Journal of Global Optimization (2021) 81:861–880
https://doi.org/10.1007/s10898-021-01027-w

Branch-and-price for a class of nonconvex mixed-integer nonlinear programs

Andrew Allman[1] · **Qi Zhang**[1]

Received: 6 January 2020 / Accepted: 24 April 2021 / Published online: 7 May 2021

Abstract
This work attempts to combine the strengths of two major technologies that have matured over the last three decades: global mixed-integer nonlinear optimization and branch-and-price. We consider a class of generally nonconvex mixed-integer nonlinear programs (MINLPs) with linear complicating constraints and integer linking variables. If the complicating constraints are removed, the problem becomes easy to solve, e.g. due to decomposable structure. Integrality of the linking variables allows us to apply a discretization approach to derive a Dantzig-Wolfe reformulation and solve the problem to global optimality using branch-andprice. It is a remarkably simple idea; but to our surprise, it has barely found any application in the literature. In this work, we show that many relevant problems directly fall or can be reformulated into this class of MINLPs. We present the branch-and-price algorithm and demonstrate its effectiveness (and sometimes ineffectiveness) in an extensive computational study considering multiple large-scale problems of practical relevance, showing that, in many cases, orders-of-magnitude reductions in solution time can be achieved.

Keywords Mixed-integer nonlinear programming · Branch-and-price · Decomposition · Nonconvex optimization

1 Introduction

Mixed-integer nonlinear programming (MINLP) has proven to be a powerful modeling paradigm and has received increased attention in recent years. However, despite the tremendous advances in the theoretical and algorithmic treatment of MINLPs, they are still significantly less scalable than their linear counterparts such that solving MINLPs of large sizes remains a challenge. In this work, we consider the exact solution of a class of generally nonconvex MINLPs whose structure is amenable to branch-and-price, hence allowing us to combine the strengths of two key enabling technologies: column generation for large-scale integer programming and global optimization of MINLPs. Specifically, we consider problems of the following form:

✉ Qi Zhang
qizh@umn.edu

[1] Department of Chemical Engineering and Materials Science, University of Minnesota, Twin Cities, Minneapolis, MN 55455, USA

$$\underset{x,y,z}{\text{minimize}} \quad c^\top x + f(y,z) \tag{1a}$$

$$\text{subject to} \quad Ax + Dy \geq b \tag{1b}$$

$$g(y,z) \leq 0 \tag{1c}$$

$$y^{\min} \leq y \leq y^{\max} \tag{1d}$$

$$x \in \mathbb{R}^m_+ \times \mathbb{Z}^{\bar{m}}_+,\ y \in \mathbb{Z}^p,\ z \in \mathbb{R}^q \times \mathbb{Z}^{\bar{q}}, \tag{1e}$$

where the vectors x and z can contain both continuous and integer variables, while the y-variables are all integer. Note that without loss of generality, x are constrained to be nonnegative. The objective function (1a) consists of a linear term, $c^\top x$, and a nonlinear term, $f(y,z)$. While the linear constraints (1b) involve x and y, $g(y,z)$ in (1c) are nonlinear functions of y and z. The functions f and g can be nonconvex. Constraints (1d) ensure that y are bounded. We assume that problem (1) can be efficiently solved if constraints (1b), which we call the *complicating constraints*, are removed.

We are particularly interested in problems that, without the complicating constraints, result in smaller MINLPs that can be solved using state-of-the-art global solvers. This most commonly occurs when the problem has a decomposable structure, such that multiple independent subproblems are generated when the complicating constraints are removed, but can also occur when the number of linear constraints is much larger than the number of nonlinear constraints. We find that many relevant problems directly fall or can be reformulated into the aforementioned class of MINLPs. Examples can be found in process and product design, production capacity planning, dynamic facility location, stochastic programming, statistical learning, and many more application domains.

In this work, we investigate the computational feasibility of a branch-and-price approach to solving MINLPs of the given form. While the suitability of the proposed algorithm has been indicated in the literature [1–3], it has barely found any application. We hence aim to systematically analyze the theoretical basis of the branch-and-price approach, highlight critical algorithmic considerations, and examine the algorithm's performance in extensive computational experiments.

The remainder of this paper is organized as follows: In Sect. 2, we review existing works for finding the exact solution of MINLPs and solving large-scale MILPs using branch-and-price. Next, we present an approach for reformulating problems of the form presented in. (1) using discretization in Sect. 3. This reformulation makes the problem amenable for solution using branch-and-price, and the algorithm for doing so is presented in Sect. 4. We show how this approach can be extended to problems where the pricing subproblem is decomposable in Sect. 5. In Sect. 6, the performance of the branch-and-price approach is assessed in four representative case studies. Finally, in Sect. 7 we provide some concluding remarks.

2 Literature review

The development of exact methods for the solution of mixed-integer *linear* programs (MILPs) dates back to the 1950s [4,5] (for more details on the history of integer programming, see [6]). Over the last decades, MILP has reached a level of maturity that has made it the primary approach to solving many industrial and scientific problems of high complexity and dimensionality. Mixed-integer *nonlinear* programming is a more recent development and was initially motivated by applications in chemical and process systems engineering [7].

The development of MINLP algorithms has initially focused on the convex case, i.e. problems in which, loosely speaking, the continuous relaxation of the MINLP is convex. Methods for solving convex MINLPs include branch-and-bound [8,9], generalized Benders decomposition [10], outer approximation [11,12], LP/NLP-based branch-and-bound [13], extended cutting plane [14], and extended supporting hyperplane [15]. For recent reviews on convex MINLP, we refer the reader to Grossmann [16], Bonami et al. [17], and Kronqvist et al. [18].

Compared to convex MINLPs, solving nonconvex MINLPs is significantly more challenging due to the nonconvexity that remains even after relaxing the integer restrictions. Exact algorithms for nonconvex MINLP incorporate concepts from global continuous optimization, such as convex relaxations and spatial branch-and-bound [19]. Major improvements have been achieved with the development of the branch-and-reduce [20,21] and α-branch-and-bound methods [22,23]. Furthermore, incorporating MILP relaxations and integer programming techniques for generating cutting planes has proven to be very effective [24]. These and other algorithmic advances are implemented in state-of-the-art global MINLP solvers, such as BARON [24], Couenne [25], LINDOGlobal [26], ANTIGONE [27], and SCIP [28]. For recent reviews focusing on nonconvex MINLP, see [29,30].

Although the performance of global MINLP solvers has improved significantly over the last two decades, they are still by far not as scalable as state-of-the-art MILP solvers. Hence, to solve large-scale nonconvex MINLPs, one often resorts to decomposition methods that exploit special model structures. Popular decomposition approaches include different variants of Lagrangean decomposition [31] and progressive hedging [32]. However, these methods have to be considered heuristics since, although they can provide lower and upper bounds, it is not guaranteed that the duality gap can be closed. Exact decomposition algorithms for nonconvex MINLP are a rarity. One of those is bilevel decomposition, which iterates between a master MILP that is a relaxation of the original MINLP and an NLP or MINLP subproblem obtained by fixing integer variables. Integer and outer-approximation cuts (or tailored cuts that can be interpreted as such) are added to the master MILP at each iteration. The convergence behavior of bilevel decomposition strongly depends on the quality of the MILP relaxation and is therefore very application-specific [33–35]. Generalized Benders decomposition has been extended to solve two-stage stochastic nonconvex separable MINLPs in which only the continuous variables are involved in the nonconvex terms [36]. Cao and Zavala[37] propose a branch-and-bound scheme to address two-stage stochastic nonconvex MINLP with mixed-integer first-stage and continuous second-stage variables. Combining generalized Benders decomposition and branch-and-cut, Li and Grossmann [38] are able to consider the case with nonconvex constraints and mixed-binary variables in both stages. Finally, Rebennack et al. [39] propose a decomposition method based on column enumeration for nonconvex MINLPs with an assignment constraint; however, since the number of columns grows exponentially with the number of assignment decisions, this method is only suited for problems with a few assignment variables.

In the same spirit of decomposition, we apply in this work branch-and-price, which has its origin in the pioneering works of Dantzig and Wolfe [40], who introduced the fundamental idea of column generation for linear programming, and Desrosiers et al. [41], who were the first to embed column generation in a branch-and-bound framework to solve a large-scale MILP. Branch-and-price has been a success story in large-scale mixed-integer optimization, with applications in vehicle routing [42,43], crew scheduling [44,45], and fleet assignment [46,47], to just name a few. For reviews on branch-and-price, see [1,48].

The vast majority of existing works on branch-and-price consider integer and mixed-integer linear problems. However, there is significant flexibility in incorporating nonlinearities

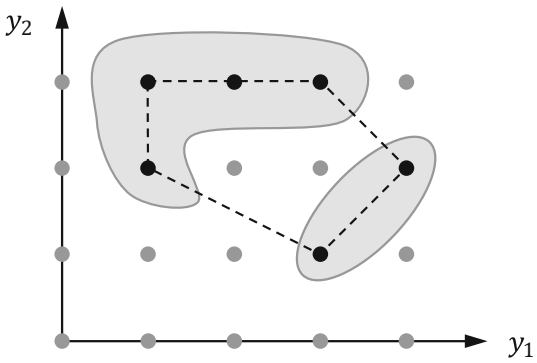

Fig. 1 The two-dimensional feasible set $\mathcal{Y}$ is given by the set of dark-colored points. The shaded area represents set $\mathcal{Y}$ without the integrality restrictions on y

in the pricing problem, which has been mentioned (in form of brief side notes) in the literature [1,2]. It is all the more surprising that there seems to be almost no published work on applying branch-and-price with mixed-integer nonlinear pricing problems. The only notable exception that we have been able to find is the work by Nowak et al. [3] in which a column-generation-based method for generating inner and outer approximations for nonconvex MINLPs is developed. We believe that there is considerable room for theory and algorithm development in the application of branch-and-price concepts to solving MINLPs.

3 Reformulation via discretization

To make the problem amenable to branch-and-price, we first apply a discretization approach [49] to derive an extensive formulation of problem (1). Consider the feasible set for y when disregarding constraints (1b):

$$\mathcal{Y} := \left\{ y : \exists z \in \mathbb{R}^q \times \mathbb{Z}^{\bar{q}} \;\text{ such that }\; g(y,z) \le 0,\;\; y^{\min} \le y \le y^{\max},\;\; y \in \mathbb{Z}^p \right\}, \tag{2}$$

which is a finite set of integer points with cardinality $K := |\mathcal{Y}|$. Hence, $\mathcal{Y}$ can be equivalently expressed as:

$$\mathcal{Y} = \{\bar{y}_1, \ldots, \bar{y}_K\} = \left\{ y : y = \sum_{k\in\mathcal{K}} \lambda_k\, \bar{y}_k,\;\; \sum_{k\in\mathcal{K}} \lambda_k = 1,\;\; \lambda_k \in \{0,1\}\; \forall\, k \in \mathcal{K} \right\}, \tag{3}$$

where $\bar{y}_k$ denotes a specific point in $\mathcal{Y}$ and $\mathcal{K} := \{1, \ldots, K\}$. Here, the binary variable λ_k is equal to 1 if $y = \bar{y}_k$ and otherwise 0. Figure 1 shows a two-dimensional example of $\mathcal{Y}$, which is given by the dark-colored points. In this representative example, the shaded area, which represents one possible set $\mathcal{Y}$ without integrality restrictions on y, is clearly nonconvex.

Remark 1 Note that in the example shown in Fig. 1, $\mathcal{Y} \neq \text{conv}(\mathcal{Y}) \cap \mathbb{Z}^p$, where $\text{conv}(\mathcal{Y})$ denotes the convex hull of $\mathcal{Y}$. In general, $\mathcal{Y} \subseteq \text{conv}(\mathcal{Y}) \cap \mathbb{Z}^p$, which is due to nonconvex constraint functions g or integer components in z. Hence, the traditional convexification technique [1] that is often used in Dantzig–Wolfe reformulations does not apply here. However, in the special case where all integer variables are binary, that is, where $y^{\min} = 0$ and $y^{\max} = 1$, the equality $\mathcal{Y} = \text{conv}(\mathcal{Y}) \cap \mathbb{Z}^p$ is guaranteed to hold.

For each $k \in \mathcal{K}$, we can define an optimal cost:

$$\bar{f}_k := \min_{z \in \mathbb{R}^q \times \mathbb{Z}^{\bar{q}}} \{ f(\bar{y}_k, z) : g(\bar{y}_k, z) \leq 0 \}, \tag{4}$$

which is the last ingredient that we need to reformulate (1) into the following *master problem*:

$$\begin{aligned}
(\text{MP}): \quad v^{\text{MP}} := \min_{x,\lambda} \quad & c^\top x + \sum_{k \in \mathcal{K}} \lambda_k \bar{f}_k && \text{(5a)}\\
\text{s.t.} \quad & Ax + D \sum_{k \in \mathcal{K}} \lambda_k \bar{y}_k \geq b && \text{(5b)}\\
& \sum_{k \in \mathcal{K}} \lambda_k = 1 && \text{(5c)}\\
& x \in \mathbb{R}^m_+ \times \mathbb{Z}^{\bar{m}}_+ && \text{(5d)}\\
& \lambda_k \in \{0, 1\} \quad \forall k \in \mathcal{K}, && \text{(5e)}
\end{aligned}$$

which is a large-scale MILP as K grows exponentially with the dimension of y. Problems (MP) and (1) are equivalent in a sense that they have the same optimal value, v^{MP}, and every solution of (MP) can be mapped to a unique solution of (1) in the (x, y)-space and a corresponding set of z-values that provide the same optimal value (the converse is trivially true).

4 The branch-and-price algorithm

In the branch-and-price algorithm, we solve (MP) using branch-and-bound, but solve the LP relaxation at each node via column generation. In the following, we describe the major elements of the algorithm.

4.1 Column generation

Consider the LP relaxation of (MP), which we denote by ($\overline{\text{MP}}$). The large number of variables in ($\overline{\text{MP}}$) typically prohibits solving it in full-space. Instead, we consider a *restricted master problem*, denoted by (RMP), which involves only a subset of columns $\widehat{\mathcal{K}} \subseteq \mathcal{K}$. The LP relaxation of (RMP), denoted by ($\overline{\text{RMP}}$), can be solved efficiently as long as the size of $\widehat{\mathcal{K}}$ is sufficiently small. New columns are generated as needed by solving the following *pricing problem*:

$$\begin{aligned}
(\text{PP}): \quad \zeta := \min_{y,z} \quad & f(y, z) - \pi^\top D y - \mu && \text{(6a)}\\
\text{s.t.} \quad & g(y, z) \leq 0 && \text{(6b)}\\
& y^{\min} \leq y \leq y^{\max} && \text{(6c)}\\
& y \in \mathbb{Z}^p, \; z \in \mathbb{R}^q \times \mathbb{Z}^{\bar{q}}, && \text{(6d)}
\end{aligned}$$

where π and μ denote the values of the dual variables associated with constraints (5b) and (5c), respectively, at the optimal solution of ($\overline{\text{RMP}}$). Problem (PP) minimizes the reduced cost; hence, if $\zeta < 0$, the corresponding y^* may improve the solution to ($\overline{\text{MP}}$) and is therefore added as a new column to $\widehat{\mathcal{K}}$. Note that (PP) is a generally nonconvex MINLP.

Since ($\overline{\text{RMP}}$) considers a restricted set of columns, its optimal value, $v^{\overline{\text{RMP}}}$, is an upper bound on the optimal value of ($\overline{\text{MP}}$), $v^{\overline{\text{MP}}}$. Following standard duality arguments [50, p. 189], one can show that a lower bound is given by $v^{\overline{\text{RMP}}} + \zeta$. Hence, we have the following relationship:

$$v^{\overline{\text{RMP}}} + \zeta \leq v^{\overline{\text{MP}}} \leq v^{\overline{\text{RMP}}}. \tag{7}$$

Algorithm 1 shows the pseudocode of the algorithm for solving ($\overline{\text{MP}}$). Here, the current lower and upper bounds on $v^{\overline{\text{MP}}}$ are denoted by $\text{LB}^{\overline{\text{MP}}}$ and $\text{UB}^{\overline{\text{MP}}}$, respectively. The column generation algorithm is finite and exact. If we set the tolerance $\bar{\epsilon}$ to zero, ζ will be zero at the last iteration and the algorithm terminates at the optimal solution of ($\overline{\text{MP}}$).

Algorithm 1 Column generation algorithm for solving ($\overline{\text{MP}}$).

1: **function** SOLVERELAXEDMP(($\overline{\text{MP}}$), (PP), $\widehat{\mathcal{K}}$)
2: Set tolerance $\bar{\epsilon}$
3: Initialize: $\text{LB}^{\overline{\text{MP}}} \leftarrow -\infty$, $\text{UB}^{\overline{\text{MP}}} \leftarrow \infty$, $k \leftarrow |\widehat{\mathcal{K}}|$
4: **while** $\text{UB}^{\overline{\text{MP}}} - \text{LB}^{\overline{\text{MP}}} > \bar{\epsilon}$ **do**
5: Solve ($\overline{\text{RMP}}$), obtain x^*, λ^*, π, μ, and $v^{\overline{\text{RMP}}}$
6: Update upper bound: $\text{UB}^{\overline{\text{MP}}} \leftarrow v^{\overline{\text{RMP}}}$
7: Solve (PP), obtain y^* and ζ
8: **if** $\zeta < 0$ **then** add column
9: $k \leftarrow k + 1$
10: $\bar{y}_k \leftarrow y^*$, $\bar{f}_k \leftarrow \zeta + \pi^\top D y^* + \mu$, and $\widehat{\mathcal{K}} \leftarrow \widehat{\mathcal{K}} \cup \{k\}$
11: **end if**
12: Update lower bound: $\text{LB}^{\overline{\text{MP}}} \leftarrow \max\{\text{LB}^{\overline{\text{MP}}}, v^{\overline{\text{RMP}}} + \zeta\}$
13: **end while**
14: **return** x^*, λ^*, $v^{\overline{\text{MP}}} \leftarrow v^{\overline{\text{RMP}}}$
15: **end function**

Remark 2 If we solve (PP) using a global MINLP solver such as BARON, we obtain, if available, lower (l) and upper (u) bounds on ζ at every iteration of the branch-and-bound algorithm. Using these bounds, we can generate new columns and compute valid bounds on $v^{\overline{\text{MP}}}$ without solving (PP) to optimality as follows: A column is added if $u < 0$. In line 10 of Algorithm 1, $\bar{f}_k$ is computed using u, i.e. $\bar{f}_k \leftarrow u + \pi^\top D y^* + \mu$. Then the following bounds can be computed:

$$v^{\overline{\text{RMP}}} + l \leq v^{\overline{\text{RMP}}} + \zeta \leq v^{\overline{\text{MP}}} \leq v^{\overline{\text{RMP}}}. \tag{8}$$

Only in the last iteration, (PP) has to be solved to $\bar{\epsilon}$-optimality in order to achieve a desired optimality gap. Not solving (PP) to optimality at every iteration can mitigate the impact of the tailing-off effect in solving the MINLP and significantly speed up the overall algorithm.

4.2 Branch-and-bound

If the solution of ($\overline{\text{MP}}$) is integer feasible, it is also the optimal solution to (MP); remarkably, our computational experiments (see Sect. 6) show that solving ($\overline{\text{MP}}$) often returns integer feasible solutions. In general, however, the solution may not satisfy the integrality restrictions, in which case we have to apply branch-and-bound. It is well known that branching on the λ-variables leads to unbalanced branch-and-bound trees; hence, it is recommended to design branching rules based on the original variables [49]. We comment, whenever appropriate, on

branching rules in the case studies in Sect. 6 as they often have to be tailored to the specific application.

An outline of the branch-and-price algorithm is shown in Algorithm 2. Here, $\mathcal{N}$ denotes the set of nodes generated in the branch-and-bound tree, which is initialized with the root node 1. Furthermore, we start with a nonempty set of feasible columns $\widehat{\mathcal{K}}$. At each node n, we solve the linear relaxation of the master problem associated with that node, denoted by $(\overline{\text{MP}})_n$, using the corresponding pricing problem $(\text{PP})_n$. The solution of $(\overline{\text{MP}})_n$ can be used to update the overall lower bound LB^{MP}, and if it is integer feasible, it also provides an upper bound on v^{MP}; otherwise, we can solve $(\text{RMP})_n$, which considers the integrality constraints, to obtain a feasible solution to (MP) and hence an upper bound. The incumbent solution $(\hat{x}^*, \hat{\lambda}^*)$ is updated every time a new feasible solution is found. The algorithm terminates when an optimality gap smaller than ϵ is reached; otherwise, branching rules are applied to update the node set $\mathcal{N}$, and the next node n to be evaluated is selected.

Algorithm 2 Branch-and-price algorithm for solving (MP).

1: Set tolerance ϵ
2: Initialize: $\text{LB}^{\text{MP}} \leftarrow -\infty$, $\text{UB}^{\text{MP}} \leftarrow \infty$, $\mathcal{N} \leftarrow \{1\}$, $n \leftarrow 1$, $\widehat{\mathcal{K}}$
3: **while** $|\mathcal{N}| > 0$ **do**
4: SOLVERELAXEDMP($(\overline{\text{MP}})_n$, $(\text{PP})_n$, $\widehat{\mathcal{K}}$)
5: Update lower bound LB^{MP}
6: **if** solution not integer feasible **then**
7: Solve $(\text{RMP})_n$, obtain x^*, λ^*, v^{RMP}
8: **end if**
9: Update upper bound UB^{MP} and incumbent solution $(\hat{x}^*, \hat{\lambda}^*)$
10: **if** $\text{UB}^{\text{MP}} - \text{LB}^{\text{MP}} > \epsilon$ **then**
11: Apply branching rules and update $\mathcal{N}$
12: Select next node n
13: **else**
14: **return** $\hat{x}^*$, $\hat{\lambda}^*$, LB^{MP}, UB^{MP}
15: **end if**
16: **end while**

Remark 3 As mentioned in Remark 2, a lower bound on $v^{(\overline{\text{MP}})_n}$ is obtained at every iteration of SOLVERELAXEDMP($(\overline{\text{MP}})_n$, $(\text{PP})_n$, $\widehat{\mathcal{K}}$). This information can be used to potentially prune the node before the column generation algorithm terminates.

Remark 4 Solving $(\text{RMP})_n$ after solving $(\overline{\text{MP}})_n$ at every node (line 7 of Algorithm 2) is not required for convergence; however, it provides upper bounds that can help reduce the total number of nodes that need to be evaluated. In fact, $(\text{RMP})_n$ can be solved at any column generation iteration, but this is typically not worth the computational effort.

Remark 5 For ease of exposition, a global set of columns, $\widehat{\mathcal{K}}$, is used in Algorithm 2. However, it is usually beneficial to consider node-specific column sets. For example, branching can render some of the columns infeasible, in which case they should be removed from the set.

5 Decomposable pricing problems

Most problems of interest that can be effectively solved using branch-and-price have a decomposable structure. Specifically, this means that the pricing problem decomposes into multiple

smaller subproblems that can be solved independently and in parallel. Such an MINLP is of the following form:

$$\underset{x,y,z}{\text{minimize}} \quad c^\top x + \sum_{i\in\mathcal{I}} f_i(y_i, z_i) \tag{9a}$$

$$\text{subject to} \quad Ax + \sum_{i\in\mathcal{I}} D_i\, y_i \geq b \tag{9b}$$

$$g_i(y_i, z_i) \leq 0 \quad \forall\, i \in \mathcal{I} \tag{9c}$$

$$y_i^{\min} \leq y_i \leq y_i^{\max} \quad \forall\, i \in \mathcal{I} \tag{9d}$$

$$x \in \mathbb{R}^m_+ \times \mathbb{Z}^{\bar{m}}_+ \tag{9e}$$

$$y_i \in \mathbb{Z}^{p_i},\ z_i \in \mathbb{R}^{q_i} \times \mathbb{Z}^{\bar{q}_i} \quad \forall\, i \in \mathcal{I}, \tag{9f}$$

where $\mathcal{I}$ denotes the set of subproblems. One can see that the problem decomposes into $|\mathcal{I}|$ independent subproblems if constraints (9b) are removed.

Following an analogous derivation as in Sect. 3, we reformulate (9) into the following master problem:

$$v^{\text{MP}} = \min_{x,\lambda} \quad c^\top x + \sum_{i\in\mathcal{I}} \sum_{k\in\mathcal{K}_i} \lambda_{ik}\, \bar{f}_{ik} \tag{10a}$$

$$\text{s.t.} \quad Ax + \sum_{i\in\mathcal{I}} D_i \sum_{k\in\mathcal{K}_i} \lambda_{ik}\, \bar{y}_{ik} \geq b \tag{10b}$$

$$\sum_{k\in\mathcal{K}_i} \lambda_{ik} = 1 \quad \forall\, i \in \mathcal{I} \tag{10c}$$

$$x \in \mathbb{R}^m_+ \times \mathbb{Z}^{\bar{m}}_+ \tag{10d}$$

$$\lambda_{ik} \in \{0, 1\} \quad \forall\, i \in \mathcal{I},\ k \in \mathcal{K}_i, \tag{10e}$$

where $\mathcal{K}_i$ denotes the set of feasible columns in subproblem i, and each $k \in \mathcal{K}_i$ is associated with a solution $\bar{y}_{ik}$ and an optimal cost $\bar{f}_{ik}$. The corresponding pricing subproblem i is as follows:

$$\zeta_i := \min_{y_i, z_i} \quad f_i(y_i, z_i) - \pi^\top D_i\, y_i - \mu_i \tag{11a}$$

$$\text{s.t.} \quad g_i(y_i, z_i) \leq 0 \tag{11b}$$

$$y_i^{\min} \leq y_i \leq y_i^{\max} \tag{11c}$$

$$y_i \in \mathbb{Z}^{p_i},\ z_i \in \mathbb{R}^{q_i} \times \mathbb{Z}^{\bar{q}_i}. \tag{11d}$$

In the decomposable case, the pricing subproblems are solved independently and columns are generated for each subproblem. The optimal value $v^{\overline{\text{MP}}}$ can then be bounded as follows:

$$v^{\overline{\text{RMP}}} + \sum_{i\in\mathcal{I}} \zeta_i \leq v^{\overline{\text{MP}}} \leq v^{\overline{\text{RMP}}}. \tag{12}$$

Note that not all ζ_i have to be nonpositive, but their sum will be.

6 Computational experiments

In this section, the computational performance of solving many problem instances from four different case studies using the proposed branch-and-price algorithm is compared to

the performance when solving the full-space problem using a state-of-the-art, off-the shelf global MINLP solver, BARON 19.7.9 [24]. All problems are solved using 25 cores on the Mesabi cluster of the Minnesota Supercomputing Institute, a Linux cluster equipped with a set of 2.5 GHz Intel Hasewell E5-2680v3 processors. For solving full-space problems without decomposition, BARON is used with the Threads option set to 25, allowing for the MILP subsolver (CPLEX) to make use of parallelization to speed up solution times. When problems are solved using branch-and-price, CPLEX 12.8 is used to solve restricted master problems, while BARON with a single thread is used to solve pricing subproblems in parallel. All computations are implemented in Julia using the JuMP modeling language, version 0.17.1 [51]. The stopping criteria used for all instances are a 0.1When the desired optimality gap is reached, we report wall time to reach that gap, and when the maximum wall time is passed, we report the optimality gap achieved at that time. For each case study presented, model formulations for the full-space problem, master problem, and pricing subproblems, as well as distributions used for the generation of random parameters, are given in the supplementary material. The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

6.1 Example 1: cutting circles from rectangles

This problem considers cutting out a set of circles $\mathcal{J}$ of different sizes from a set of given rectangles $\mathcal{R}$, which are also of different sizes. It decides which rectangles should be used and which circles should be cut from these rectangles in order to minimize the total trim loss, which is the area of the used rectangles that was not cut into circles. More generally, it is a simple nonlinear example of a generalized assignment problem, where tasks need to be assigned to different available resources, and can also be thought of as a nonlinear example of the cutting stock problem, one of the motivating examples for the development of branch-and-price [52]. This problem was originally considered by Rebennack et al. [39], where it is solved using column enumeration, which differs from the column generation approach used in this paper in that the master problem (5) is solved considering the full set of columns $\mathcal{K}$, instead of a dynamically updated subset of columns.

This problem decomposes into one pricing subproblem for each rectangle $r \in \mathcal{R}$ which decides which circles should be cut from each respective rectangle. These decisions are returned as a column to the master problem, which decides which rectangles should actually be used given complicating assignment constraints that state that each circle must be assigned to exactly one rectangle. For this case study, the master problem convexity constraint (5c) can be relaxed to an inequality, where it is possible to not choose any column for a given rectangle if no circles are cut from that rectangle in the optimal solution. The overall size of the problem can be increased in two ways: increasing the number of rectangles, which increases the number of subproblems in the branch-and-price algorithm and should have minimal effect on solution times when subproblems are parallelized, and increasing the number of circles, which increases the size of subproblems in the branch-and-price algorithm and should result in an increase in solution times. Computational results for 25 instances of varying problem size are presented in Table 1. Note that for this case study, random problem instances are generated in the same way as in the previous work analyzing this problem.

For this case study, a feasible initial set of columns is required to ensure feasibility of the master problem in early iterations. The initial columns chosen are feasible (but likely suboptimal) cases where a single circle is cut from a rectangle, as optimal column costs can be trivially calculated for these cases as the rectangle area minus the circle area. The "number

of columns generated" information presented in Table 1 includes these initial columns. To ensure a fair comparison between the two methods, the full-space problem is also initialized with a feasible (but likely suboptimal) solution where different rectangles are each assigned one circle. Nonetheless, for many of the problem instances with a large number of rectangles, BARON is unable to determine an upper bound on the objective for the problem within the time allotted. Overall, the performance of the branch-and-price algorithm is clearly superior to solving the full-space problem in BARON for the problems tested in this case study, as the full-space model cannot be solved to optimality in under 10,000 s for any instance, and only returns solutions with very large optimality gaps after that time. Conversely, 16 of the 25 instances are solved to global optimality using branch-and-price, and the worst gap returned from branch-and-price (78.8%) is still better than the best gap returned from the full-space solution (95.1%). We also note that, as expected, our solution times for the branch-and-price algorithm seem to scale greatly with the number of circles, or subproblem size, but only minimally with the number of rectangles, or number of subproblems. Note that 12 of the 25 instances tested in this case study require branching to solve to global optimality; here, the branching strategy used is to find the circle of the largest size with a noninteger assignment to a rectangle, and to branch on this variable.

6.2 Example 2: multiperiod capacity planning with congestion effects

This problem considers capacity planning at a facility that can use a set of candidate production units $\mathcal{K}$ to produce a set of demanded products $\mathcal{J}$. Demands for these products change over time throughout a set of time periods $\mathcal{T}$. Demands are placed into a production queue, and congestion within this queue results in uncertainties in lead times for meeting demands. It decides which units to build during which time points in order to meet demands at minimum cost. Nonlinear chance constraints are used to ensure lead times remain below a given threshold with some probability. This problem is a multiperiod extension of the capacity planning with congestion problem considered by Rajagopalan and Yu [53].

In extending the original formulation to the multiperiod case, we define two different sets of variables which are related to a unit being available for use: z_{kt}, which corresponds to the number of units k newly built during time period t, and y_{kt}, which corresponds to the number of units k that are actively used during time period t. These variables are related according to the following conservation equation:

$$y_{k,t} \leq y_k^0 + \sum_{t'=1}^{t} z_{kt'} \quad \forall k \in \mathcal{K},\ t \in \mathcal{T}, \tag{13}$$

where y_k^0 is the number of units k initially present. This problem decomposes into one pricing subproblem for each time period $t \in \mathcal{T}$ which decides the assignment of orders to production units such that demands are met within the respective time periods. Subproblems are linked to the master problem via number of units actively used at each time period, and the master problem then determines which units should be built when subject to constraints (13). We again increase the overall size of the problem in two ways: by increasing the number of subproblems (increasing $|\mathcal{T}|$) and by increasing the size of subproblems (increasing $|\mathcal{K}|$ and $|\mathcal{J}|$). Computational results for 20 instances of varying problem size are presented in Table 2. Note that for this case study, random parameters for problem instances are generated in the same way as in the previous work analyzing this problem, with the exception that demand

Table 1 Computational performance of 25 instances of cutting circles from rectangles

$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{R}\rvert$	Branch-and-price nodes	Column generation iterations	Number of columns generated	Branch-and-price time (s) or *gap*(%)	Branch-and-price objective	Full-space gap (%)	Full-space objective
6	20	3	7	122	335	10.2	95.1	11.2
6	40	1	5	185	377	9.4	98.5	12.4
6	60	1	5	307	576	10.3	99.9	13.0
6	80	3	7	448	946	9.5	N/A	Not found
6	100	1	5	507	733	8.0	99.5	12.5
7	20	9	19	161	3229	11.9	98.9	14.3
7	40	13	21	323	4047	10.2	99.3	12.1
7	60	1	4	365	1489	9.5	N/A	Not found
7	80	3	6	509	2391	9.9	N/A	Not found
7	100	1	3	674	1112	9.3	N/A	Not found
8	20	1	5	182	1390	10.5	98.7	13.2
8	40	1	4	309	1544	10.2	99.5	13.0
8	60	1	6	477	2562	9.9	N/A	Not found
8	80	9	12	639	*0.67*	9.2	N/A	Not found
8	100	7	8	768	*2.3*	9.3	N/A	Not found

Table 1 continued

$\|\mathcal{J}\|$	$\|\mathcal{R}\|$	Branch-and-price nodes	Column generation iterations	Number of columns generated	Branch-and-price time (s) or *gap*(%)	Branch-and-price objective	Full-space gap (%)	Full-space objective
9	20	7	20	281	9962	9.9	99.6	19.2
9	40	11	20	421	*4.7*	9.8	N/A	Not found
9	60	5	10	595	*3.3*	10.2	N/A	Not found
9	80	1	6	715	*48.5*	9.6	N/A	Not found
9	100	5	7	986	*1.7*	9.2	N/A	Not found
10	20	1	8	254	2075	9.7	N/A	Not found
10	40	1	9	459	9878	8.6	99.9	16.0
10	60	3	8	698	*6.0*	9.1	N/A	Not found
10	80	1	6	859	*64.8*	8.9	N/A	Not found
10	100	1	8	1056	*78.8*	8.7	N/A	Not found

Optimality gaps after 10,000 s are reported when problem is not solved within this time

Table 2 Computational performance of 20 instances of capacity planning with congestion

$\|\mathcal{K}\|$	$\|\mathcal{J}\|$	$\|\mathcal{T}\|$	Column generation iterations	Number of columns generated	Branch-and-price time (s) or *gap*(%)	Branch-and-price objective ($*10^3$)
5	10	6	10	50	338	5.8
5	10	12	19	138	917	5.8
5	10	24	22	196	1741	9.0
5	10	36	28	233	2682	10.9
8	15	6	34	165	5886	7.7
8	15	12	27	303	*0.36*	9.6
8	15	24	18	346	*18.8*	14.0
8	15	36	10	313	*46.4*	15.4
10	20	6	2	18	389	22.7
10	20	12	8	57	1970	19.6
10	20	24	6	83	3874	26.0
10	20	36	7	114	3661	22.4
12	25	6	2	15	485	22.7
12	25	12	2	37	448	25.7
12	25	24	7	78	3482	27.8
12	25	36	3	106	2145	31.8
15	30	6	3	15	772	30.7
15	30	12	3	39	705	38.4
15	30	24	13	125	4652	29.8
15	30	36	3	90	3434	37.3

Optimality gaps after 10,000 s are reported when problem is not solved within this time

is modified to ensure that it increases, on average, over time, and installation costs decrease over time in a manner consistent with typical notions of time value of money.

To obtain an initial set of columns for the branch-and-price algorithm, the pricing subproblems were solved in the case where all dual variables are set to zero. We do not explicitly define starting values for the variables in either the initial subproblems or in the full-space problem. While subproblems are solved relatively quickly, we note that in all instances tested, we are unable to obtain even a feasible solution when solving the full-space problem using BARON. For this case study it is again seen that the branch-and-price approach is superior, with feasible solutions found in all instances and globally optimal solutions found in 17 of the 20 instances tested. Interestingly, we find that solution times tend to increase more strongly with number of subproblems than size of subproblems, which is the opposite of what is expected. We note that this is likely due to the fact that instances with smaller subproblem sizes seem to require more column generation iterations to converge, and that the time required per iteration of column generation seems to better conform to the expected trends. We also note that in all instances tested only column generation is needed to solve the problems, as the root node solutions of the relaxed master problem are integer. This observation will also be true for the next two case studies tested.

6.3 Example 3: multiscenario synthesis of integrated water networks

This problem considers the optimal design of a waste water network with process units which add impurities to the water, as well as remediation units which remove these impurities. It considers uncertainties in the amount of impurities added by the process units and removed by remediation units, which are manifested through a set of different scenarios $\mathcal{S}$. It decides which pipes should be used to connect different units and which remediation units should be installed in order to satisfy constraints on the allowable amount of impurities in the process inputs and system output at minimum cost. The problem considered here is an adaptation of the original problem presented by Karuppiah and Grossmann [54], using the same network structure but assuming that allowable unit and pipe sizes come from a discrete set $\mathcal{V}$, rather than being allowed to vary continuously.

To solve this problem using branch-and-price, we introduce copies of the design variables for each scenario $s \in \mathcal{S}$. These variables are constrained by non-anticipativity constraints, which take into account the fact that we do not know *a priori* which of the scenarios will actually be realized when making design decisions, and as such designs must be the same for all scenarios:

$$y_{ivs} = y_{ivs'} \quad \forall i \in \mathcal{I},\ v \in \mathcal{V},\ (s, s') \in \mathcal{S}, \tag{14}$$

where y_{ivs} refers to a design decision to build unit or pipe i of size v in scenario s. Note that the set $\mathcal{I}$ refers to all different pipes and units. The problem then decomposes to one pricing subproblem for each scenario $s \in \mathcal{S}$ which decides the optimal network design for a specific scenario. The master problem then considers the weights of all scenarios and finds a design which minimizes the expected cost subject to non-anticipativity constraints (14). The number of subproblems is increased by increasing $|\mathcal{S}|$, and the subproblem size is increased by increasing $|\mathcal{V}|$. Computational results for 20 instances of varying size are presented in Table 3. Note that for this case study, random parameters are used for different scenarios in different instances based on the ranges used in the previous work analyzing this problem.

For this problem, the knowledge of the non-anticipativity constraints is used to modify the algorithm for generating columns. Whenever a column with negative reduced cost is generated for one scenario, a cost for that column from each scenario is found by solving each pricing subproblem with design variables fixed at the value given by the new column. As such, this problem only consists of one set of columns, instead of multiple sets corresponding to each respective subproblem. Initial columns are generated using this strategy with dual variables equal to zero. This approach ensures that the master problem is always feasible. However, it is possible that columns generated from one scenario may be infeasible for another. These infeasible columns are kept in a separate set and integer cuts are introduced to the subproblems to ensure that they are not generated again. Note that the number of infeasible columns increases as the number of discrete unit sizes increases, which makes sense as this introduces a larger number columns that may be infeasible for a scenario. Again, the branch-and-price algorithm outperforms solving the full-space model using BARON for all instances. However, the time equired to solve the problem clearly scales very poorly with subproblem size in this case, to the point where when $|\mathcal{V}| \geq 3$, no instance is solved to optimality within the time limit. Branch-and-price still finds a feasible solution in all cases, which is not the case when solving the full-space problem, w hich only finds a feasible solution in 10 out of 20 instances.

Table 3 Compuational performance of 20 instances of multiscenario water network synthesis

$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{S}\rvert$	Column generation iterations	Number of columns generated	Infeasible columns generated	Branch-and-price time (s) or *gap* (%)	Branch-and-price objective ($*10^5$)	Full space gap (%)	Full space objective ($*10^5$)
1	5	2	3	1	172	6.9	37.2	6.9
1	10	7	39	1	1451	6.6	73.1	6.9
1	15	7	34	1	2197	6.9	N/A	Not found
1	20	7	38	1	1537	6.6	N/A	Not found
1	25	3	30	1	618	6.7	N/A	Not found
2	5	12	47	1	2857	6.7	68.9	6.7
2	10	16	114	1	4909	6.5	70.6	6.5
2	15	9	66	1	5254	6.7	71.2	6.7
2	20	17	116	1	*1.9*	6.7	72.5	6.7
2	25	14	104	1	9456	6.7	72.4	6.7
3	5	19	91	3	*13.6*	6.9	70.9	7.2
3	10	13	125	7	*38.8*	6.7	73.6	6.9
3	15	11	137	5	*46.2*	6.9	N/A	Not found
3	20	8	127	2	*28.1*	6.6	N/A	Not found
3	25	6	129	1	*34.3*	6.7	N/A	Not found
4	5	25	73	29	*52.0*	6.7	74.8	7.0
4	10	14	99	19	*12.7*	6.8	N/A	Not found
4	15	10	121	8	*41.5*	6.7	N/A	Not found
4	20	8	122	18	*15.0*	6.6	N/A	Not found
4	25	12	227	22	*33.2*	6.6	N/A	Not found

Optimality gaps after 10,000 s are reported when problem is not solved within this time

6.4 Example 4: multiscenario design of multiproduct batch plants

This problem considers the optimal design of a batch plant with a set of stages $\mathcal{J}$ that can produce multiple products. It considers uncertainties in the demands for each product, as well as in the operating parameters of each of the different batch stages, which are manifested through a set of different scenarios $\mathcal{S}$. It decides the number of batch units to include at each stage, as well as the batch sizes and times for each product, such that demands are met at minimum cost. The problem presented here is an adaptation of a problem presented in [7], assuming that the volume of the batch unit is fixed instead of being a decision variable, and adding a nonlinear operating cost term to the objective function.

To solve this problem using branch-and-price, we introduce copies of the variable corresponding to number of batch units built for each scenario $s \in \mathcal{S}$. Like the previous example, these variables are constrained by non-anticipativity constraints. As such, we use the same modified algorithm for generating columns. To combat the possibility of generating infeasible columns, scenarios are grouped together into a set of bunches $\mathcal{B}$ such that each bunch roughly contains an equal amount of "easy-to-satisfy" scenarios where demands are low and "hard-to-satisfy" scenarios where demands are high. Additionally, we add logical cuts to all subproblems when an infeasible column is generated which state that if a certain design is infeasible, any new design must build more batch units than in the infeasible design in at least one stage:

$$N_j + N_{jc}^{inf} z_{jc} \geq N_{jc}^{inf} + 1 \quad \forall\, j \in \mathcal{J}, c \in \mathcal{C}_i, \tag{15}$$

$$\sum_{j \in \mathcal{J}} z_{jc} \leq |\mathcal{J}| - 1 \quad \forall\, c \in \mathcal{C}_i, \tag{16}$$

where N_j is the number of units built for stage j, N_{jc}^{inf} is the number of units built for stage j in infeasible column c, z_{jc} is a binary variable which is 0 only if $N_j > N_{jc}^{inf}$, and $\mathcal{C}_i$ is a set of infeasible columns. The problem then decomposes into one pricing subproblem for each scenario bunch $b \in \mathcal{B}$ which decides the optimal system design for the corresponding group of scenarios. The master problem then considers the weights of all scenario bunches and finds a design which minimizes the expected cost subject to non-anticipativity constraints. The number of subproblems is increased by increasing $|\mathcal{B}|$, where each bunch contains 5 scenarios, and the size of subproblems is increased by increasing $|\mathcal{J}|$. Computational results for 20 instances of varying size are presented in Table 4. Note that for this case study, the randomly generated stochastic parameters for different instances are generated using the deterministic parameter values from the previous work as a starting point for random perturbations.

For this problem, unlike the other three examples presented, the branch-and-price approach performs very similarly to solving the full-space problem using BARON, particularly as the subproblem size increases. For all problems, both approaches are able to find the same objective upper bound. We note that for some instances, BARON is able to find a slightly better objective value than the branch-and-price approach, although in all instances these differences are within either the 0.1% gap used as a global optimality stopping criteria, or within the reported gap after 10,000 s. It is also apparent that as the number of stages increases, the number of infeasible columns generated also increases, contributing to the added difficulty for solving this problem using branch and price. However, we do note the number of infeasible columns generated is reduced by bunching columns and adding logical

Table 4 Compuational performance of 20 instances of multiscenario batch system design

$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{B}\rvert$	Column generation iterations	Number of columns generated	Infeasible columns generated	Branch-and-price time (s) or *gap*(%)	Branch-and-price objective ($*10^5$)	Full-space gap (%)	Full-space objective ($*10^5$)
3	10	9	47	0	1934	8.3	1.11	8.3
3	20	9	42	25	3120	8.6	8.03	8.6
	30	7	58	16	2471	8.4	1.31	8.4
3	40	5	45	0	1349	8.5	1.44	8.5
3	50	5	43	39	2271	8.6	0.30	8.6
4	10	39	57	26	*0.13*	16.0	3.77	16.0
4	20	8	41	35	2458	17.2	1.14	17.2
4	30	9	101	70	*1.42*	15.6	3.12	15.6
4	40	12	69	91	*0.13*	16.7	0.43	16.7
4	50	6	61	53	3449	16.4	0.22	16.4
5	10	25	119	79	*2.49*	17.8	2.63	17.8
5	20	25	237	146	*1.36*	17.5	2.99	17.5
5	30	15	187	96	*4.00*	18.0	4.46	18.0
5	40	12	183	133	*2.47*	17.9	1.64	17.9
5	50	10	139	166	*2.77*	18.9	2.61	18.9
6	10	33	149	155	*5.33*	21.9	4.03	21.9
6	20	32	238	301	*4.78*	22.5	3.16	22.5
6	30	18	242	192	*3.16*	23.1	3.01	23.1
6	40	14	212	274	*4.44*	23.0	2.26	23.0
6	50	12	221	238	*3.29*	24.4	4.86	24.3

Optimality gaps after 10,000 s are reported when problem is not solved within this time

infeasibility cuts, as this number can be as much as an order of magnitude greater when these options are not used.

7 Conclusions

Applied branch-and-price to a class of nonconvex MINLPs with linear complicating constraints and integer linking variables. We exploit the structure of the problem to construct a Dantzig-Wolfe reformulation via a discretization approach, which then allows the problem to be solved using a branch-and-price scheme. This approach is especially effective in cases where the pricing problem decomposes into multiple small subproblems such that solving each subproblem using a global MINLP solver is considerably more tractable than solving the original full-space MINLP. Through several case studies, we have shown that many relevant problems directly fall or can be reformulated into the given class of MINLPs, and have demonstrated the computational feasibility of the proposed algorithm. In most tested model instances, the branch-and- price algorithm clearly outperforms solving the full-space problem directly using a global MINLP solver, often achieving orders-of-magnitude speedups.

Acknowledgements The authors gratefully acknowledge financial support from the University of Minnesota and the Minnesota Supercomputing Institute (MSI) at the University of Minnesota for providing resources that contributed to the research results reported within this paper. We also thank Angela Flores-Quiroz for insightful discussions on our work.

References

1. Lübbecke, M.E., Desrosiers, J.: Selected topics in column generation. Oper. Res. **53**(6), 1007–1023 (2005). https://doi.org/10.1287/opre.1050.0234
2. Singh, K.J., Philpott, A.B., Wood, R.K.: Dantzig–Wolfe decomposition for solving multistage stochastic capacity-planning problems. Oper. Res. **57**(5), 1271–1286 (2009). https://doi.org/10.1287/opre.1080.0678
3. Nowak, I., Breitfeld, N., Hendrix, E.M., Njacheun-Njanzoua, G.: Decomposition-based inner- and outer-refinement algorithms for global optimization. J. Glob. Optim. **72**(2), 305–321 (2018). https://doi.org/10.1007/s10898-018-0633-2
4. Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large-scale traveling-salesman problem. Oper. Res. **2**(4), 393–410 (1954)
5. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bull. Am. Math. Soc. **64**(5), 275–278 (1958)
6. Jünger, M., Liebling, T., Naddef, D., Nemhauser, G.L., Pulleyblank, W., Reinelt, G., Rinaldi, G., Wolsey, L.A.: 50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art. Springer (2009)
7. Grossmann, I.E., Sargent, R.W.: Optimum design of multipurpose chemical plants. Ind. Eng. Chem. Process Des. Dev. **18**(2), 343–348 (1979). https://doi.org/10.1021/i260070a031
8. Gupta, O.K., Ravindran, A.: Branch and bound experiments in convex nonlinear integer programming author. Manag. Sci. **31**(12), 1533–1546 (1985)
9. Stubbs, R.A., Mehrotra, S.: A branch-and-cut method for 0–1 mixed convex programming. Math. Progr., Ser. B **86**(3), 515–532 (1999). https://doi.org/10.1007/s101070050103
10. Geoffrion, A.M.: Generalized benders decomposition. J. Optim. Theory Appl. **10**(4), 237–260 (1972). https://doi.org/10.1097/ACI.0000000000000254
11. Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Progr. **36**, 307–339 (1986). https://doi.org/10.1007/BF02592064
12. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Math. Progr. **66**, 327–349 (1994)

13. Quesada, I., Grossmann, I.E.: An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. Comput. Chem. Eng. **16**(10–11), 937–947 (1992). https://doi.org/10.1016/0098-1354(92)80028-8
14. Westerlund, T., Pettersson, F.: An extended cutting plane method for solving convex MINLP problems. Comput. Chem. Eng. **19**(Suppl. 1), 131–136 (1995). https://doi.org/10.1016/0098-1354(95)87027-X
15. Kronqvist, J., Lundell, A., Westerlund, T.: The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. J. Glob. Optim. **64**(2), 249–272 (2016). https://doi.org/10.1007/s10898-015-0322-3
16. Grossmann, I.: Review of nonlinear mixed-integer and disjunctive programming techniques. Optim. Eng. **3**(3), 227–252 (2002). https://doi.org/10.1023/A:1021039126272
17. Bonami, P., Kilinç, M., Linderoth, J.: Algorithms and software for convex mixed integer nonlinear programs. In: Mixed Integer Nonlinear Programming, pp. 1–39. Springer (2012). https://doi.org/10.1007/978-1-4614-1927-3
18. Kronqvist, J., Bernal, D.E., Lundell, A., Grossmann, I.E.: A Review and Comparison of Solvers for Convex MINLP, vol. 20. Springer (2019). https://doi.org/10.1007/s11081-018-9411-8
19. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: part I—convex underestimating problems. Math. Progr. **10**(1), 147–175 (1976). https://doi.org/10.1007/BF01580665
20. Ryoo, H.S., Sahinidis, N.V.: A branch-and-reduce approach to global optimization. J. Glob. Optim. **8**(2), 107–138 (1996). https://doi.org/10.1007/bf00138689
21. Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: a theoretical and computational study. Math. Progr. **591**, 563–591 (2004)
22. Androulakis, I.P., Maranas, C.D., Floudas, C.A.: αBB: A global optimization method for general constrained nonconvex problems. J. Glob. Optim. **7**(4), 337–363 (1995). https://doi.org/10.1007/BF01099647
23. Adjiman, C.S., Androulakis, I.P., Floudas, C.A.: Global optimization of mixed-integer nonlinear problems. AIChE J. **46**(9), 1769–1797 (2000)
24. Kılınç, M.R., Sahinidis, N.V.: Exploiting integrality in the global optimization of mixed-integer nonlinear programming problems with BARON. Optim. Methods Softw. **33**(3), 540–562 (2018). https://doi.org/10.1080/10556788.2017.1350178
25. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tighteningtechniques for non-convex MINLP. Optim. Methods Softw. **24**(4–5), 597–634 (2009). https://doi.org/10.1080/10556780903087124
26. Lin, Y., Schrage, L.: The global solver in the LINDO API. Optim. Methods Softw. **24**(4–5), 657–668 (2009). https://doi.org/10.1080/10556780902753221
27. Misener, R., Floudas, C.A.: ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. J. Glob. Optim. **59**(2–3), 503–526 (2014). https://doi.org/10.1007/s10898-014-0166-2
28. Vigerske, S., Gleixner, A.: SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. Optim. Methods Softw. **33**(3), 563–593 (2018). https://doi.org/10.1080/10556788.2017.1335312
29. Burer, S., Letchford, A.N.: Non-convex mixed-integer nonlinear programming: a survey. Surv. Oper. Res. Manag. Sci. **17**(2), 97–106 (2012). https://doi.org/10.1016/j.sorms.2012.08.001
30. Boukouvala, F., Misener, R., Floudas, C.A.: Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization. CDFO. Eur. J. Oper. Res. **252**(3), 701–727 (2016). https://doi.org/10.1016/j.ejor.2015.12.018
31. Guignard, M.: Lagrangean relaxation. Top **11**(2), 151–200 (2003). https://doi.org/10.1007/BF02579036
32. Watson, J.P., Woodruff, D.L.: Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. Comput. Manag. Sci. **8**(4), 355–370 (2011). https://doi.org/10.1007/s10287-010-0125-4
33. Lotero, I., Trespalacios, F., Grossmann, I.E., Papageorgiou, D.J., Cheon, M.S.: An MILP-MINLP decomposition method for the global optimization of a source based model of the multiperiod blending problem. Comput. Chem. Eng. **87**, 13–35 (2016). https://doi.org/10.1016/j.compchemeng.2015.12.017
34. Lara, C.L., Trespalacios, F., Grossmann, I.E.: Global optimization algorithm for capacitated multi-facility continuous location-allocation problems. J. Glob. Optim. **71**(4), 871–889 (2018). https://doi.org/10.1007/s10898-018-0621-6
35. Elsido, C., Martelli, E., Grossmann, I.E.: A bilevel decomposition method for the simultaneous heat integration and synthesis of steam/organic Rankine cycles. Comput. Chem. Eng. **128**, 228–245 (2019). https://doi.org/10.1016/j.compchemeng.2019.05.041
36. Li, X., Tomasgard, A., Barton, P.I.: Nonconvex generalized benders decomposition for stochastic separable mixed-integer nonlinear programs. J. Optim. Theory Appl. **151**(3), 425–454 (2011). https://doi.org/10.1007/s10957-011-9888-1

37. Cao, Y., Zavala, V.M.: A scalable global optimization algorithm for stochastic nonlinear programs. J. Glob. Optim. **75**(2), 393–416 (2019). https://doi.org/10.1007/s10898-019-00769-y
38. Li, C., Grossmann, I.E.: A generalized Benders decomposition-based branch and cut algorithm for two-stage stochastic programs with nonconvex constraints and mixed-binary first and second stage variables. J. Glob. Optim. **75**(2), 247–272 (2019). https://doi.org/10.1007/s10898-019-00816-8
39. Rebennack, S., Kallrath, J., Pardalos, P.M.: Column enumeration based decomposition techniques for a class of non-convex MINLP problems. J. Glob. Optim. **43**(2–3), 277–297 (2009). https://doi.org/10.1007/s10898-007-9271-9
40. Dantzig, G.B., Wolfe, P.: Decomposition principle for linear programs. Oper. Res. **8**(1), 101–111 (1960)
41. Desrosiers, J., Soumis, F., Desrochers, M.: Routing with time windows by column generation. Networks **14**(4), 545–565 (1984). https://doi.org/10.1002/net.3230140406
42. Desrochers, M., Desrosiers, J., Solomon, M.: A new optimization algorithm for the vehicle routing problem with time windows. Oper. Res. **40**(2), 342–354 (1992)
43. Desaulniers, G., Desrosiers, J., Solomon, M.M.: Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In: Essays and Surveys in Metaheuristics, pp. 309–324. Springer (2002)
44. Desrochers, M., Soumis, F.: A column generation approach to the urban transit crew scheduling problem. Transp. Sci. **23**(1), 1–13 (1989). https://doi.org/10.1287/trsc.23.1.1
45. Stojković, M., Soumis, F., Desrosiers, J.: The operational airline crew scheduling problem. Transp. Sci. **32**(3), 232–245 (1998). https://doi.org/10.1287/trsc.1090.0306
46. Ioachim, I., Desrosiers, J., Soumis, F., Bélanger, N.: Fleet assignment and routing with schedule synchronization constraints. Eur. J. Oper. Res. **119**(1), 75–90 (1999). https://doi.org/10.1016/S0377-2217(98)00343-9
47. Bélanger, N., Desaulniers, G., Soumis, F., Desrosiers, J.: Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues. Eur. J. Oper. Res. **175**(3), 1754–1766 (2006). https://doi.org/10.1016/j.ejor.2004.04.051
48. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H.: Branch-and-price: column generation for solving huge integer programs. Oper. Res. **46**(3), 316–329 (1998). https://doi.org/10.1287/opre.46.3.316
49. Vanderbeck, F.: On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Oper. Res. **48**(1), 111–128 (2000). https://doi.org/10.1287/opre.48.1.111.12453
50. Wolsey, L.A.: Integer Programming. Wiley (1998)
51. Lubin, M., Dunning, I.: Computing in operations research using Julia. INFORMS J. Comput. **27**, 237–248 (2015)
52. Gilmore, P.C., Gomory, R.E.: A linear programming approach to the cutting-stock problem. Oper. Res. **9**, 849–859 (1961)
53. Rajagopalan, S., Yu, H.L.: Capacity planning with congestion effects. Eur. J. Oper. Res. **134**(2), 365–377 (2001). https://doi.org/10.1016/S0377-2217(00)00254-X
54. Karuppiah, R., Grossmann, I.E.: Global optimization of multiscenario mixed integer nonlinear programming models arising in the synthesis of integrated water networks under uncertainty. Comput. Chem. Eng. **32**, 145–160 (2008). https://doi.org/10.1016/j.compchemeng.2007.03.007