A Center-Cut Algorithm for Quickly Obtaining Feasible Solutions and Solving Convex MINLP Problems

Jan Kronqvist[a], David E. Bernal[b], Andreas Lundell[a], and Tapio Westerlund[a]

[a] Faculty of Science and Engineering Åbo Akademi University, Turku, Finland
[b]Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh PA, USA

February, 2018

Abstract

Here we present a center-cut algorithm for convex mixed-integer nonlinear programming (MINLP) that can either be used as a primal heuristic or as a deterministic solution technique. Like many other algorithms for convex MINLP, the center-cut algorithm constructs a linear approximation of the original problem. The main idea of the algorithm is to use the linear approximation differently in order to find feasible solutions within only a few iterations. The algorithm chooses trial solutions as the center of the current linear outer approximation of the nonlinear constraints, making the trial solutions more likely to satisfy the constraints. The ability to find feasible solutions within only a few iterations makes the algorithm well suited as a primal heuristic, and we prove that the algorithm finds the optimal solution within a finite number of iterations. Numerical results show that the algorithm obtains feasible solutions quickly and is able to obtain good solutions.

1 Introduction

Many optimization problems arising in engineering and science contain both some form of distinct decision making and nonlinear correlations. Mixed-integer nonlinear programming (MINLP) deals with such optimization problems by combining the modeling capabilities of mixed-integer linear programming (MILP) and nonlinear programming (NLP) into a powerful modeling framework. The integer variables make it possible to incorporate discrete decisions and logical relations in the optimization problems, and the combination of linear and nonlinear functions make it possible to accurately describe a variety of different phenomena. The ability to accurately model real-world problems has made MINLP an active research area and there exists a vast number of applications in fields such as engineering, computational chemistry, and finance, *e.g.,* see Biegler & Grossmann (2004) and Floudas (2000).

MINLP problems are often classified as either convex or non-convex based on properties of the nonlinear functions (Lee & Leyffer, 2011). Here, we focus on convex MINLP problems, and the convex properties are exploited in the algorithm presented later on. Most deterministic methods for solving convex MINLP problems are based on some kind of decomposition technique, where the optimal solution is obtained by solving a sequence of tractable subproblems. Such methods are, *e.g.,* extended cutting plane (ECP) (Westerlund & Petterson, 1995), extended supporting hyperplane (ESH) (Kronqvist et al., 2016), generalized Benders decomposition (GBD) (Geoffrion, 1972), outer approximation (OA) (Duran & Grossmann, 1986) and branch and bound (BB) techniques (Dakin, 1965). Even if there are several methods available for solving convex MINLP problems, these are still challenging as shown in the solver comparison by Kronqvist et al. (2016). Thus, further research in the field is still motivated.

In recent years there has been a growing interest in so-called primal heuristics, *i.e.,* algorithms intended to quickly obtain good feasible solutions to an optimization problem. Such algorithms are useful not only since they can provide a good feasible solution, but knowing a feasible solution can also greatly improve the performance of solvers, *e.g.,* Fischetti & Lodi (2011) claimed that primal heuristics were one of the most crucial improvements for MILP in the last decade. Different primal heuristics have also been proposed for MINLP problems, *e.g.,* undercover (Berthold & Gleixner, 2014) and feasibility pump (Bonami et al., 2009). A review of several primal heuristics for MINLP are given by Berthold (2014).

Primal heuristics can be a valuable tool, especially for difficult MINLP problems, since it may be the only way to obtain a good solution, and for some applications such as real-time optimization, it may be of utter importance to quickly obtain a feasible solution. Knowing a feasible solution can also improve the performance of MINLP solvers as shown in Berthold (2014) and Bernal et al. (2017). A good feasible solution can significantly reduce the search tree in branch and bound, and in solvers based on ECP or ESH, it provides an upper bound on the objective enabling the use of optimality gap as stopping criterion. It is also possible to solve pseudo convex MINLP problems as a sequence of feasibility problems as in the GAECP algorithm (Westerlund & Pörn, 2002).

In this paper, we describe a new center-cut algorithm for convex MINLP problems, that can either be used as a primal heuristic or as deterministic solution technique. The algorithm was first presented briefly in a conference paper from the ESCAPE27 conference (Kronqvist et al., 2017), and here we continue with more details and a rigorous proof that the algorithm will obtain the optimal solution in a finite number of iterations. Like OA, ECP or ESH, the center-cut algorithm also constructs a polyhedral approximation of the feasible region defined by the nonlinear constraints. However, here we use the polyhedral approximation differently, in a way that will enable us to find feasible solutions within only a few iterations. The main idea of the algorithm is to choose the trial solutions in the center of the polyhedral approximation, instead of choosing the trial solutions on the boundary of the polyhedral approximation as in ECP and ESH. A similar concept for solving NLP problems was proposed by Elzinga & Moore (1975). The algorithm should be well suited as a primal heuristic, but it can also be used as a stand-alone solution technique with guaranteed convergence.

2 Background

A convex MINLP problem can be defined compactly as

$$\min_{\mathbf{x},\mathbf{y}\in N\cap L\cap Y} \mathbf{c}_1^T\mathbf{x}+\mathbf{c}_2^T\mathbf{y} \tag{P-MINLP}$$

where the sets N, L and Y are given by

$$\begin{aligned} &N=\{(\mathbf{x},\mathbf{y})\in\mathbb{R}^n\times\mathbb{R}^m \mid g_k(\mathbf{x},\mathbf{y})\leq 0 \quad \forall k=1,\dots l\},\\ &L=\{(\mathbf{x},\mathbf{y})\in\mathbb{R}^n\times\mathbb{R}^m \mid A\mathbf{x}+B\mathbf{y}\leq \mathbf{b}\},\\ &Y=\{\mathbf{y}\in\mathbb{Z}^m\}. \end{aligned} \tag{1}$$

In Eq. (1) A and B are matrices defining the linear constraints, including variable bounds. Throughout this paper we consider the following assumptions to be true:

Assumption 1. The nonlinear functions $g_1,\dots,g_l$ are convex and continuously differentiable.

Assumption 2. The intersection $L\cap Y$ defines a compact nonempty set, *i.e.,* all variables must be bounded.

Assumption 3. By fixing the integer variables in the MINLP problem to a feasible integer combination $\mathbf{y}$, the resulting NLP problem satisfies Slater's condition, see Slater et al. (1959).

These assumptions are needed in order to rigorously prove that the algorithm converges to the optimal solution in a finite number of iterations. Similar conditions are also needed to guarantee convergence of OA and ESH, see Fletcher & Leyffer (1994) and Kronqvist et al. (2016). It should be possible to handle MINLP problems with nonsmooth convex functions with the algorithm, although such problems are not considered here.

One of the key elements in both ECP, ESH, and OA is to construct an iteratively improving polyhedral approximation of the set N. The approximation is obtained by first-order Taylor series expansions of the nonlinear constraints generated at the trial solutions; at iteration i it is given by

$$\hat{N}_i=\left\{g_k(\mathbf{x}^j,\mathbf{y}^j)+\nabla g_k(\mathbf{x}^j,\mathbf{y}^j)^T\begin{bmatrix}\mathbf{x}-\mathbf{x}^j\\ \mathbf{y}-\mathbf{y}^j\end{bmatrix}\leq 0, j=1,\dots i,\ k\in K_j\right\}, \tag{2}$$

where K_j contains the indexes of all nonlinear constraints active at the trial solution $(\mathbf{x}^j, \mathbf{y}^j)$, *i.e.,* all nonlinear constraints such that $g_k(\mathbf{x}^j, \mathbf{y}^j) \geq 0$. Due to convexity, we know that the polyhedral approximation will overestimate the set N and every point within N is also a point within $\hat{N}_i$, *i.e.,* $N \subset \hat{N}_i$.

The standard approach for using the polyhedral approximation is to simply replace the set N by $\hat{N}_i$ in problem (P-MINLP). The next trial solution can then be obtained by solving the following MILP problem

$$\left(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}\right) \in \underset{(\mathbf{x},\mathbf{y}) \in \hat{N}_i \cap L \cap Y}{\arg\min} \quad \mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y}. \tag{3}$$

Both ECP and ESH choose the trial solutions by solving problem (3), and OA selects the integer combination by the same approach. However, if we choose the trial solutions by solving problem (3), then we will not obtain a feasible solution before the very last iteration, see *e.g.,* Kronqvist et al. (2016). By this approach, the trial solutions tend to be selected as points on the boundary of the set $\hat{N}_i$.

In the center-cut algorithm we will use the polyhedral approximation differently, instead of choosing points on the boundary we will select the trial solutions as points in the center of the polyhedral approximation. Since we know that the feasible set defined by the nonlinear constraints is contained somewhere in $\hat{N}_i$, it seems natural to search for a feasible solution in the center of the set.

3 The center-cut algoritm

As previously mentioned, the main idea of the center cut algorithm is to choose the trial solutions as the center of the polyhedral approximation of the feasible set defined by the nonlinear constraints. There are several definitions of the center of a set, and here we will use the Chebyshev center. The Chebyshev center is defined as the point furthest away from the boundary in all directions, which is also the center of the largest n-dimensional ball inscribed in the set (Boyd & Vandenberghe, 2004). Since the set $\hat{N}_i$ is a polyhedral set defined by linear inequality constraints, we can find the Chebyshev center of the set simply by solving the following linear programming (LP) problem

$$\begin{aligned}
&\max_{\mathbf{x},\mathbf{y},r} \ r \\
&\text{s.t.} \\
&g_k(\mathbf{x}^j, \mathbf{y}^j) + \nabla g_k(\mathbf{x}^j, \mathbf{y}^j)^T \begin{bmatrix} \mathbf{x} - \mathbf{x}^j \\ \mathbf{y} - \mathbf{y}^j \end{bmatrix} + r \left\| \nabla g_k(\mathbf{x}^j, \mathbf{y}^j) \right\|_2 \leq 0, \\
&\qquad\qquad\qquad\qquad\qquad\qquad j = 1, \ldots i, \ k \in K_j, \\
&\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m, r \in \mathbb{R},
\end{aligned} \tag{4}$$

where r is the radius of the inscribed ball. For more details on how to find the Chebyshev center of a polyhedral set, see chapter 8.5 in Boyd & Vandenberghe (2004). To simplify notation we introduce a new set B_i defined as

$$\begin{aligned}
B_i = \Big\{ &g_k(\mathbf{x}^j, \mathbf{y}^j) + \nabla g_k(\mathbf{x}^j, \mathbf{y}^j)^T \begin{bmatrix} \mathbf{x} - \mathbf{x}^j \\ \mathbf{y} - \mathbf{y}^j \end{bmatrix} + r \left\| \nabla g_k(\mathbf{x}^j, \mathbf{y}^j) \right\|_2 \leq 0, \\
&\qquad\qquad\qquad\qquad\qquad\qquad j = 1, \ldots i, \ k \in K_j \ \Big\},
\end{aligned} \tag{5}$$

which contains all constraints defining the set $\hat{N}_i$. In order to obtain a feasible solution to the MINLP problem, we also have to take the linear constraints and integer requirements into consideration. A new trial solution will, therefore, be chosen as the center of the largest ball inscribed in the set $\hat{N}_i$, with the restrictions that the center has to satisfy all linear constraints and integer requirements. The linear constraints and integer restrictions therefore only affect the location of the center, and not directly the radius of the ball. A new trial solution is, thus, obtained by solving the following MILP problem

$$(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}, r^{i+1}) \in \underset{(\mathbf{x},\mathbf{y},r) \in B_i \cap L \cap Y}{\arg\max} \quad r. \tag{MILP-i}$$

Since we are maximizing the radius of the ball inscribed in $\hat{N}_i$, it results in a trial solution minimizing the left-hand side of the linearized constraints in Eq. (2). Once we have obtain a new trial solution $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ there are two possibilities: either it violates some of the nonlinear constraints or it is a feasible solution.

In case the trial solution $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ violates some of the nonlinear constraints, then we can improve the polyhedral approximation by generating cutting planes according to

$$g_k\left(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}\right) + \nabla g_k\left(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}\right)^T \begin{bmatrix} \mathbf{x} - \mathbf{x}^{i+1} \\ \mathbf{y} - \mathbf{y}^{i+1} \end{bmatrix} \leq 0 \ \forall k \in K_{i+1}, \tag{6}$$

where K_{i+1} is the index set of all active and violated constraints. The new cutting planes will exclude the solution $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ from the search space, see *e.g.,* Westerlund & Pörn (2002). The new cutting planes are added to the polyhedral approximation, and we denote the new approximation as $\hat{N}_{i+1}$. The polyhedral approximation of the set N is, thus, improved by the accumulation of cutting planes. In the next iteration, we solve subproblem (MILP-i) updated with new cuts to obtain a new trial solution.

Now, in case the trial solution $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ is feasible it may still not be the best possible one with the integer combination given by $\mathbf{y}^{i+1}$. Therefore, we will fix the integer variables in the original MINLP problem to the values given by $\mathbf{y}^{i+1}$, resulting in the following convex NLP problem

$$\begin{aligned} (\mathbf{x}^{i+1}, \mathbf{y}^{i+1}) \in \ & \underset{(\mathbf{x},\mathbf{y}) \in N \cap L \cap Y}{\arg\min} \ \mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y} \\ & \text{s.t.} \quad \mathbf{y} = \mathbf{y}^{i+1}. \end{aligned} \tag{NLP-fixed}$$

By solving problem (NLP-fixed) we obtain the optimal solution for this specific integer combination. However, the obtained solution may still not be the optimal one to the original MINLP problem. In order to obtain better solutions, we will therefore, generate an objective cut according to

$$\mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y} \leq \mathbf{c}_1^T \mathbf{x}^{i+1} + \mathbf{c}_2^T \mathbf{y}^{i+1}, \tag{7}$$

where $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ is the solution obtained by solving subproblem (NLP-fixed). The cut given by Eq. (7) will exclude all solutions that have a worse objective function value than the obtained feasible solution, and will thus reduce the search space. To obtain better solutions we include the objective cut in the polyhedral approximation $\hat{N}_{i+1}$. Subproblem (MILP-i), by which we choose the new trial solutions, will then contain the objective cut given by Eq. (7) in the following form

$$\mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y} + r \left\| \begin{matrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{matrix} \right\|_2 \leq \mathbf{c}_1^T \mathbf{x}^{i+1} + \mathbf{c}_2^T \mathbf{y}^{i+1}, \tag{8}$$

forcing the next inscribed ball to also take the objective cut into consideration. As long as we obtain solutions to subproblems (MILP-i) with $r^i > 0$, it is clear that the constraint given by Eq. (8) will force the trial solutions to have a strictly lower objective function value than the obtained feasible solution. Thus, the objective cut will force the algorithm to search for better solutions. Once an objective cut has been added to the polyhedral approximation $\hat{N}_i$ it will no longer be an outer approximation of the set N. However, due to convexity, we know that the optimal solution will not be excluded from the search space. The search space can further be reduced by generating cutting planes for all nonlinear constraints active at the feasible solution $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ according to Eq. (6).

In each iteration the radius of the ball inscribed in $\hat{N}_i$ is reduced since the sets $\hat{N}_i$ shrink due to the added cuts. Later, we prove that the cuts added in each iteration force the radius to converge to zero. If the radius of the largest ball inscribed in $\hat{N}_i$ is zero, the set $\hat{N}_i$ will then have an empty interior, thus verifying that the optimal solution has been found. In case the original MINLP problem is infeasible, the radius will converge to zero without finding any feasible solution. The convergence properties are discussed in more detail in Section 5.

The center-cut algorithm is summarized as a pseudo-code in Algorithm 1. In the algorithm, we use the radius as an optimality measure, since a smaller radius will ensure better solutions. However, in order to guarantee that the best-found solution is the optimal solution we must continue until the radius is reduced to zero.

Note that, in case the original MINLP problem has a convex nonlinear objective function, we can simply replace the left-hand side of the objective cut in Eq. (7) by a linearization of the objective. There is, therefore, no need to reformulate the problem to obtain a linear objective function.

In the next section, we apply the center-cut algorithm to an illustrative example with two variables to give a geometric interpretation of the algorithm.

Algorithm 1 Pseudo-code of the center-cut algorithm

```
Specify a tolerance $r_{min} \geq 0$.
  1. Initialization.
      1.1 Set $B_1 = \mathbb{R}^{n+m}$, set iteration counter $i = 1$.
      1.2 Solve problem (MILP-$i$) to obtain $(\mathbf{x}^1, \mathbf{y}^1)$ and $r^1$.
  2. While $r^i > r_{min}$.
      2.a If $(\mathbf{x}^i, \mathbf{y}^i)$ satisfies all nonlinear constraints:
          * Solve problem (NLP-fixed) to obtain the optimal solution with the given integer solution and store
            the solution as $(\mathbf{x}^i, \mathbf{y}^i)$.
          * Construct cutting planes for any active constraint according to Eq. (6) and the objective cut according
            to Eq. (7).
          * Generate the set $B_{i+1}$ by adding the new cuts to $B_i$
      2.b If $(\mathbf{x}^i, \mathbf{y}^i)$ does not satisfies all nonlinear constraints:
          * Obtain cutting planes for all violated constraints according to Eq. (6).
          * Generate the set $B_{i+1}$ by adding all cutting planes to $B_i$
      2.c Solve problem (MILP-$i$) to obtain $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}, r^{i+1})$. and set $i = i + 1$.
  3. Return the best found feasible solution $(\mathbf{x}^i, \mathbf{y}^i)$.
```

4 Illustrative example

For MINLP problems with only two variables, the center-cut algorithm chooses the trial solutions by inscribing the largest possible circle in $\hat{N}_i$, such that the center of the circle satisfies all linear constraints and integer requirements. To illustrate the basics of the center-cut algorithm, consider the following simple MINLP problem

$$\begin{aligned} \min \quad & -3x - y \\ \text{s.t.} \quad & x^2 + y^2 \leq 25, \\ & x^2 + (5-y)^2 \leq 36, \\ & (5-x)^2 + y^2 \leq 25, \\ & 0 \leq x \leq 10, \quad 0 \leq y \leq 10, \\ & x \in \mathbb{R},\ y \in \mathbb{Z}. \end{aligned} \tag{Ex 1}$$

The MINLP problem (Ex 1) is illustrated in Figure 1. We have applied the basic center-cut algorithm, as presented in Algorithm 1, to the illustrative example problem (Ex 1) and the iterations are shown in Figure 2. In the first iteration, we have no cutting planes approximating the set N and the radius is therefore not limited, and any solution satisfying the linear constraints and integer requirement can be chosen. In the second iteration, we obtain a solution where the circle's center satisfies all constraints, and the solution is improved by solving problem (NLP-fixed). In iteration 2 we generate an objective cut according to Eq. (7) and a cutting plane for the second nonlinear constraint. The optimal solution is obtained in iteration 4, but we need an additional iteration to verify optimality. In iteration 5 we find that largest inscribed circle has a radius of zero, thus proving that the optimal solution has been found. As a comparison, it takes 9 iterations with the basic ECP algorithm to find a feasible solution and 3 iterations with OA.

In the next section, we describe why the radius of the inscribed ball can be used as an optimality measure, and we prove that the center-cut algorithm will find an optimal solution to the MINLP problem in a finite number of iterations.

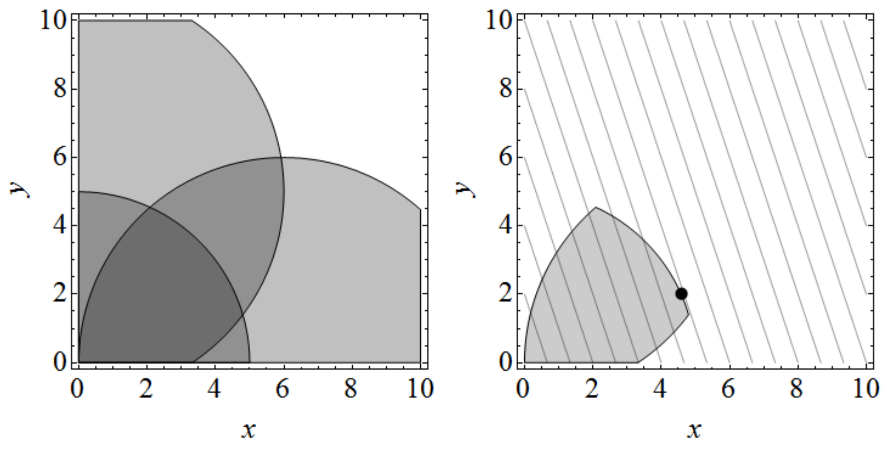

Figure 1: The figure to the left shows the feasible regions defined by the nonlinear constraint of problem (Ex 1). The second figure shows the feasible region defined by the constraints, contours of the objective function and the optimal solution.

5 Proof of convergence

Here we focus on the convergence properties of the center-cut algorithm. We show that the radius of the inscribed ball converges to zero, and from there we can show that the algorithm will converge to the optimal solution of a convex MINLP problem.

To prove that the radius of the inscribed balls will converge to zero, we need some properties presented in Lemma 1.

Lemma 1. *In iteration i with the center-cut algorithm, the radius of the ball inscribed in $\hat{N}_i$ will be bounded by distance between the current center $(\mathbf{x}^i, \mathbf{y}^i)$ and any previously obtained center $(\mathbf{x}^{i-l}, \mathbf{y}^{i-l})$ according to*

$$r^i \leq \left\| \begin{matrix} \mathbf{x}^i - \mathbf{x}^{i-l} \\ \mathbf{y}^i - \mathbf{y}^{i-l} \end{matrix} \right\|_2,$$

where $0 < l < i$.

Proof. At iteration $i - l$, we generate cuts according to Eq. (6) and if we obtain a feasible solution we also add an objective cut according to Eq. (7). These cuts will either exclude the center $(\mathbf{x}^{i-l}, \mathbf{y}^{i-l})$ from the set $\hat{N}_{i-l+1}$ or result in a cut which passes through it. The center $(\mathbf{x}^{i-l}, \mathbf{y}^{i-l})$ is, thus, either outside $\hat{N}_{i-l+1}$ or on the boundary of $\hat{N}_{i-l+1}$. Therefore, the radius r^i cannot be greater than the distance from the current center $(\mathbf{x}^i, \mathbf{y}^i)$ to the previously obtained center $(\mathbf{x}^{i-l}, \mathbf{y}^{i-l})$, otherwise parts of the ball would be outside of $\hat{N}_{i-l+1}$. □

By using the bounds on the radius given by Lemma 1, it is possible to prove that the radius converges to zero as described by the following theorem.

Theorem 1. *With the center-cut algorithm, the radius r^i of the inscribed balls converge to zero when $i \to \infty$.*

Proof. Assume that the algorithm does not stop and we obtain an infinite sequence of centers $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^{\infty}$. Due to Assumption 2, we know that all centers in the sequence belong to a compact subset of $\mathbb{R}^{n+m}$. According to the Bolzano-Weirstrass theorem, the sequence must contain at least one convergent subsequence $\{\mathbf{x}^{i_j}, \mathbf{y}^{i_j}\}_{i_j=1}^{\infty}$. The convergent subsequence also forms a Cauchy sequence with the following property

$$\lim_{j \to \infty} \left\| \begin{matrix} \mathbf{x}^{i_j} - \mathbf{x}^{i_j - 1} \\ \mathbf{y}^{i_j} - \mathbf{y}^{i_j - 1} \end{matrix} \right\|_2 = 0.$$

Note that Lemma 1 is true for any two centers, and therefore we obtain

$$\lim_{i \to \infty} r^i = 0.$$

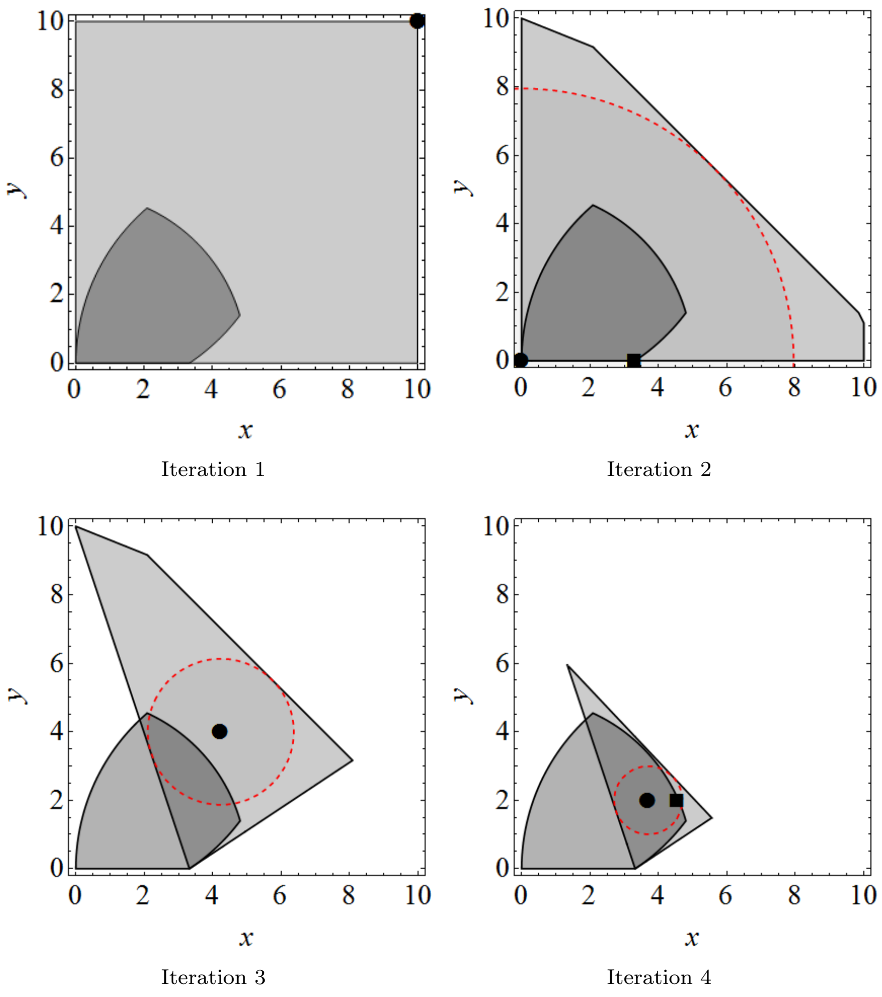

Figure 2: The figure shows the first four iterations of the center-cut algorithm applied to problem (Ex 1). The figures show the feasible region defined by the nonlinear constraints and the region defined by the sets $\hat{N}_i$. The circular dot represents the center of the inscribed circle and the dashed curves represent the circle. The solution obtained by solving subproblem (NLP-fixed) is shown by the squared dot.

$\square$

In order to prove that the algorithm obtains the optimal solution in a finite number of iterations, we need some further properties presented in Lemma 2, regarding the geometry of the feasible region. The proof of Lemma 2 follows from Slater's condition, but for the sake of completeness, we have included the proof. Here we denote the optimal value of the objective function of the MINLP problem as z^*.

Lemma 2. *For any $\epsilon > 0$, we can inscribe a ball with nonzero radius in the set*

$$N^* = \left\{(\mathbf{x}, \mathbf{y}) \mid \mathbf{c}_1^T\mathbf{y} + \mathbf{c}_2^T\mathbf{x} \leq z^* + \epsilon,\ g_k(\mathbf{x}, \mathbf{y}) \leq 0 \quad \forall\ k\right\}, \tag{9}$$

such that the center of the ball satisfies all constraints of the MINLP problem.

Proof. Note that z^* is given by $z^* = \mathbf{c}_1^T\mathbf{x}^* + \mathbf{c}_2^T\mathbf{y}^*$, where $(\mathbf{x}^*, \mathbf{y}^*)$ is an optimal solution to the MINLP problem. The optimal solution strictly satisfies the restriction on the objective function

$$\mathbf{c}_1^T\mathbf{x}^* + \mathbf{c}_2^T\mathbf{y}^* < z^* + \epsilon.$$

However, the optimal solution might be located on the boundary of the set N^* and therefore we cannot use it as center for the ball. By Assumption 3, we know that the nonlinear constraints satisfy Slater's condition even if we fix the integer variables to $\mathbf{y}^*$, *i.e.,* $\exists\ \bar{\mathbf{x}} : A\bar{\mathbf{x}} + B\mathbf{y}^* \leq \mathbf{b},\ g_k(\bar{\mathbf{x}}, \mathbf{y}^*) < 0\ \forall\ k$. Next, we define a new point as

$$\hat{\mathbf{x}} = \alpha\mathbf{x}^* + (1-\alpha)\bar{\mathbf{x}}, \tag{10}$$

where $\alpha \in [0, 1)$ is an interpolation parameter. Now, we want to choose α such that $(\hat{\mathbf{x}}, \mathbf{y}^*)$ strictly satisfies all the constraints in the set N^*. If $\mathbf{c}_1^T(\bar{\mathbf{x}} - \mathbf{x}^*) < \epsilon$, it is sufficient to choose $\alpha = 0$ to strictly satisfy the objective constraint in N^*. Otherwise α can be chosen as $\alpha = \left(\frac{\epsilon}{2} + \mathbf{c}_1^T(\mathbf{x}^* - \bar{\mathbf{x}})\right)/\mathbf{c}_1^T(\mathbf{x}^* - \bar{\mathbf{x}}) < 1$, which results in

$$\mathbf{c}_1^T\hat{\mathbf{x}} + \mathbf{c}_2^T\mathbf{y}^* = z^* + \frac{\epsilon}{2}.$$

Since $\hat{\mathbf{x}}$ was chosen as an interpolation between two points, with the same integer combination and both satisfy all the constraints, it is clear that $(\hat{\mathbf{x}}, \mathbf{y}^*)$ will satisfy all constraints. Furthermore, since $(\bar{\mathbf{x}}, \mathbf{y}^*)$ strictly satisfies the nonlinear constraints and $\alpha < 1$ we get

$$g_k(\hat{\mathbf{x}}, \mathbf{y}^*) < 0\ \forall\ k.$$

The point $(\hat{\mathbf{x}}, \mathbf{y}^*)$ is, thus, located within the interior of the set N^*, and therefore it is possible to put a ball with a nonzero radius at $(\hat{\mathbf{x}}, \mathbf{y}^*)$ such that the entire ball is contained in the set N^*. $\square$

Now, we have all the intermediate results needed for proving that the optimal solution is obtained in a finite number of iterations.

Theorem 2. *The center-cut algorithm obtains the optimal solution to problem* (P-MINLP) *in a finite number of iterations.*

Proof. As before we denote the optimal solution to the MINLP problem as $(\mathbf{x}^*, \mathbf{y}^*)$ and the optimal objective value as z^*. Next we can choose an $\epsilon > 0$, such that $N^* \cap L \cap Y$ only contains optimal values for the integer variables $\mathbf{y}$, where N^* is given by Eq. (9). From Lemma 2, we know that we can inscribe a ball with radius $r^* > 0$ in the set N^*, such that the center satisfies all constraints.

As long as the algorithm has not obtained the optimal solution, the entire set N^* will be contained within $\hat{N}_i$, *i.e.,* $N^* \subset \hat{N}_i$. This is true because all the cutting planes added according to Eq. (6) are overestimating the feasible region defined by the nonlinear constraints. As long as $N^* \subset \hat{N}_i$, we know that the radius of the inscribed balls will be greater or equal to r^*. From Theorem 1 we know that the radius of the inscribed balls converges to zero, and therefore there exists a finite integer p such that

$$\forall i \geq p \quad r^i < r^*.$$

The only way to reduce the radius below r^*, is to generate an objective cut according to Eq. (7) stricter than the objective cut in N^*. Such an objective cut must be generated in iteration p at a feasible solution $(\mathbf{x}^p, \mathbf{y}^p)$ such that $\mathbf{c}_1^T\mathbf{x}^p + \mathbf{c}_2^T\mathbf{y}^p < z^* + \epsilon$. In the beginning, we chose ϵ such that the objective function can only be within ϵ from the optimum if the integer variables takes on optimal values, *i.e.,* $\mathbf{y}^p = \mathbf{y}^*$. Furthermore, the variables in iteration p will be chosen by solving subproblem (NLP-fixed) with the integer variables fixed as $\mathbf{y}^*$, and the subproblem will then return an optimal solution for the continuous variables, *i.e.,* $(\mathbf{x}^p, \mathbf{y}^p) = (\mathbf{x}^*, \mathbf{y}^*)$. The optimal solution to the MINLP problem was, thus, obtained in iteration p. $\square$

From the proof of Theorem 2, it follows that the optimal solution will be obtained once the radius of the inscribed ball is reduced below a certain value. Furthermore, if the radius is reduced to zero it automatically verifies that the optimal solution has been obtained. In the algorithm, we, therefore, use the radius as an optimality measure and termination criterion. For rigorously verifying optimality, the radius needs to be reduced to zero, however, in practice it is often sufficient to stop once the radius is close to zero, *e.g.,* smaller than 10^{-4}.

In this section, we have proven that the algorithm will find the optimal solution to any convex MINLP problem satisfying Assumptions 1, 2 and 3. The next section deals with some details regarding the implementation of the algorithm.

6 Implementing the algorithm

In previous sections, we have described the basics of the center-cut algorithm. To test the practical performance of the center-cut algorithm, we implemented it in Matlab 2017a and used Ipopt 3.12.7 (Wächter & Biegler, 2006) and Gurobi 7.5.2 as subsolvers for the NLP and MILP subproblems, respectively. We have also used OPTI Toolbox (Currie & Wilson, 2012) to read the test problems. When implementing the center-cut algorithm, it is possible to incorporate some tricks and features from other algorithms and solvers; next, we describe some of these that can easily be exploited.

First, when solving an MINLP problem with the center-cut algorithm it is not necessary to solve every single MILP subproblem to optimality. It is sufficient to obtain a feasible solution, such that the inscribed ball has a radius strictly greater than zero. This is an important detail since solvers such as Cplex or Gurobi are often able to quickly find several feasible solutions to an MILP problems, and often the majority of the solution time is spent on proving optimality. The MILP subproblems are also by far the most time-consuming part of the center-cut algorithm. By stopping the MILP solver after a specific number of found feasible solutions, it is often possible to significantly reduce the solution time while still obtaining good solutions to the mixed-integer subproblems. This can be done with Gurobi by using the solution limit parameter. In the implementation of the center-cut algorithm, we simply start with the solution limit parameter set to 2 and increase the solution limit parameter by one each time the radius of the inscribed ball is less than 0.001 and the solution was not reported as optimal. We also use a second test for increasing the solution limit, where the solution limit is increased if the radius is less than half the radius in the previous iteration and the solution was not optimal. When choosing the solution limit by this technique, we start with a low solution limit and gradually increase it during the iterations to make sure we obtain good solutions to the subproblems. When using this technique one must be careful with the termination criterion, and the search should not be terminated unless the MILP subproblem was solved to optimality in the last iteration. A similar approach for speeding up the MILP subproblems and consequently speeding up the MINLP solution procedure is also used with the GAECP algorithm, see Westerlund & Pörn (2002).

In some cases, it might also be possible to speed up the algorithm by solving additional NLP subproblems. Even if the trial solution $(\mathbf{x}^i, \mathbf{y}^i)$ obtained by solving subproblem (MILP-i) does not satisfy the nonlinear constraints, $\mathbf{y}^i$ may still be a feasible integer combination. It might, therefore, be possible to obtain a feasible solution by fixing the integer variables to $\mathbf{y}^i$ and solving subproblem (NLP-fixed). This situation is illustrated in iteration 3 in Figure 2, where it would have been possible to obtain a feasible solution by solving an NLP subproblem. By solving such NLP problems it may be possible to obtain feasible solutions more frequently, but the additional NLP problems may also take some time to solve. In the implementation of the center-cut algorithm, we try to fix the integer variables and solve subproblem (NLP-fixed) in every third iteration. A similar technique is used in both the GAMS solver AlphaECP (Lastusilta et al., 2009) and in the SHOT solver (Kronqvist et al., 2016). The NLP problems with fixed integers may be infeasible in some iterations, however, in this case, Ipopt returns a solution that minimizes the constraint violation with the specific integer combination. By adding cuts according to Eq. (6) at the infeasible solution returned by Ipopt, we are able to exclude the infeasible integer combination from the search space, for details on such cuts see Fletcher & Leyffer (1994). In the implementation of the center-cut algorithm, we use this technique for generating cuts, when the NLP solver cannot find a feasible solution.

In the implementation, we have used the center-cut algorithm as described in Algorithm 1 together with the additional features described in here. With the NLP solver Ipopt, we have used the default settings for all parameters. With Gurobi we have used the solution limit strategy as described earlier and for the other parameters, we have used default settings.

7 Numerical results

To test the practical performance of the center-cut algorithm, we considered some convex MINLP test problems taken from the library MINLPLib2 (GAMSWorld, 2017). For the tests, we have used a standard desktop computer with an Intel i7 processor and 16GB of RAM.

First, we have chosen 8 test problems from MINLPLib 2 that represent different types of MINLP problems, such as some facility layout problems (Castillo et al., 2005), retrofit planning problems (Sawaya, 2006) and trim loss problems (Harjunkoski et al., 1998). The largest of these problems contains 1500 binary variables,

Results obtained by the center-cut implementation.

Name of MINLP problem	flay06h	gams01	ibs2	o7
Time/iterations to find a a feasible solution.	0.2 s / 2	1.3 s / 2	0.9 s / 2	2.8 s / 8
Time/iterations to find a second feas. sol.	*	1.9 s / 3	*	10 s / 10
Time/iterations to find a sol. within 5% of opt.	0.2 s / 2	52 s / 28	0.9 s / 2	313 s / 28
Time/iterations to find a sol. within 1% of opt.	0.2 s / 2	52 s / 28	0.9 s / 2	433 s / 30
Sub-optimality of best-found solution.	0 %	0.5 %	1 %	0 %

Name of MINLP problem	rsyn0815m04h	stockcycle	tls6	tls7
Time/iterations to find a feasible solution.	0.2 s / 2	0.1 s / 1	0.7 s / 9	2.5 s / 14
Time/iterations to find a second feas. sol.	1.3 s / 3	0.7 s / 2	13 s / 36	5 / 26
Time/iterations to find a sol. within 5% of opt.	13 s / 13	63 s / 103	830 s / 159	*
Time/iterations to find a sol. within 1% of opt.	22 s / 21	231 s / 157	*	*
Sub-optimality of best-found solution.	0.5 %	0.5 %	3 %	33 %

Results obtained by the feasibility pump in DICOPT

Name of MINLP problem	flay06h	gams01	ibs2	o7
Time/iterations to find a feasible solution.	0.2 s / 1	0.3 s / 1	*	0.3 s / 4
Time/iterations to find a second feas. sol.	0.3 s / 2	0.7 s / 2	*	0.4 s / 5
Time/iterations to find a sol. within 5%.	0.4 s / 3	8.3 s / 21	*	3.1 s / 19
Time/iterations to find a sol. within 1% of opt.	0.6 s / 5	*	37 s / 45	66 s / 23
Sub-optimality of best-found solution.	0%	0%	∞	0%

Name of MINLP problem	rsyn0815m04h	stockcycle	tls6	tls7
Time/iterations to find a feasible solution.	2.7 s / 1	0.1 s / 1	167 s / 48	*
Time/iterations to find a second feas. sol.	3.9 s / 3	0.2 s / 2	*	*
Time/iterations to find a sol. within 5% of opt.	5.7 s / 10	4.7 s / 51	*	*
Time/iterations to find a sol. within 1% of opt.	9.1 s / 14	226 s / 295	*	*
Sub-optimality of best-found solution.	0%	1%	12%	∞

Table 1: The table shows the results obtained with the center-cut implementation and the feasibility pump in DICOPT. The sign * indicates that no such solution was obtained within the time limit of 1800 seconds.

1500 continuous variables, and 1821 constraints. These specific problems were chosen since they are known to be difficult to solve. We have applied the center-cut implementation to the test problems and the results are shown in Table 1. The table shows the time and number of iterations needed to find a feasible solution, a second feasible solution, a solution within 5 % of the best-known solution, a solution within 1 % of the best-known solution and the quality of the best-found solution. Besides the settings described in the previous section, we have used a time limit of 1800 seconds.

To get a reference point for evaluating the performance, we have used the feasibility pump available in the state-of-the-art solver DICOPT in GAMS (Grossmann et al., 2002) on the same problems. As previously mentioned, the feasibility pump is a primal heuristic intended for quickly finding good solutions. The results obtained with the feasibility pump are shown in the last part of Table 1. With the feasibility pump, we have used Conopt and Gurobi as subsolvers, and we have used the following parameters to make sure we can obtain a solution within 1 % of the optimal solution: `fp_cutoffdecr = 0.01, fp_timelimit = 1800, fp_stalllimit = 10000` and `fp_sollimit = 10000`. We used Conopt as a NLP subsolver since it gave the best performance with the feasibility pump. Note this is not intended as a comparison because comparing the feasibility pump in DICOPT with the Matlab implementation of the center-cut algorithm is not fair. The Matlab implementation is quite simple and mainly intended as a proof of concept and to show the potential of the center-cut algorithm.

TTable 1 shows that the center-cut implementation is able to find a feasible solution to all of the 8 problems within less than 3 seconds. Furthermore, we are able to find good solutions to all of the problems except tls7, where the best-obtained solution is still far from the best-known solution. However, it should be noted that these are difficult problems and tls7 is one of the few convex MINLP problems in MINLPLib2

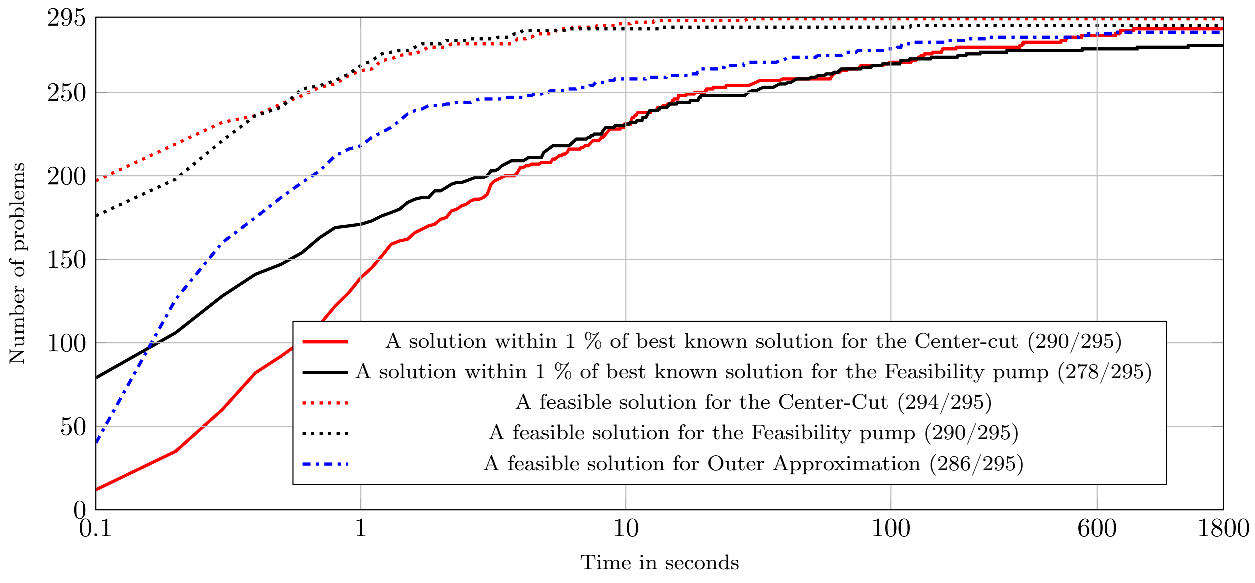

Figure 3: The lines shows number of problems that the center-cut implementation is able to find a solution to as a function of running time.

that are still considered as unsolved. The feasibility pump struggles with some of the problems and is not able to find any solution for two of the problems. The feasibility pump is quicker at obtaining solutions to problems gams01 and o7_ar2_1, but for the other problems, the center-cut implementation seems to be more efficient.

To further test the center-cut implementation, we applied it to 295 test problems from MINLPLib2. In this test set, we chose all convex problems from MINLPLib2 containing at least one discrete variable, and we removed the instances where the only nonlinearity was due to a quadratic objective function. Convex problems where the only nonlinear term is a quadratic objective can be solved efficiently directly with Gurobi. For such problems, the center-cut algorithm is not necessarily a good choice, and therefore, we removed these test problems. The results obtained with the center-cut implementation are presented in Figure 3. We have applied the feasibility pump to the test problems to get a reference point for the evaluating the results. We have also used a basic implementation of OA, described in **?**, to show the time needed to obtain feasible solution with OA.

Figure 3 shows the number of problems that the center-cut implementation is able to find a feasible solution to as a function of time. The figure shows that the center-cut implementation is able to find feasible solutions to these problems quickly. For 250 of the test problems, it is possible to find a feasible solution by running center-cut implementation for less than 1 second. Furthermore, the implementation requires less than 10 seconds to find a feasible solution to 291 of the 295 test problems and there is only one problem (tls12) where the implementation fails to find a feasible solution. Compared with the feasibility pump, the center-cut is able to find feasible solution for more problems in 0.1 s and the overall performance is similar. Within the given time limit the center-cut is able to find feasible solutions to all but one of the test instances, while the feasibility pump is not able to find a feasible solution to five of the instances. The figure also shows that OA is significantly slower at obtaining feasible solutions.

Finding a solution within 1% of the best-known solution requires more time. By running the center-cut implementation for 10 seconds it is possible to find a solution within the 1% tolerance for 236 of the test problems, and within the given time limit we managed to find a solution within the tolerance for 288 of the test problems. The feasibility pump is bit faster at obtaining solutions within the 1% tolerance for the easier test problems, which is partially due to a more efficient implementation. For the instances requiring more than 3 seconds, the performance of the feasibility pump and center-cut is quite similar. However, in the end, the feasibility pump fails to find a solution within the tolerance for 17 of the test problems, whereas the center-cut implementation only fails on 7 of the problems.

The intention of the numerical results has not been to compare the feasibility pump in DICOPT with the simple Matlab implementation of the center-cut algorithm, but to show the potential of the center-cut algorithm. However, the results have shown that the performance of center-cut implementation is actually on par with the feasibility pump in the state-of-the-art solver DICOPT.

The numerical results have shown that the center-cut algorithm may be well suited as a primal heuristic. For the test problems we were able to quickly find feasible solutions to almost all of the 295 test problems, and furthermore, we are also able to obtain solutions of good quality.

8 Conclusions

In this paper, we have given a detailed presentation of the center-cut algorithm for convex MINLP, and we have proven that the algorithm finds the optimal solution in a finite number of iterations. The algorithm uses a different approach to obtaining trial solutions which should make it able to quickly obtain feasible solutions, and this was verified by the numerical results. The ability to quickly obtain feasible solutions makes the algorithm well suited as a primal heuristic, but it can also be used as a deterministic solution technique. With the center-cut algorithm we do not directly obtain a lower bound as in ESH or ECP, and therefore it could be efficient to combine these algorithms in a solver to obtain both a lower and upper bound.

Acknowledgement

The authors are grateful for the support from GAMS Development Corporation, and furthermore, David Bernal would like to thank the Center for Advanced Process Decision-making (CAPD) for financial support.

References

Bernal, D. E., Vigerske, S., Trespalacios, F., & Grossmann, I. E. (2017). Improving the performance of dicopt in convex minlp problems using a feasibility pump, .

Berthold, T. (2014). *Heuristic algorithms in global MINLP solvers*. Verlag Dr. Hut.

Berthold, T., & Gleixner, A. M. (2014). Undercover: a primal minlp heuristic exploring a largest sub-mip. *Mathematical Programming*, *144*, 315–346.

Biegler, L. T., & Grossmann, I. E. (2004). Retrospective on optimization. *Computers & Chemical Engineering*, *28*, 1169–1192.

Bonami, P., Cornuéjols, G., Lodi, A., & Margot, F. (2009). A feasibility pump for mixed integer nonlinear programs. *Mathematical Programming*, *119*, 331–352.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

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*, 54–69.

Currie, J., & Wilson, D. I. (2012). OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User. In N. Sahinidis, & J. Pinto (Eds.), *Foundations of Computer-Aided Process Operations*. Savannah, Georgia, USA.

Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, *8*, 250–255.

Duran, M. A., & Grossmann, I. E. (1986). An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, *36*, 307–339.

Elzinga, J., & Moore, T. G. (1975). A central cutting plane algorithm for the convex programming problem. *Mathematical Programming*, *8*, 134–145.

Fischetti, M., & Lodi, A. (2011). Heuristics in mixed integer programming. *Wiley Encyclopedia of Operations Research and Management Science*, .

Fletcher, R., & Leyffer, S. (1994). Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, *66*, 327–349.

Floudas, C. A. (2000). Deterministic Global Optimization, vol. 37 of Nonconvex Optimization and its Applications.

GAMSWorld (2017). Mixed-integer nonlinear programming library. URL: `http://www.gamsworld.org/minlp/minlplib2/html/` [Online; accessed 27-December-2017].

Geoffrion, A. M. (1972). Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, *10*, 237–260.

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

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*, 594–603.

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

Kronqvist, J., Lundell, A., & Westerlund, T. (2017). A center-cut algorithm for solving convex mixed-integer nonlinear programming problems. In *Computer Aided Chemical Engineering* (pp. 2131–2136). Elsevier volume 40.

Lastusilta, T., Bussieck, M. R., & Westerlund, T. (2009). An experimental study of the GAMS/AlphaECP MINLP solver. *Industrial & Engineering Chemistry Research*, *48*, 7337–7345.

Lee, J., & Leyffer, S. (Eds.) (2011). *Mixed Integer Nonlinear Programming* volume 154. Springer Science & Business Media.

Sawaya, N. (2006). *Reformulations, relaxations and cutting planes for generalized disjunctive programming* volume 67.

Slater, M. et al. (1959). *Lagrange multipliers revisited*. Technical Report Cowles Foundation for Research in Economics, Yale University.

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, *106*, 25–57.

Westerlund, T., & Petterson, F. (1995). An extended cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, *19*, S131–S136.

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