$$

Proposition 1 *The cut given by eq.* (12) *forms a valid inequality for $N \cap L \cap Y$, and is at least as tight as the cut given by eq.* (9)*.*

Proof: From optimality of problem (10) it directly follows that cut (12) forms a supporting hyperplane to the feasible set of problem (10), which contains $N \cap L \cap Y$. Since the feasible set of problem (10) is contained within $N \cap L$, it follows that $z^* \leq \beta$. $\square$

Solving (10) as smaller individual convex problems also enables us to further tighten the cut. To further strengthen the cut, we considering each term of the disjunction in problem (10) and form a convex NLP problem for each $i \in I_D$

$$\begin{aligned} b_i = \max_{\mathbf{x}} \quad & \alpha^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in N \cap L, \\ & x_i = 1, \\ & x_j = 0, \quad \forall j \in I_D \setminus i. \end{aligned} \tag{NLP-i}$$

Note that each problem (NLP-i) is a subproblem of (11). To simplify the derivation and analysis, we first assume that all $i \in I_D$ result in a feasible problem (NLP-i). Solving problem (NLP-i) for each $i \in I_D$ gives the values b_i that can be used as individual right-hand side values for each integer assignment of the exclusive selection constraint (2). A new strengthened cut is then given by

$$\alpha^\top \mathbf{x} \leq \sum_{i \in I_D} b_i x_i, \tag{13}$$

and the properties of the new cut are presented in the following two theorems.

Theorem 1 *The cut given by eq.* (13) *forms a valid inequality for $N \cap L \cap Y$.*

Proof: The theorem is easily proven by contradiction. First, assume $\exists\ \bar{\mathbf{x}} \in N \cap L \cap Y$:

$$\alpha^\top \bar{\mathbf{x}} > \sum_{i \in I_D} b_i x_i. \tag{14}$$

Due to the exclusive selection constraint, one and only one of the binary variables $x_{i \in I_D}$ can be nonzero. Let j be the index of the nonzero binary variable, and the strict inequality (14) can now be written as

$$\alpha^\top \bar{\mathbf{x}} > b_j. \tag{15}$$

By assumption, $\bar{\mathbf{x}}$ must satisfy all constraints of problem (NLP-i). This implies that b_j cannot be an optimal solution to problem (NLP-i), and this leads to a contradiction. $\square$

Before analyzing the tightness of the cuts, we first describe our definition of a tighter cut. Here we consider cut (13) to be tighter than cut (12) in the sense that any $\mathbf{x}$ satisfying eq. (13) will satisfy eq. (12), but not vice versa. In integer programming, this tightness relation is commonly referred to as cut (13) strictly dominating cut (12), *e.g.,* see [4].

Theorem 2 *The cut given by eq.* (13) *is always as tight or tighter than the cut given eq.* (12)*.*

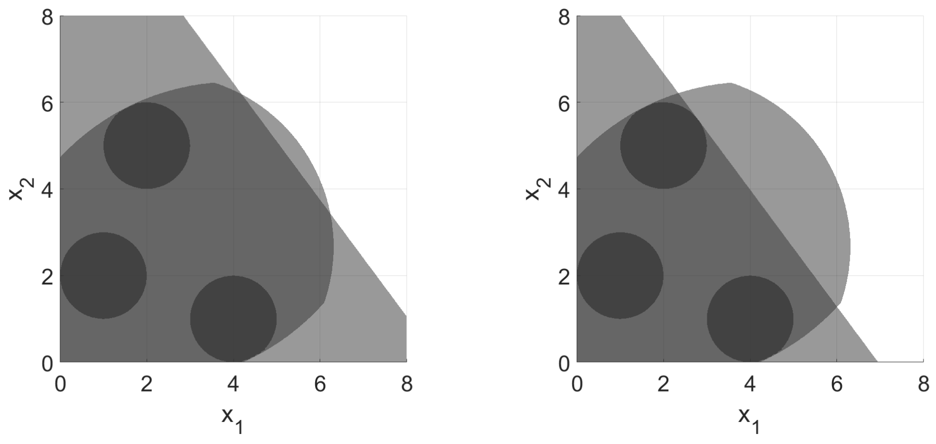


Figure 2: The figures show the true feasible set of problem (EX1) and the continuously relaxed feasible set projected onto the (x_1, x_2)-space. The left figure shows the outer approximation given by cut (17) and the right figure shows the outer approximation given by cut (18).

Proof: Since z^* is chosen as the maximum of $\alpha^\top \mathbf{x}$ over all integer assignments of the exclusive selection constraint intersected with $N \cap L$, it follows that $z^* = max_{i \in I_D}\{b_i\}$. Therefore, each b_i can be split into two parts $b_i = z^* - \Delta_i$, where each $\Delta_i \geq 0$. The cut given by eq. (13) can now be written as

$$\alpha^\top \mathbf{x} \leq z^* - \sum_{i \in I_D} \Delta_i x_i, \tag{16}$$

proving that the cut is always as tight as cut (12). Furthermore, if a single $\Delta_i > 0$, then the cut given by (13) will strictly dominate cut (12). □

Earlier we assumed that all $i \in I_D$ result in a feasible problem (NLP-i), which is not a necessary assumption for the cut strengthening. Finding such infeasible integer assignments enables us to remove the corresponding binary variable, as further described in the following proposition.

Proposition 2 *If $i \in I_D$ result in an infeasible problem* (NLP-i)*, then the binary variable x_i can be eliminated by permanently fixing the variable to zero.*

Proof: In problem (NLP-i) all variables, except those included in the exclusive selection constraint, are relaxed to continuous variables and they are only restricted by the original constraints. Variable x_i is fixed to one, which automatically fixes the other variables in the exclusive selection constraint to zero. Therefore, the only case where problem (NLP-i) can be infeasible is when $x_i = 1$ is an infeasible partial integer assignment to the MINLP problem. □

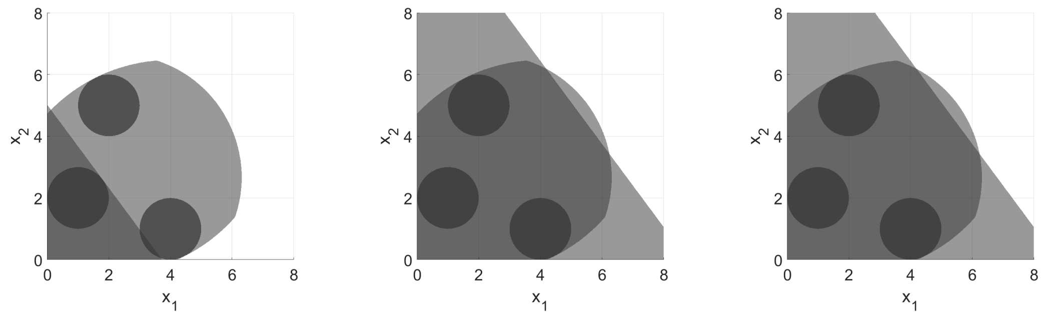
To illustrate the difference between the two cuts, we again consider problem (EX1). By applying the cut strengthening technique to the cut given by the ESH algorithm, we can generate the following two cuts

$$5.920x_1 + 4.536x_2 + 29.944x_3 \leq 52.029, \tag{17}$$

$$5.920x_1 + 4.536x_2 \leq (52.029 - 29.944)x_3 + 41.192x_4 + 35.451x_5. \tag{18}$$

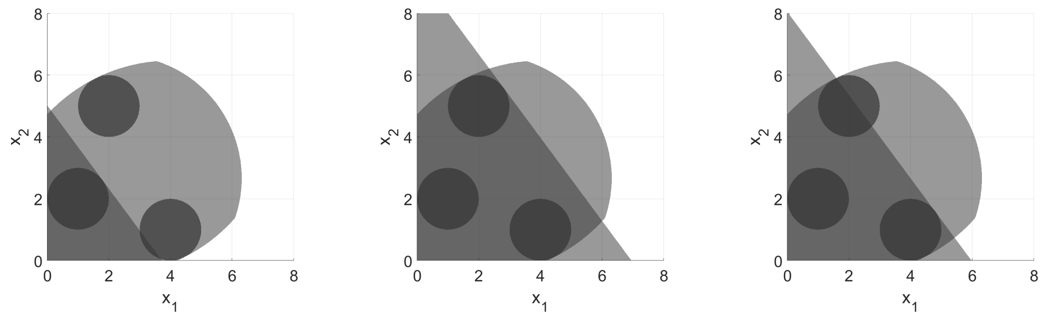
The outer approximations obtained given by the two different cuts are shown in Figure 2. The figure shows a clear advantage of the second cut, resulting in a significantly tighter linear relaxation of the MINLP problem. However, comparing Figure 1 and Figure 2 show that both cuts are significantly stronger than the standard ESH cut.

In an outer approximation type algorithm, it is not only important to obtain a tight continuous relaxation, but also to obtain a tight MILP relaxation, *i.e.,* a tight linear relaxation for given



(a) $x_3 = 1$ (b) $x_4 = 1$ (c) $x_5 = 1$

Figure 3: The dark circles show the feasible set of problem (EX1) projected onto the (x_1, x_2)-space. The light gray area in the figures shows the feasible set of the continuous relaxation. Furthermore, the figures also show the feasible set of cut (17) for each feasible integer assignment.



(a) $x_3 = 1$ (b) $x_4 = 1$ (c) $x_5 = 1$

Figure 4: The figures show the feasible set of cut (18) for each feasible integer assignment in the (x_1, x_2)-space.

integer assignments. The two are obviously related, but it is possible to have a tight MILP relaxation with a weak continuous relaxation. To further illustrate the differences between the two types of cuts, we analyze how the feasible region of the cuts to problem (EX1) varies with the feasible integer assignments. Figure 3 shows the feasible region of the cut given by eq. (17) for each feasible integer assignment. The figure shows that cut (17) is tight for one of the feasible integer assignments, but not as tight as possible for the other two.

Figure 4 shows the cut given by eq. (18) forms a supporting hyperplane to the feasible set of each term of the disjunction in problem (EX1), *i.e.,* for each feasible integer assignment the cut is as tight as possible. The example highlights the fact that the individually tightened cuts, *i.e.,* cuts formed by eq. (13), can give both significantly tighter continuous and MILP relaxations than the cut given by eq. (12) and the original ESH cut.

In this section, we have presented a framework for strengthening cuts obtained by the ESH algorithm. However, the same approach can also be used to strengthen cuts obtained by a similar algorithm, such as ECP, OA or generalized Benders decomposition. The next section will focus more on the computational aspects, and how to practically utilize the cut strengthening framework within a solver.

4 A cut strengthening algorithm

This section focuses on the computational aspects and how to utilize the cut strengthening techniques from the previous section in an algorithm. We present a simple strategy for selecting one out of multiple exclusive selection constraints, and describe some computational enhancements along with a discussion on how to deal with tolerances.

The cut strengthening techniques in the previous section utilizes the exclusive selection constraint (2) to tighten cuts of the type given by eq. (9). However, MINLP problems can contain multiple exclusive selection constraints, *e.g.,* originating from multiple disjunctive constraints. Given a cut, there is a choice of which exclusive selection constraint and the corresponding variables to choose for the tightening procedure. Ideally one wants to choose the exclusive selection constraint with the binary variables x_i for $i \in I_D$ such that the coefficients b_i obtained by solving (NLP-i) are as small as possible. However, such an optimal choice cannot trivially be determined, and instead, we will make the choice based on the variable connections.

Suppose that we have obtained cut (9), which is given by linearizing the nonlinear constraint $g_j(\mathbf{x}) \leq 0$. To compare the different exclusive selection constraints, and their corresponding variables x_i for $i \in I_D$, we check the connections of the variables x_i for $i \in I_D$ to the constraint $g_j(\mathbf{x}) \leq 0$. Here we consider two types of connections, *direct connections* and *step-one connections.* Variable x_i is directly connected to $g_j(\mathbf{x}) \leq 0$, if the variable is included in the constraint. In a step-one connection, the variable x_i is included in another constraint (linear or nonlinear) that has at least one variable in common with $g_j(\mathbf{x}) \leq 0$. The number of direct connections in an exclusive selection constraint is given by number of variables in I_D that are directly connected to the nonlinear constraint $g_j(\mathbf{x}) \leq 0$, and similarly for the step-one connections. Here, we use the following heuristic rule for selecting an exclusive selection constraint.

Rule 1 *Given cut* (9)*, select the exclusive selection constraint with the largest number of direct connections to the corresponding nonlinear constraint. If there are no direct connections, chose the one with the largest number of step-one connections. In case of multiple exclusive selection constraints with the same number of connections, chose one of them randomly.*

A feasible solution to the MINLP problem $\hat{\mathbf{x}}$ can also be utilized within the cut strengthening procedure. This is done by simply including the objective reduction constraint

$$\mathbf{c}^\top \mathbf{x} \leq \mathbf{c}^\top \hat{\mathbf{x}}, \tag{19}$$

as a constraint in problem (NLP-i). Including the objective reduction constraint can further reduce the coefficients b_i, resulting in a stronger cuts. Furthermore, including the objective reduction constraint can enforce infeasibility on some partial integer assignments, and cause assignments in problem (NLP-i) to be infeasible. As mentioned earlier, the only way problem (NLP-i) can be infeasible is if the partial assignment, *i.e.,* $x_i = 1 \; i \in I_D, \; x_j = 0 \; \forall j \in I_D \setminus i$, is infeasible for the MINLP problem. Finding such infeasibilities is desirable since it allows us to eliminate a variable from the MINLP problem by fixing it to zero.

Including the previously tightened cuts into problem (NLP-i) can also improve performance by tightening the continuous relaxation. Obtaining a tighter continuous relaxation in problem (NLP-i) can further strengthen the cut and infer infeasibilities. In the numerical results presented in Section 6, it was noticed that including the tightened cuts and an objective reduction constraint can greatly help in identifying infeasible or non-optimal partial integer assignments. The ability to identify and eliminate these from the search space can result in fewer iterations but can also reduce the complexity of the MILP relaxations, used by algorithms such as ESH, ECP, and OA.

The cut strengthening techniques are summarized as pseudo-code in Algorithm 1. In the algorithm, the two different cuts from the previous section are considered as different strategies. The cut given by eq. (13) is referred to as a *Multi Tightening* (MT) strategy, since it effectively uses multiple values for the right-hand side. Similarly, the cut given by eq. (12) is referred to as a *Single Tightening* (ST) strategy.

Algorithm 1 Cut strengthening algorithm overview

```
1: procedure CutStrengthening($\alpha, \beta$, Strategy, $\hat{\mathbf{x}}$)
2:     $I_D \leftarrow$ SelectExclusiveConstraint($\alpha^\top \mathbf{x} \leq \beta$)          ▷ Rule 1, Section 4
3:     if A feasible solution is known then
4:         MINLP $\leftarrow$ IncludeObjectiveConstraint($\hat{\mathbf{x}}$)          ▷ Equation (19), Section 4
5:     end if
6:     for $i \in I_D$ do
7:         $b_i \leftarrow$ SolveNLP($i$)          ▷ Problem (NLP-i), Section 3
8:         if NLP is infeasible then
9:             Remove variable $x_i$ from MINLP problem          ▷ Fix $x_i = 0$
10:        end if
11:    end for
12:    if Strategy = ST then
13:        $z^* \leftarrow \max_{i \in I_D} \{b_i\}$
14:        return The cut $\alpha^\top \mathbf{x} \leq z^*$          ▷ Cut by eq. (12)
15:    else if Strategy = MT then
16:        return The cut $\alpha^\top \mathbf{x} \leq \sum_{i \in I_D} b_i x_i$          ▷ Cut by eq. (13)
17:    else if Strategy = ESH then
18:        return The original ESH cut.
19:    end if
20: end procedure
```

4.1 Computational comments

When solving an optimization problem to generate a cut, it is important to take the solver tolerances into consideration. The tolerances are especially important when dealing with nonlinear problems, where it is rare that a solver returns an *exact* optimal solution. In the cut strengthening procedure, presented in the previous section, the solver tolerance will only affect the coefficients b_i. If we can ensure that the solution of problem (NLP-i) is within an ϵ-tolerance from the true minimum objective value, then the suboptimality can easily be handled by relaxing the cut, *i.e.,* adding ϵ to the right-hand side.

As a comparison, some other techniques to obtain strong cuts for convex MINLP problems use the minimum distance (separation) problem to generate cuts [14, 69, 74]. In these approaches, the minimizer of an NLP subproblem forms the coefficients of both the left- and right-hand side of the cut. For these cuts, it is important to obtain a high optimality accuracy in the variable space, since it affects both the angle and level of the cut. Issues with numerical tolerances can be reduced or effectively eliminated, *e.g.,* by post-processing the cut and optimizing over each term in the disjunction to determine a valid right-hand side, but this comes at a significant computational expense. However, since both the coefficients on the left- and right-hand side are optimized, this approach is not limited to a specific cut but can basically generate any supporting hyperplane to the convex hull of the disjunction. Generating cuts by solving the separation problem can, therefore, result in stronger cuts than the cut strengthening procedure which is limited by the structure of the original cut.

In the cut strengthening procedure, we optimize over each term of a disjunction in problem (10) separately. This allows us to obtain stronger cuts and identify infeasible partial integer assignments, as described in Section 3. In an efficient implementation, the individual problems given by (NLP-i) can be solved in parallel since they are completely independent. This approach also has computational advantages, since the convex hull formulation and the perspective function, in particular, comes with numerical challenges. There are formulations to avoid division by zero [62] and for some types of problems, the convex hull is second-order cone representable, which can be handled more efficiently [6]. However, if some of the partial integer assignments are infeasible it can cause difficulties for solvers since the convex hull of problems (NLP-i) will then have an empty interior even though it is feasible. Such issues can be eliminated by analyzing each term of the disjunction in a pre-processing and eliminating infeasible terms, but this also comes at a significant computational expense.

As previously mentioned, our cut strengthening approach is limited to a specific cut and, therefore, it may result in a weaker cut compared to generating the cut from solving a separation problem. The main advantage of our cut strengthening approach is that the cut is obtained by solving several smaller independent convex problems, compared to solving the larger separation problem. Therefore, the trade-off of our cut strengthening approach is a reduced computational complexity at the expense of a possibly weaker cut.

5 Computational setup

To compare the cuts and to show the advantage of the cut strengthening, we have included a numerical study where we compare the ESH algorithm with and without the cut strengthening technique. These are preliminary results and are mainly intended as a proof of concept. To focus on the effects of cut strengthening, we apply them to a basic implementation of the ESH algorithm. As shown by [49, 50] several other techniques can be combined with the algorithm to improve the computational performance, such as early MILP termination and multiple cut generation strategies. Before presenting the results, we will give a more detailed description of the computational setup.

5.1 A Convex MINLP algorithm

To solve the MINLP problems we will use the ESH algorithm, which was briefly presented in Section 2. In each iteration, we use the cut strengthening algorithm from Section 4 to strengthen the cut generated by the ESH algorithm. It is known that the basic ESH algorithm tends to only generate a single cut per iteration [39, 50]. However, in some iterations the root-search can result in a point where multiple constraints are active, resulting in multiple cuts. Here, we will only strengthen one cut per iteration. If we obtain multiple cuts in an iteration, then we randomly pick one of them for the strengthening procedure. We do not use the LP-preprocessing from [39], which simplifies the algorithm and allows us to better focus on the effect of the cut strengthening.

Besides the basic ESH algorithm, we only include two simple primal heuristics that have proven to be effective within this framework [39, 51]. Without any primal heuristics the ESH algorithm will generally not obtain feasible solutions during the solution procedure, making it difficult to terminate based on the optimality gap. Therefore, the primal heuristics is an important enhancement to the ESH algorithm and practically needed within a solver. From the numerical tests, we also noticed that feasible solutions improve the cuts and help to identify non-optimal partial integer assignments. The primal heuristics we use here are checking the alternative solution in the MILP solver's solution pool, and fixing the integer assignments in the MINLP and solving the resulting convex NLP problem in every fourth iteration. The primal heuristics are summarized as a pseudo-code in Algorithm 2. For more details on heuristics in combination with the ESH algorithm, see [50]. For a summary of different primal heuristics see, for example, [8, 20].

As a termination criterion we use the relative optimality gap defined as

$$\text{gap} = \frac{\text{ub} - \text{lb}}{|\text{ub}| + 10^{-10}}, \tag{20}$$

where ub and lb are upper and lower bounds on the optimal objective value of the MINLP problem. Here, ub is given by the best found feasible solution and lb is given by $\mathbf{c}^\top \mathbf{x}^k$, where $\mathbf{x}^k$ is given by problem (MILP-r). We consider the MINLP problem as solved when the relative gap is reduced to 10^{-3}, thus proving that the best found solution is within 0.1% of the global optimum. The method used for solving the MINLP problems is summarized as a pseudo-code in Algorithm 3.

Algorithm 2 Primal heuristics

```
1:  procedure PrimalHeuristic($\mathbf{c}, N, L, Y$, SolutionPool, $k$, $\hat{\mathbf{x}}, \mathbf{x}^k$)
2:      for $\mathbf{x}^i \in$ SolutionPool do
3:          if $\mathbf{x}^i \in N \cap L \cap Y \mid \mathbf{c}^\top \mathbf{x}^i < \mathbf{c}^\top \hat{\mathbf{x}}$ then      ▷ Check alternative MILP solutions
4:              $\hat{Y} \leftarrow \{\mathbf{x} \in L \cap N \mid x_i = x_j^i \ \forall j \in I_\mathbb{Z}\}$      ▷ Fixed integer assignment
5:              $\hat{\mathbf{x}} \leftarrow \arg\min_{\mathbf{x} \in \hat{\mathbf{Y}}} \mathbf{c}^\top \mathbf{x}$
6:          end if
7:      end for
8:      if $k$ mod $4 = 0$ then      ▷ Every fourth main iteration
9:          $\hat{Y} \leftarrow \{\mathbf{x} \in L \cap N \mid x_i = x_i^k \ \forall i \in I_\mathbb{Z}\}$      ▷ Fixed integer assignment
10:         if MINLP is feasible for the integer assignment then
11:             $\tilde{\mathbf{x}} \leftarrow \arg\min_{\mathbf{x} \in \hat{\mathbf{Y}}} \mathbf{c}^\top \mathbf{x}$
12:             if $\mathbf{c}^\top \tilde{\mathbf{x}} < \mathbf{c}^\top \hat{\mathbf{x}}$ then
13:                 $\hat{\mathbf{x}} \leftarrow \tilde{\mathbf{x}}$
14:             end if
15:         end if
16:     end if
17:     return $\hat{\mathbf{x}}$
18: end procedure
```

Algorithm 3 Overview of the ESH algorithm with cut strengthening

```
1:  Select Strategy $\in$ {MT, ST, ESH}.
2:  Initialize: $k \leftarrow 1$, $\hat{N}_0 \leftarrow \mathbb{R}^n$, gap $\leftarrow \infty$.
3:  $\mathbf{x}_{\text{int}} \leftarrow$ ObtainInteriorPoint($N, L$)      ▷ Problem (NLP-IP), Section 2
4:  while gap $> 0.001$ do
5:      $(\mathbf{x}^k$, SolutionPool$) \leftarrow$ Relaxation($L, \hat{N}_{k-1}$)      ▷ Problem (MILP-r), Section 2
6:      $\bar{\mathbf{x}}^k \leftarrow$ RootSearch($\mathbf{x}_{\text{int}}, \mathbf{x}^k, N$)      ▷ Section 2
7:      $(\alpha, \beta) \leftarrow$GenerateCut($\bar{\mathbf{x}}^k, N$)      ▷ Cut given by eq. (7)
8:      $(\alpha, \beta^*) \leftarrow$ CutStrengthening($\alpha, \beta$, Strategy, $\hat{\mathbf{x}}$)      ▷ Algorithm 1
9:      $\hat{N}_k \leftarrow \hat{N}_{k-1} \cap \alpha^\top \mathbf{x} \leq \beta^*$
10:     $\hat{\mathbf{x}} \leftarrow$ PrimalHeuristics($\mathbf{c}, N, L, Y$, SolutionPool, $k$, $\hat{\mathbf{x}}, \mathbf{x}^k$)      ▷ Algorithm 2
11:     if A feasible solution $\hat{\mathbf{x}}$ is found then
12:         gap $\leftarrow \frac{\mathbf{c}^\top(\hat{\mathbf{x}} - \mathbf{x}^k)}{|\mathbf{c}^\top \hat{\mathbf{x}}| + 10^{-10}}$      ▷ Eq. (20)
13:     end if
14:     $k \leftarrow k + 1$
15: end while
16: return $\hat{\mathbf{x}}$
```

5.2 Implementation and hardware

For the numerical comparison, we use a simple implementation of the ESH algorithm utilizing IPOPT 3.12.9 [77] and Gurobi 8.1 [34] as subsolvers for NLP and MILP subproblems. For reading and parsing the MINLP problems, we use the open-source MATLAB toolbox OPTI Toolbox [19]. In the current implementation, we are not able to run the cut strengthening NLP subproblems in parallel, which could significantly speedup the cut strengthening. However, the computational results in the following section still clearly show an advantage of the cut strengthening, both in terms of total computational time and in number of iterations.

The numerical comparisons are performed on a basic desktop computer with an Intel i7-7700k processor, 16 GB RAM, and Windows 10. For the subsolvers, we use default settings except for allowing Gurobi to run on 8 threads. By running Gurobi on multiple threads, the MILP subproblems are solved faster and this is simply done by changing the Threads-parameter. The root-search, in the approximative projection, is done to a tolerance of 10^{-16} in the λ-variable.

6 Numerical results

To test the efficiency of the cut strengthening, we apply the simple implementation of the ESH algorithm, described in Algorithm 3, to a set of test problems. The ESH algorithm forms the baseline for the numerical comparison, and we compare how the cut strengthening techniques affect the number of iterations and solution times. As already mentioned, the results are mainly intended as a proof of concept to show the impact of the cut strengthening. By using techniques such as early stopping [51] and running the cut tightening NLP problems in parallel it would be possible to significantly reduce the solution times.

For the numerical test we have chosen convex MINLP instances from MINLPLib [54] containing at least one exclusive selection constraint. The cut strengthening is mainly intended to strengthen the linear approximation of the nonlinear constraints, and it is expected to be most beneficial for problems containing the big-M formulation of disjunctions containing nonlinear constraints. The disjunctions are identified through the exclusive selection constraints, and the cuts are strengthened through a tighter representation of the disjunctions. If nonlinear disjunctions are represented by the convex hull formulation, our approach will not necessarily be able to tighten the relaxation. For example, if a nonlinear disjunction is represented by the convex hull, then the ESH algorithm can generate supporting hyperplanes to the convex hull of the disjunction and the ST-strategy will not be able to change such cuts. The MT-strategy could still give a tighter approximation for integer feasible solutions, as it may cut off parts of the convex hull for some integer values as shown in Figure 4. Since the cut strengthening is an expensive operation, it is better suited for problems with the big-M formulation as the impact will be more significant and the subproblems are smaller. Therefore, we focus on problems where disjunctions, with either linear or nonlinear constraints, are represented by the big-M formulation. The problems we consider from MINLPLib are different versions of the problems *clay, flay, slay, sssd*, and *tls*. These problems represent optimization tasks such as trimloss problems [35], optimal placement tasks [62] and service systems design [25]. We also consider a problem called *stockcycle* [67], which is known to be difficult to solve without any reformulations [42]. Furthermore, we also consider a class of test problems called *p_ball*, that are described in the Appendix. The *p_ball* instances contain several relatively large nonlinear disjunctions, and are designed to be challenging due to both the nonlinearity and the combinatorial aspects. We use a 2 hour time limit for all the problems except for *stockcycle*, where we use a time limit of 96 hours.

The results are presented in Table 1, showing both the number of iterations and the time needed to solve each problem. The table shows that both cut strengthening techniques can significantly reduce the number of iterations needed to solve the problems. However, the table shows a clear advantage of the multi tightening (MT) strategy. This result aligns well with the theory since cut (13) used in the MT strategy can dominate the cuts used by the single tightening (ST) strategy. On average, the ST strategy reduces the number of iterations by a factor of 1.5, and the MT strategy

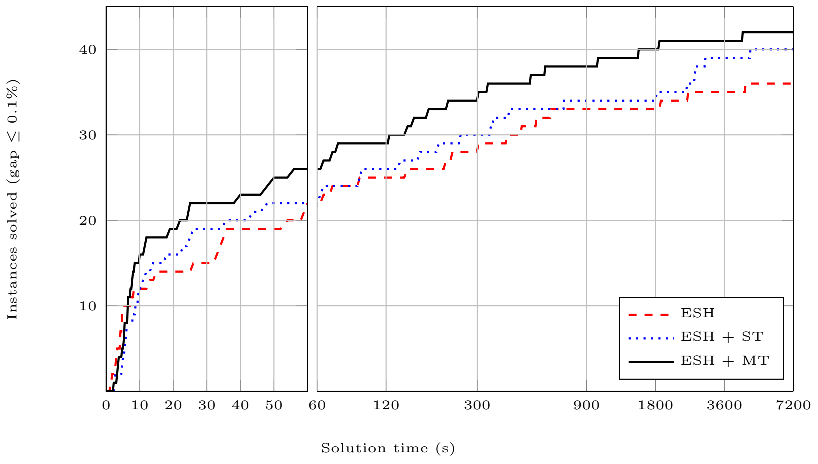


Figure 5: Solution profiles for the ESH algorithm and the ESH algorithm with the cut strengthening techniques. The graphs show the number of instances solved as a function of time. The test set has 43 instances (*clay*, flay*, p_ball*, slay*, sssd*, stockcycle**). An instance is considered solved when it reaches a relative gap of less than 0.1%

gives a further reduction with a factor of 2.9 on average. Both cut strengthening strategies give a significant reduction in solution times, but the MT strategy has a clear advantage and is faster by a factor of 2 compared to the ST strategy. The performance in terms of speed for the strategies is illustrated in Figure 5, which shows the performance profiles of the different strategies. From the figure, it can be observed that MT strategy gives a great advantage for the more challenging problems.

As shown in Table 1, the cut strengthening is especially powerful for the *clay* and *p_ball* problems. These problems contain nonlinear disjunctions that are represented by the big-M formulations, giving weak continuous relaxations that can be efficiently strengthened by the cut strengthening technique. For the *p_ball* problems, the MT cut strengthening reduces the number of iterations by a factor of 11.7 on average. Without the cut strengthening the larger *p_ball* problems are practically intractable with the ESH algorithm, and the optimality gap remained large after 2 hours.

As previously mentioned, the cut tightening comes at a computational cost of solving convex NLP subproblems. For example, problem *p_ball_40b_5p_4d* contains nonlinear disjunctions of size 40, which results in 40 subproblems for the cut tightening per iteration. Solving these subproblems accumulates to about 35% of the total solution time. However, this is well compensated for by the great reduction in the number of iterations.

It is worth mentioning that the cut strengthening techniques do not necessarily result in computationally more demanding iterations. For example, the average iteration time for *slay10m* is 10.6 seconds with the ESH strategy and less than 2 seconds with both the ST and MT strategies. There are two reasons behind the significantly faster iterations. First, the strengthened cuts can result in a tighter continuous relaxation, making the MILP relaxations easier to solve. But more importantly, the cut strengthening procedure can sometimes identify infeasible or non-optimal integer assignments during the solution procedure, see Section 3 for details. For *slay10m*, the cut strengthening is able to fix 53 of the binary variables to zero. Similarly, the cut strengthening eliminates 299 of the 432 binary variables in stockcycle. By further studying these problems, we found that the binary variables fixed by the cut tightening cannot trivially be removed, *e.g.*, by performing LP-based bounds tightening. Some of the integer assignments immediately resulted in problem (NLP-i) to be infeasible in the cut tightening, and some became infeasible due to bounds on the objective and accumulation of strengthened cuts. The ability to identify the infeasible or non-optimal integer assignments can greatly reduce the complexity of the MINLP problems and comes as a desirable side effect of the cut tightening.

	ESH		ESH + ST		ESH + MT	
Instance	Iter.	Time	Iter.	Time	Iter.	Time
clay0203m	48	12.83	29	9.40	**22**	**6.47**
clay0204m	21	4.70	**13**	5.43	17	**3.27**
clay0205m	37	14.41	25	8.71	**21**	**7.01**
clay0303m	81	34.28	37	17.49	**20**	**3.66**
clay0304m	121	90.27	62	46.12	**30**	**11.39**
clay0305m	53	34.08	25	13.16	**21**	**8.41**
flay02m	14	**1.51**	10	2.46	**8**	2.09
flay03m	**28**	**3.14**	**28**	6.01	**28**	6.31
flay04m	**30**	**4.25**	**30**	7.78	**30**	7.76
flay05m	**52**	**53.51**	**52**	61.01	**52**	62.19
flay06m	62	1879.98	62	1801.56	**61**	**1511.69**
p_ball_10b_5p_2d	246	59.45	126	91.84	**31**	**21.56**
p_ball_15b_5p_2d	389	144.72	209	341.44	**51**	**73.12**
p_ball_20b_5p_2d	586	299.42	280	717.67	**34**	**68.96**
p_ball_30b_5p_2d	1026	465.31	328	2677.33	**48**	**303.26**
p_ball_30b_7p_2d	>1039	>7200	417	4665.69	**65**	**1008.56**
p_ball_10b_5p_3d	491	543.35	185	168.69	**60**	**48.04**
p_ball_10b_7p_3d	>433	>7200	299	2952.08	**101**	**593.01**
p_ball_20b_5p_3d	>979	>7200	450	2453.01	**97**	**331.40**
p_ball_30b_5p_3d	>1258	>7200	385	2660.03	**81**	**509.99**
p_ball_40b_5p_3d	>1877	>7200	534	>7200	**112**	**1859.94**
p_ball_10b_5p_4d	879	2496.35	265	410.06	**115**	**122.82**
p_ball_40b_5p_4d	>2134	>7200	640	>7200	**178**	**4311.56**
slay04m	55	**2.79**	**24**	4.79	27	5.15
slay05m	78	**4.90**	**44**	9.51	45	7.73
slay06m	88	**8.08**	45	11.52	**44**	11.05
slay07m	150	25.40	109	22.88	**94**	**18.28**
slay08m	120	33.99	**80**	25.04	89	**24.61**
slay09m	130	68.26	**88**	63.86	104	**47.46**
slay10m	420	4432.49	**105**	**203.14**	109	219.22
sssd08-04	**11**	**4.62**	**11**	5.85	**11**	5.37
sssd12-05	13	**7.56**	13	10.39	**12**	9.81
sssd15-04	**11**	**2.52**	**11**	4.51	**11**	5.41
sssd15-06	21	33.39	**17**	24.72	**17**	**24.53**
sssd15-08	25	626.97	**17**	**134.49**	**17**	146.36
sssd16-07	20	59.91	**17**	**34.73**	**17**	38.96
sssd18-06	17	62.11	15	43.22	**13**	**6.35**
sssd18-08	40	232.48	**25**	**93.29**	30	182.83
sssd20-04	14	**4.03**	**13**	5.53	**13**	4.51
sssd20-08	25	402.08	25	351.03	**21**	**54.03**
stockcycle	>1205	>96h	>3901	>96h	**2910**	**36.07h**
tls2	**8**	**1.01**	**8**	2.66	**8**	3.09
tls4	164	216.33	167	252.42	**126**	**158.88**
Geometric mean	94.08	102.03	63.25	87.07	**38.56**	**44.62**

Table 1: The table shows the solution times in seconds and the number of iterations needed to solve (to a relative gap $\leq 0.1\%$) the MINLP instances with different cut strategies. The strategies include a simple implementation of the ESH algorithm and the ESH algorithm combined with two new cut strengthening techniques. The sign ">" indicates that the time limit was exceeded before the search could be terminated. The fewest iterations and the fastest times are in bold for each problem. All of the instances in this table use a big-M formulation.

	ESH		ESH + ST		ESH + MT	
Instance	Avg. iter.	Avg. time	Avg. iter.	Avg. time	Avg. iter.	Avg. time
rsyn	**9.79**	**4.12**	**9.79**	6.20	**9.79**	6.04
syn	**8.95**	**0.73**	**8.95**	1.47	**8.95**	1.43

Table 2: The table shows the average number of iterations and solution times (to a relative gap $\leq 0.1\%$) for the rsyn and syn problems.

MINLPLib also contains a large number of problems called *syn* and *rsyn* [62, 75]. These problems do have a disjunctive structure, although mainly involving linear constraints. These problems are all easy to solve, and on average they require less than 10 iterations and 3 seconds with the ESH algorithm. There are in total 24 *rsyn* and 24 *syn* problems with the big-M formulation. For these problems, the cut strengthening did not provide any significant advantages. The average times and number of iterations for these two problem types are shown in Table 2.

For the *rsyn* and *syn* instances the cut tightening procedure has little effect on the cuts, and does not result in fewer iterations. In these problems the nonlinear constraints only contain three variables, and there is only a single nonlinear variable in each constraint. It is possible that these constraints are tight to begin with, which would explain why the strengthening does not have much effect for these specific problems.

The cut strengthening seems to be most efficient for problems that contain disjunctions with nonlinear constraints, *e.g., clay* and *p_ball* problems. Some aspects of why the multi tightening strategy works particularly well for these problems are described in the next section. For problems with nonlinear disjunctions, the choice of which exclusive selection constraint to perform the strengthening on is also straight forward since binary variables will be present in the nonlinear constraints. The cut strengthening also performed well on the problems *slay*, *sssd*, and *stockcyckle*, where there are only disjunctions with linear constraints.

6.1 Comparing strong problem formulations and cut strengthening

These results show that the strengthening procedure can give a great advantage for problems where disjunctions are represented by big-M constraints. To further analyze the cut strengthening procedure, we compare the cut strengthening procedure with applying the basic ESH algorithm on the same MINLP instances in a convex hull form, where all or some disjunctions are represented by the convex hull formulation. For this test, we use all problems from the previous section that are available in both a big-M and convex hull form. The results are presented in Table 3. Here we only use the multi strengthening technique, since it results in stronger cuts than the single tightening at the same computational cost.

For nonlinear disjunctions represented by a convex hull formulation, the ESH algorithm can generate supporting hyperplanes to the convex hull of the disjunction. Therefore, applying the ESH algorithm to MINLP instances where nonlinear disjunctions are represented by the convex hull can result in significantly tighter cuts compared to cuts obtained from big-M constraints. This can be seen from the results in Table 3, which shows that the ESH algorithm requires fewer iterations for most of the problems in the convex hull form. The ESH algorithm is still faster on some of the problems in big-M form, which is most likely due to the smaller subproblems.

It is important to notice that the cut strengthening procedure will not necessarily result in similar cuts as applying the ESH algorithm to the convex hull formulation of the problem. This is well illustrated by problem (EX1), where the single tightening strategy does not result in a supporting hyperplane to the convex hull of the disjunction as illustrated in Figure 2. The multi tightening strategy forms a supporting hyperplane to the convex hull of the disjunction, but it still behaves differently compared to a cut obtained by applying the ESH algorithm to the convex hull form of the problem. Figure 4 shows that the multi tightened cut not only forms a supporting hyperplane to the convex hull of the disjunction, but for each feasible integer assignment it also

	ESH		ESH		ESH + MT	
	Big-M		Convex hull		Big-M	
Instance	Iter.	Time	Iter.	Time	Iter.	Time
clay0203(m/h)	48	12.83	46	9.52	**22**	**6.47**
clay0204(m/h)	21	4.70	**17**	6.44	**17**	**3.27**
clay0205(m/h)	37	14.41	**21**	28.36	**21**	**7.01**
clay0303(m/h)	81	34.28	63	10.07	**20**	**3.66**
clay0304(m/h)	121	90.27	94	30.60	**30**	**11.39**
clay0305(m/h)	53	34.08	41	47.00	**21**	**8.41**
flay02(m/h)	14	**1.51**	8	2.48	**8**	2.09
flay03(m/h)	**28**	**3.14**	**28**	8.32	**28**	6.31
flay04(m/h)	30	**4.25**	**29**	12.61	30	7.76
flay05(m/h)	52	**53.51**	**51**	250.15	52	62.19
flay06(m/h)	62	1879.98	>42	>7200	**61**	**1511.69**
p_ball_10b_5p_2d	246	59.45	49	**11.23**	**31**	21.56
p_ball_15b_5p_2d	389	144.72	80	**26.06**	**51**	73.12
p_ball_20b_5p_2d	586	299.42	95	**28.61**	**34**	68.96
p_ball_30b_5p_2d	1026	465.31	167	**84.97**	**48**	303.26
p_ball_30b_7p_2d	>1039	>7200	165	3981.03	**65**	**1008.56**
p_ball_10b_5p_3d	491	543.35	69	48.35	**60**	**48.04**
p_ball_10b_7p_3d	>433	>7200	**73**	700.48	101	**593.01**
p_ball_20b_5p_3d	>979	>7200	157	1073.71	**97**	**331.40**
p_ball_30b_5p_3d	>1258	>7200	194	1921.30	**81**	**509.99**
p_ball_40b_5p_3d	>1877	>7200	>269	>7200	**112**	**1859.94**
p_ball_10b_5p_4d	879	2496.35	**103**	212.49	115	**122.82**
p_ball_40b_5p_4d	>2134	>7200	>210	>7200	**178**	**4311.56**
slay04(m/h)	55	**2.79**	57	10.39	**27**	5.15
slay05(m/h)	78	**4.90**	78	17.30	**45**	7.73
slay06(m/h)	88	**8.08**	88	30.89	**44**	11.05
slay07(m/h)	150	25.40	150	108.41	**94**	**18.28**
slay08(m/h)	120	33.99	115	294.99	**89**	**24.61**
slay09(m/h)	130	68.26	130	1221.01	**104**	**47.46**
slay10(m/h)	420	4432.49	>62	>7200	**109**	**219.22**
Geometric mean	173.47	126.71	72.21	115.60	**47.48**	**47.17**

Table 3: The table shows the solution times in seconds and the number of iterations needed with the ESH algorithm to solve (to a relative gap $\leq 0.1\%$) the MINLP instances in both the big-M and convex hull form. The table also shows the solution times and number of iterations with the ESH algorithm combined with multi tightening strategy applied to the big-M formulation. The sign ">" indicates that the time limit was exceeded before the search could be terminated. The fewest iterations and the fastest times are in bold for each problem.

forms a supporting hyperplane to the corresponding term of the disjunction. For problem (EX1), a single multi tightened cut effectively acts as a supporting hyperplane to three different nonlinear constraints for each feasible integer assignments. The multi tightened cuts behave similarly for the *p_ball* instances, where each disjunction corresponds to assigning a point to one of the balls. For a feasible integer assignment, a cut obtained by multi tightening will then act as a supporting hyperplane to each ball for one of the points. For example, for the problem *p_ball_40b_5p_3d* a multi tightened cut effectively behaves as a tight cut for 40 different nonlinear constraints. This behaviour can make the multi tightened cuts especially powerful for problems with nonlinear disjunctions, which is also shown by the results in Table 3.

Only the *p_ball* and *clay* instances contain nonlinear disjunctions and for most of these problems the multi tightening strategy significantly reduces both solution times and number of iterations. For problems with only linear disjunctions, the multi tightening strategy does not necessarily give the same advantage. However, the multi tightening strategy also performed well on the test problems with only linear disjunctions. On average the multi tightening strategy reduces the number of iterations by a factor of 7.2 compared to the ESH algorithm with the big-M formulation and by a factor of 1.5 compared to ESH algorithm with the convex hull formulation of the problems. In terms of total solution time, the multi tightening strategy reduces the total solution time by more than a factor of 3 on average compared to the other two approaches.

7 Conclusions

In this paper, we have presented a new framework for strengthening cuts to obtain tighter outer approximations for convex MINLP. The cut strengthening is based on analyzing disjunctive structures in the MINLP problem, and either strengthen the cut for the entire disjunction or separately for each term of the disjunction. We have proven that the strengthening results in valid cuts that can dominate the original cut. The numerical results show that the strengthening can greatly reduce the number of iterations and time needed to solve convex MINLP problems. We have focused on strengthening cuts derived from the ESH algorithm, but the same techniques can just as well be used to strengthen cuts obtained by OA, ECP or generalized Benders decomposition.

Acknowledgements

The work was funded by a Newton International Fellowship by the Royal Society (NIF\R1\182194) to JK, a grant by the Swedish Cultural Foundation in Finland to JK, and by an Engineering & Physical Sciences Research Council Research Fellowship to RM [grant number EP/P016871/1].

References

[1] Achterberg T, Wunderling R (2013) Mixed integer programming: Analyzing 12 years of progress. In: Facets of Combinatorial Optimization, Springer, pp 449–481

[2] Atamtürk A, Narayanan V (2010) Conic mixed-integer rounding cuts. Mathematical Programming 122(1):1–20

[3] Balas E (1979) Disjunctive programming. In: Annals of Discrete Mathematics, vol 5, Elsevier, pp 3–51

[4] Balas E, Margot F (2013) Generalized intersection cuts and a new cut generating paradigm. Mathematical Programming 137(1-2):19–35

[5] Balas E, Ceria S, Cornuéjols G (1993) A lift-and-project cutting plane algorithm for mixed 0–1 programs. Mathematical Programming 58(1-3):295–324

[6] Ben-Tal A, Nemirovski A (2001) Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications, vol 2. Siam

[7] Bernal DE, Vigerske S, Trespalacios F, Grossmann IE (2020) Improving the performance of DICOPT in convex MINLP problems using a feasibility pump. Optimization Methods and Software 35(1):171–190

[8] Berthold T (2014) Heuristic algorithms in global MINLP solvers. PhD thesis, Technische Universität Berlin

[9] Bienstock D (1996) Computational study of a family of mixed-integer quadratic programming problems. Mathematical Programming 74(2):121–140

[10] Bonami P (2011) Lift-and-project cuts for mixed integer convex programs. In: International Conference on Integer Programming and Combinatorial Optimization, Springer, pp 52–64

[11] Bonami P, Gonçalves JP (2012) Heuristics for convex mixed integer nonlinear programs. Computational Optimization and Applications 51(2):729–747

[12] Bonami P, Lejeune MA (2009) An exact solution approach for portfolio optimization problems under stochastic and integer constraints. Operations Research 57(3):650–670

[13] Bonami P, Biegler LT, Conn AR, Cornuéjols G, Grossmann IE, Laird CD, Lee J, Lodi A, Margot F, Sawaya N, Wächter A (2008) An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optimization 5(2):186–204

[14] Bonami P, Cornuéjols G, Lodi A, Margot F (2009) A feasibility pump for mixed integer nonlinear programs. Mathematical Programming 119(2):331–352

[15] Bonami P, Kilinç M, Linderoth J (2012) Algorithms and software for convex mixed integer nonlinear programs. In: Mixed integer nonlinear programming, Springer, pp 1–39

[16] Boukouvala F, Misener R, Floudas CA (2016) Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. European Journal of Operational Research 252(3):701–727

[17] Castillo I, Westerlund J, Emet S, Westerlund T (2005) Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods. Computers & Chemical Engineering 30(1):54–69

[18] Ceria S, Soares J (1999) Convex programming for disjunctive convex optimization. Mathematical Programming 86(3):595–614

[19] Currie J, Wilson DI (2012) OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User. In: Sahinidis N, Pinto J (eds) Foundations of Computer-Aided Process Operations, Savannah, Georgia, USA

[20] D'Ambrosio C, Frangioni A, Liberti L, Lodi A (2012) A storm of feasibility pumps for nonconvex MINLP. Mathematical Programming 136(2):375–402

[21] Duran M, Grossmann I (1986) A mixed-integer nonlinear programming algorithm for process systems synthesis. AIChE Journal 32(4):592–606

[22] Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming 36(3):307–339

[23] Duran MA, Grossmann IE (1986) Simultaneous optimization and heat integration of chemical processes. AIChE Journal 32(1):123–138

[24] Durán-Peña MA (1984) A mixed-integer nonlinear programming approach for the systematic synthesis of engineering systems. PhD thesis, Carnegie-Mellon University

[25] Elhedhli S (2006) Service system design with immobile servers, stochastic demand, and congestion. Manufacturing & Service Operations Management 8(1):92–97

[26] Eronen VP, Kronqvist J, Westerlund T, Mäkelä MM, Karmitsa N (2017) Method for solving generalized convex nonsmooth mixed-integer nonlinear programming problems. Journal of Global Optimization 69(2):443–459

[27] Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Mathematical Programming 66(1):327–349

[28] Frangioni A, Gentile C (2006) Perspective cuts for a class of convex 0–1 mixed integer programs. Mathematical Programming 106(2):225–236

[29] Geißler B, Morsi A, Schewe L, Schmidt M (2015) Solving power-constrained gas transportation problems using an MIP-based alternating direction method. Computers & Chemical Engineering 82:303 – 317

[30] Geoffrion AM (1972) Generalized Benders decomposition. Journal of Optimization Theory and Applications 10(4):237–260

[31] Gomory R (1960) An algorithm for the mixed integer problem. Tech. rep., RAND Corp. Santa Monica, CA

[32] Grossmann IE, Viswanathan J, Vecchietti A, Raman R, Kalvelagen E, et al (2002) Gams/dicopt: A discrete continuous optimization package. GAMS Corporation Inc

[33] Gupta OK, Ravindran A (1985) Branch and bound experiments in convex nonlinear integer programming. Management Science 31(12):1533–1546

[34] Gurobi (2019) Gurobi optimizer reference manual. Gurobi Optimization, LLC, URL `https://www.gurobi.com/documentation/8.1/refman/index.html`

[35] Harjunkoski I, Westerlund T, Pörn R, Skrifvars H (1998) Different transformations for solving non-convex trim-loss problems by MINLP. European Journal of Operational Research 105(3):594–603

[36] Hijazi H, Bonami P, Ouorou A (2013) An outer-inner approximation for separable mixed-integer nonlinear programs. INFORMS Journal on Computing 26(1):31–44

[37] Kılınç MR, Linderoth J, Luedtke J (2017) Lift-and-project cuts for convex mixed integer nonlinear programs. Mathematical Programming Computation 9(4):499–526

[38] Kröger O, Coffrin C, Hijazi H, Nagarajan H (2018) Juniper: An Open-Source Nonlinear Branch-and-Bound Solver in Julia. arXiv preprint: 1804.07332

[39] Kronqvist J, Lundell A, Westerlund T (2016) The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. Journal of Global Optimization 64(2):249–272

[40] Kronqvist J, Bernal D, Lundell A, Westerlund T (2018) A center-cut algorithm for quickly obtaining feasible solutions and solving convex MINLP problems. Computers & Chemical Engineering

[41] Kronqvist J, Bernal DE, Grossmann IE (2018) Using regularization and second order information in outer approximation for convex MINLP. Mathematical Programming pp 1–26

[42] Kronqvist J, Lundell A, Westerlund T (2018) Reformulations for utilizing separability when solving convex MINLP problems. Journal of Global Optimization pp 1–22

[43] Kronqvist J, Bernal DE, Lundell A, Grossmann IE (2019) A review and comparison of solvers for convex MINLP. Optimization and Engineering 20(2):397–455

[44] Lastusilta T (2011) GAMS MINLP solver comparisons and some improvements to the AlphaECP algorithm. PhD thesis, Åbo Akademi University

[45] Lee S, Grossmann IE (2000) New algorithms for nonlinear generalized disjunctive programming. Computers & Chemical Engineering 24(9-10):2125–2141

[46] Linderoth JT, Lodi A (2011) MILP Software. doi:10.1002/9780470400531.eorms0524

[47] Lodi A, Tanneau M, Vielma JP (2019) Disjunctive cuts for mixed-integer conic optimization. arXiv preprint arXiv:191203166

[48] Lundell A, Westerlund T (2017) Solving global optimization problems using reformulations and signomial transformations. Computers & Chemical Engineering (available online)

[49] Lundell A, Kronqvist J, Westerlund T (2016) Improvements to the Supporting Hyperplane Optimization Toolkit Solver for Convex MINLP. In: XIII GLOBAL OPTIMIZATION WORKSHOP GOW'16, vol 16, pp 101–104

[50] Lundell A, Kronqvist J, Westerlund T (2017) SHOT – A global solver for convex MINLP in Wolfram Mathematica. In: Computer Aided Chemical Engineering, vol 40, Elsevier, pp 2137–2142

[51] Lundell A, Kronqvist J, Westerlund T (2018) The supporting hyperplane optimization toolkit – a polyhedral outer approximation based convex MINLP solver utilizing a single branching tree approach. Preprint, Optimization Online URL `http://www.optimization-online.org/DB_HTML/2018/06/6680.html`

[52] Mahajan A, Leyffer S, Linderoth J, Luedtke J, Munson T (2017) Minotaur: A Mixed-Integer Nonlinear Optimization Toolkit. Preprint, Optimization Online URL `http://www.optimization-online.org/DB_FILE/2017/10/6275.pdf`

[53] Melo W, Fampa M, Raupp F (2018) An overview of MINLP algorithms and their implementation in Muriqui Optimizer. Annals of Operations Research pp 1–25

[54] MINLPLib (2020) Mixed-integer nonlinear programming library. URL `http://www.minlplib.org/`, [Accessed 01-January-2020]

[55] Misener R, Floudas CA (2009) Advances for the pooling problem: Modeling, global optimization, and computational studies. Applied and Computational Mathematics 8(1):3–22

[56] Misener R, Smadbeck JB, Floudas CA (2015) Dynamically generated cutting planes for mixed-integer quadratically constrained quadratic programs and their incorporation into GloMIQO 2. Optimization Methods and Software 30(1):215–249

[57] Modaresi S, Kılınç MR, Vielma JP (2015) Split cuts and extended formulations for mixed integer conic quadratic programming. Operations Research Letters 43(1):10–15

[58] Muts P, Nowak I, Hendrix EM (2020) The decomposition-based outer approximation algorithm for convex mixed-integer nonlinear programming. Journal of Global Optimization pp 1–22

[59] Nowak I, Breitfeld N, Hendrix EM, Njacheun-Njanzoua G (2018) Decomposition-based inner- and outer-refinement algorithms for global optimization. Journal of Global Optimization 72(2):305–321

[60] Quesada I, Grossmann IE (1992) An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Computers & Chemical Engineering 16(10-11):937–947

[61] Raman R, Grossmann IE (1994) Modelling and computational techniques for logic based integer programming. Computers & Chemical Engineering 18(7):563–578

[62] Sawaya N (2006) Reformulations, relaxations and cutting planes for generalized disjunctive programming. PhD thesis, Carnegie Mellon University

[63] Sawaya NW, Grossmann IE (2007) Computational implementation of non-linear convex hull reformulation. Computers & Chemical Engineering 31(7):856–866

[64] Serra T (2020) Reformulating the disjunctive cut generating linear program. Annals of Operations Research pp 1–22

[65] Serrano F, Schwarz R, Gleixner A (2019) On the relation between the extended supporting hyperplane algorithm and Kelley's cutting plane algorithm. arXiv preprint arXiv:190508157

[66] Sherali HD, Adams WP (2013) A reformulation-linearization technique for solving discrete and continuous nonconvex problems, vol 31. Springer Science & Business Media

[67] Silver E, Moon I (1999) A fast heuristic for minimising total average cycle stock subject to practical constraints. Journal of the Operational Research Society 50(8):789–796

[68] Slater M (1950) Lagrange multipliers revisited. Tech. rep., Cowles Foundation for Research in Economics, Yale University

[69] Stubbs RA, Mehrotra S (1999) A branch-and-cut method for 0-1 mixed convex programming. Mathematical Programming 86(3):515–532

[70] Su L, Tang L, Grossmann IE (2015) Computational strategies for improved minlp algorithms. Computers & Chemical Engineering 75:40–48

[71] Su L, Tang L, Bernal DE, Grossmann IE (2018) Improved quadratic cuts for convex mixed-integer nonlinear programs. Computers & Chemical Engineering 109:77–95

[72] Trespalacios F, Grossmann IE (2014) Review of mixed-integer nonlinear and generalized disjunctive programming methods. Chemie Ingenieur Technik 86(7):991–1012

[73] Trespalacios F, Grossmann IE (2015) Improved Big-M reformulation for generalized disjunctive programs. Computers & Chemical Engineering 76:98–103

[74] Trespalacios F, Grossmann IE (2016) Cutting plane algorithm for convex generalized disjunctive programs. INFORMS Journal on Computing 28(2):209–222

[75] Türkay M, Grossmann IE (1996) Logic-based MINLP algorithms for the optimal synthesis of process networks. Computers & Chemical Engineering 20(8):959–978

[76] Veinott Jr AF (1967) The supporting hyperplane method for unimodal programming. Operations Research 15(1):147–152

[77] Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming 106(1):25–57

[78] Westerlund T, Petterson F (1995) An extended cutting plane method for solving convex MINLP problems. Computers & Chemical Engineering 19:131–136

[79] Westerlund T, Pörn R (2002) Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. Optimization and Engineering 3(3):253–280

[80] Zhu Y, Kuno T (2006) A disjunctive cutting-plane-based branch-and-cut algorithm for 0-1 mixed-integer convex nonlinear programs. Industrial & Engineering Chemistry Research 45(1):187–196

Appendices

A New nonlinear disjunctive test problems

To further test the cut strengthening for problems with different sized disjunctions containing nonlinearities of varying difficulty, we have generated 12 new test problems. The underlying optimization task is simple, select n-points in m-dimensional balls, such that the ℓ_1-distance between all points is minimized. Only one point can be assigned to each ball, and in total there are l balls with radius one. The problem has a clear disjunctive programming structure, where the disjunctions arise from the assignment of each point to one of the balls. In total we get n disjunctions of size l, *i.e.,* one disjunction per point and one disjunctive term per ball.

Even if this optimization task can be represented as a binary quadratic problem, it is a challenging problem for OA-type algorithms. Without any reformulations, OA-type algorithms will require a large number of iterations due to the difficulties of accurately approximating an n-dimensional ball with hyperplanes [36]. Higher-dimensional balls render the outer approximation task more difficult, and the number of nonlinear constraints is given by the number of balls times the number of points. There is a clear combinatorial structure to the problem, and the complexity increases with the number of points and balls as the number of possible discrete configurations drastically increases. The seemingly simple optimization problem is, thus, challenging both due to the combinatorial nature and the nonlinearity.

Before presenting the MINLP formulation, we briefly describe the notation and some details of the problem formulation. Here, $\mathbf{c}^i \in \mathbb{R}^m$ denotes the center of ball i and c_1^i refers to the first coordinate of the center. Similarly, $\mathbf{p}^i \in \mathbb{R}^m$ refers to point i and p_1^i is the first coordinate of the point. To simplify the notation, we introduce the sets $D = \{1, 2, \dots, m\}$, $P = \{1, 2, \dots, n\}$, $P^i = \{i+1, i+2, \dots, n\}$, and $B = \{1, 2, \dots, l\}$. The ℓ_1-distance can be represented by linear constraints by introducing auxiliary variables $\mathbf{d}^{i,j} \in \mathbb{R}^m$. As before, $d_k^{i,j}$ refers to the k-th component of the vector $\mathbf{d}^{i,j}$. To act as the absolute value, we use the following constraints

$$\begin{aligned} d_k^{i,j} &\geq p_k^i - p_k^j \qquad \forall k \in D,\ \forall i \in P,\ \forall j \in P^i, \\ d_k^{i,j} &\geq p_k^j - p_k^i \qquad \forall k \in D,\ \forall i \in P,\ \forall j \in P^i, \end{aligned}$$

and the distance between the points i and j is now given by $\sum_{k=1}^m d_k^{i,j}$. For the test problems, we randomly chose each coordinate of the balls' centers between 0 and 10, which also limits each variable $d_k^{i,j}$ to the interval $[0, 10]$. We use a binary variable $b_{i,r}$ for selecting if point i is assigned to ball r. There are several identical solutions to these optimization problems due to symmetries, *e.g.,* you can switch places of the first and second point to obtain another equally good solution. To eliminate some of the symmetrical solutions we include an ordering of the points along the first coordinate. The ordering is enforced by including the constraints that the first point must be closer to the origin along the first coordinate than the second point, and similarly for the following points. Using the big-M formulation the problem can be written as

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=1}^{m} d_k^{i,j} & & \\
& d_k^{i,j} \geq p_k^i - p_k^j & \forall k \in D,\ \forall i \in P,\ \forall j \in P^i, & \\
& d_k^{i,j} \geq p_k^j - p_k^i & \forall k \in D,\ \forall i \in P,\ \forall j \in P^i, & \\
& \sum_{k=1}^{m} \left(p_k^i - c_k^r\right)^2 \leq 1 + M_r(1 - b_{i,r}) & \forall i \in P,\ \forall r \in B, & \\
& \sum_{r=1}^{l} b_{i,r} = 1 & \forall i \in P, & \qquad (21) \\
& \sum_{i=1}^{n} b_{i,r} \leq 1 & \forall r \in B, & \\
& p_1^i \leq p_1^{i+1} & \forall i \in P \setminus n, & \\
& b_{i,r} \in \{0, 1\} & \forall i \in P,\ \forall r \in B, & \\
& \mathbf{d}^{i,j} \in [0, 10]^m & \forall i \in P,\ \forall j \in P^i, & \\
& \mathbf{p}^i \in [0, 10]^m & \forall i \in P, &
\end{aligned}
$$

where M_r are sufficiently large constants. For these problems the smallest valid M_r is simply given by

$$M_r = \max_{i \in B} \left\{ \left(\left\| \mathbf{c}^r - \mathbf{c}^i \right\|_2 + 1 \right)^2 - 1 \right\}. \qquad (22)$$

Here, M_r are based on the largest squared Euclidean distance between the center of a ball and the point furthest away in any other ball. This gives the smallest valid M constants, resulting in a tight Big-M formulation. It could be possible to obtain a stronger formulation, *e.g.,* by eliminating furhter symmetries or by the techniques presented by [73]. However, the goal here is not to derive an optimal problem formulation, but simply to generate a few test problems of different size and difficulty.

We have generated 12 random test instances, where the centers of the unit balls are chosen randomly. The test problems are of different size, and the main attributes are summarized in Table (4). The problems range in size from 50-210 binary variables and 30-85 continuous variables.

It is also possible to represent the assignment of points to circles with the convex hull formulation to obtain a tighter continuous relaxation. For each point $i \in P$, we need to make l copies of the variables $\mathbf{p}^i$, and we get the new variables $\nu_r^{\mathbf{p}^i} \in \mathbb{R}^m$. It would be possible to represent the convex hull by second-order cones, but to fit within the framework of this paper we use the formulation presented by [63]. The problem can then be written as

Name	Bin. vars.	Cont. vars.	Nl. cons.	Balls	Points	B. dim.
p_ball_10b_5p_2d	50	30	50	10	5	2
p_ball_15b_5p_2d	75	30	75	15	5	2
p_ball_20b_5p_2d	100	30	100	20	5	2
p_ball_30b_5p_2d	150	30	150	30	5	2
p_ball_30b_7p_2d	210	56	210	30	7	2
p_ball_10b_5p_3d	50	45	50	10	5	3
p_ball_10b_7p_3d	70	85	70	10	7	3
p_ball_20b_5p_3d	100	45	100	20	5	3
p_ball_30b_5p_3d	150	45	150	30	5	3
p_ball_40b_5p_3d	200	45	200	40	5	3
p_ball_10b_5p_4d	50	60	50	10	5	4
p_ball_40b_5p_4d	200	60	200	40	5	4

Table 4: The table shows the main properties of the test problems with the Big-M formulation. For each problem, the table lists the number of binary variables (Bin. vars.), number of continuous variables (Cont. vars.), number of nonlinear constraints (Nl. cons.), number of balls, number of points, and the dimensionality of the unit balls (B. dim.).

$$
\begin{aligned}
\min\quad & \sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=1}^{m} d_k^{i,j} && \\
& d_k^{i,j} \geq p_k^i - p_k^j && \forall k \in D,\ \forall i \in P,\ \forall j \in P^i, \\
& d_k^{i,j} \geq p_k^j - p_k^i && \forall k \in D,\ \forall i \in P,\ \forall j \in P^i, \\
& ((1-\epsilon)b_{i,r}+\epsilon)\left(\left\|\frac{\nu_r^{\mathbf{p}^i}}{(1-\epsilon)b_{i,r}+\epsilon} - c^r\right\|_2^2 - 1\right) && \\
& -\epsilon\left(\|c^r\|_2^2 - 1\right)(1-b_{i,r}) \leq 0 && \forall i \in P,\ \forall r \in B, \\
& \sum_{r=1}^{l} b_{i,r} = 1 && \forall i \in P, \\
& \sum_{i=1}^{n} b_{i,r} \leq 1 && \forall r \in B, \\
& p_1^i \leq p_1^{i+1} && \forall i \in P \setminus n, \\
& \sum_{r=1}^{l} \nu_r^{\mathbf{p}^i} = \mathbf{p}^i && \forall i \in P \\
& \mathbf{0} \leq \nu_r^{\mathbf{p}^i} \leq \mathbf{10}\cdot b_{i,r} && \forall i \in P, \forall r \in B, \\
& b_{i,r} \in \{0,1\} && \forall i \in P,\ \forall r \in B, \\
& \mathbf{d}^{i,j} \in [0,10]^m && \forall i \in P,\ \forall j \in P^i, \\
& \mathbf{p}^i \in [0,10]^m && \forall i \in P, \\
& \nu_r^{\mathbf{p}^i} \in [0,10]^m && \forall i \in P,\ \forall r \in B.
\end{aligned}
\tag{23}
$$

In the numerical tests we use $\epsilon = 10^{-9}$ which was stable for the subsolvers. The main properties of the test problems with the convex hull formulation are summarized in Table 5. The problems range in size from 50-210 binary variables and 130-860 continuous variables.

Numerical results are presented in Section 6, which show that the test problems are challenging for the ESH algorithm with both problem formulations. Finally, all the test problems can be downloaded from `https://github.com/jkronqvi/points_in_circles`.

Name	Bin. vars.	Cont. vars.	Nl. cons.	Balls	Points	B. dim.
p_ball_10b_5p_2d	50	130	50	10	5	2
p_ball_15b_5p_2d	75	180	75	15	5	2
p_ball_20b_5p_2d	100	230	100	20	5	2
p_ball_30b_5p_2d	150	330	150	30	5	2
p_ball_30b_7p_2d	210	476	210	30	7	2
p_ball_10b_5p_3d	50	195	50	10	5	3
p_ball_10b_7p_3d	70	295	70	10	7	3
p_ball_20b_5p_3d	100	345	100	20	5	3
p_ball_30b_5p_3d	150	495	150	30	5	3
p_ball_40b_5p_3d	200	645	200	40	5	3
p_ball_10b_5p_4d	50	260	50	10	5	4
p_ball_40b_5p_4d	200	860	200	40	5	4

Table 5: The table shows the main properties of the test problems with the convex hull formulation. For each problem, the table lists the number of binary variables (Bin. vars.), number of continuous variables (Cont. vars.), number of nonlinear constraints (Nl. cons.), number of balls, number of points, and the dimensionality of the unit balls (B. dim.).