



# On tackling reverse convex constraints for non-overlapping of unequal circles

Akang Wang<sup>1,2</sup> · Chrysanthos E. Gounaris<sup>1,2</sup> 

Received: 17 December 2019 / Accepted: 6 December 2020 / Published online: 4 January 2021  
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC part of Springer Nature 2021

## Abstract

We study the unequal circle-circle non-overlapping constraints, a form of reverse convex constraints that often arise in optimization models for cutting and packing applications. The feasible region induced by the intersection of circle-circle non-overlapping constraints is highly non-convex, and standard approaches to construct convex relaxations for spatial branch-and-bound global optimization of such models typically yield unsatisfactory loose relaxations. Consequently, solving such non-convex models to guaranteed optimality remains extremely challenging even for the state-of-the-art codes. In this paper, we apply a purpose-built branching scheme on non-overlapping constraints and utilize strengthened intersection cuts and various feasibility-based tightening techniques to further tighten the model relaxation. We embed these techniques into a branch-and-bound code and test them on two variants of circle packing problems. Our computational studies on a suite of 75 benchmark instances yielded, for the first time in the open literature, a total of 54 provably optimal solutions, and it was demonstrated to be competitive over the use of the state-of-the-art general-purpose global optimization solvers.

**Keywords** Non-overlapping constraints · Circle packing · Circular open dimension problem · Branching scheme · Strengthened intersection cuts · Feasibility-based tightening

## 1 Introduction

The circle-circle non-overlapping constraint is imposed to guarantee that two circles (generally of different radius) do not overlap, which can be achieved by requiring that their centers are sufficiently far in terms of Euclidean distance. In particular, the constraint has the mathematical form

$$(a_i - a_j)^2 + (b_i - b_j)^2 \geq (r_i + r_j)^2, \quad (1)$$

where  $(a_i, b_i)$  and  $(a_j, b_j)$  represent the coordinates of the centers of two circles  $i$  and  $j$ , while  $r_i$  and  $r_j$  denote their corresponding radii, respectively.

---

✉ Chrysanthos E. Gounaris  
gounaris@cmu.edu

<sup>1</sup> Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

<sup>2</sup> Center for Advanced Process Decision-making, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Studying this type of constraint has both theoretical and practical interest. From a theoretical perspective, mathematical models with non-overlapping constraints usually have a highly non-convex solution space whose convexification towards a tight relaxation still poses a tremendous challenge for the global optimization community; solving these models to guaranteed optimality remains extremely hard, even for the state-of-the-art solvers. From a practical point of view, circle-circle non-overlapping constraints often appear in many models for cutting and packing applications, and efforts to solve them more efficiently can add significant value to industry.

The first and foremost, archetypal family of problems of interest are the *circle packing problems*. These come in many variants, such as packing identical circles into a rectangular container with the objective of minimizing the container's area [14], or identifying the minimal radius of a circle within which other circles can simultaneously be placed [27], among others. Related applications include container loading, cylinder packing and wireless communication network layout, to name but a few [11]. Another related cutting and packing application is the irregular shape nesting problem, in which one seeks a feasible configuration for a given set of irregular shapes within a rectangular sheet of a fixed width such that no overlap among these shapes exists and the sheet length is minimized. This problem often appears when carving out of metal rolls parts for automobiles, airplanes and other machinery, as well as when cutting leather and fabrics for apparel and upholstery applications. The work of [17] proposed to enforce no overlap between any two irregular shapes by imposing that circles inscribed within one shape do not overlap with circles inscribed within another one, resulting in an optimization model with non-convex quadratic constraints. Using this approach, the authors obtained for the first time optimal nesting solutions to a four polygon problem, using off-the-shelf global optimization solvers to tackle the non-convex quadratically constrained models. Focusing on the circle-inscribing approach for nesting problems, [34] proposed a novel branching scheme on circle-circle non-overlapping constraints and demonstrated the ability to find global optimal nestings to benchmark instances with five polygons, as well as solutions to instances with up to seven polygons under fixed rotation. Both the circle packing problem and the irregular shape nesting problem have been studied extensively, with most of the focus being on the development of heuristics to obtain large-scale packings [4,20]. While heuristic methods are practically valuable for the generation of feasible solutions, they cannot rigorously quantify the optimality gap, and therefore provide no guarantees regarding how much value is left on the table by the packings produced. Arguably, the exact solution of optimization models with non-overlapping constraints is of great practical importance.

Mathematical models with circle-circle non-overlapping constraints fall into the *quadratically constrained quadratic programming* (QCQP) class, which is currently a very active research area in global optimization. Generally speaking, there are two approaches for convexifying QCQPs, namely semi-definite programming (SDP) relaxation and multi-term polyhedral relaxation with reformulation-linearization techniques [7]. The polyhedral approach calls for approximating the convex hull of the non-convex feasible solution space via linear inequalities, while the SDP method is to characterize the convex hull via semi-definite cones. Interested readers are referred to [5] for details. Although the SDP relaxation is usually tighter than the polyhedral one, optimizing an SDP problem is generally computationally less efficient than a linear one. Therefore, general-purpose global solvers [24,32,33] commonly rely on polyhedral relaxations to address QCQPs. Despite many recent advances in global optimization solvers, however, optimizing models with non-overlapping constraints is still incredibly challenging. Our computational studies on circle packing instances (Section 5) also show that general-purpose global solvers could only solve instances of up to 10 circles to guaranteed optimality.

At the same time, there exist only a handful of attempts in the open literature to address such models in a customized approach. The work of [18] considered the problem of packing a fixed number of *identical* circles into a given square with the objective of maximizing the circle radius. The authors conducted a theoretical comparison of several convexification techniques on non-overlapping constraints, including polyhedral and semi-definite relaxations, and assessed their strength theoretically. They pessimistically concluded that the current state-of-the-art bounding techniques within general-purpose global optimization algorithms are only effective for small-size instances (e.g., with up to 10 or so circles). In contrast, several tailored exact algorithms, utilizing interval arithmetic-based branch-and-bound methods, have been shown to be effective at solving to optimality instances of packing an excess of 30 identical circles in the unit square [14]. More specifically, the authors of [19] incorporated into a rectangular branch-and-bound approach optimality conditions satisfied by at least one optimal solution, solving instances of up to 39 circles to optimality. In addition, [22] and [30] proposed methods for eliminating large sets of suboptimal points of the equivalent point packing problem. In [23], the authors employed interval arithmetic to expedite the branch-and-bound algorithm, while the work of [21] presented geometric results concerning the structural properties of such problems of packing circles in the unit square. Finally, for more general settings, the work of [26] proposed to approximate the quadratic function in the left-hand side of a non-overlapping constraint via piecewise linear functions. This approach necessitates the introduction of binary variables to represent the latter, resulting in a mixed-integer linear programming model, and thus might become impractical as the approximation accuracy increases.

This paper presents an extension of the branching strategy on non-overlapping constraints previously developed in [34]. The main idea is rooted in the geometric interpretation of a circle-circle non-overlapping constraint. More specifically, the constraint (1) dictates that any feasible point in the transformed variable space  $(a_i - a_j) - (b_i - b_j)$  should lie outside (or exactly on) the circumference of the circle that has a center with coordinates  $(0, 0)$  and has a radius of  $r_i + r_j$ . Unlike a traditional, general-purpose spatial branch-and-bound based solver, which must generically handle each non-convex constraint by relaxing it individually into its convex relaxation and tighten that relaxation via branching on the domains of the problem's variables, our perspective is to split the feasible, non-convex domain imposed by non-overlapping constraints and enforce feasibility in a more direct fashion. This approach is also clearly distinct from the aforementioned works on identical circles packing [19,21–23,30], since those utilize branching on variable intervals.

More specifically, we will follow an approach that branches on the domain of constraint, rather than the domain of variables. Additionally, we observe that the non-overlapping constraint imposes a *reverse convex* region [15]. It is well-known that reverse convex constraints can induce *intersection cuts* [6,15], which are also called *concavity cuts* [16] in the global optimization community. These cuts are computationally cheap to generate, and they can be utilized to strengthen the model relaxation. In order to further tighten the model relaxation, we also propose three feasibility-based tightening techniques. The distinct contributions of our work can be summarized as follows.

- We develop a customized branch-and-bound (BB) approach for solving problems with unequal circle-circle non-overlapping constraints.

- We apply a generalized version of the intersection cut formula from the seminal paper of [6], as well as propose three types of feasibility-based tightening techniques, to strengthen the BB node relaxations.
- We conduct a comprehensive computational study based on two popular variants of the circle packing problem to demonstrate that our approach achieves superior performance over the use of various state-of-the-art general-purpose global optimization solvers.

The remainder of the paper is organized as follows. In Sect. 2, we provide a concise description of an optimization model with non-convexities that stem from the presence of non-overlapping constraints. In Sect. 3, we discuss how to construct a suitable linear relaxation of such a model that we can use as the basis of our BB algorithm, followed by a presentation of the complete algorithmic procedure. In Sect. 4, we discuss strengthening techniques, including strengthened intersection cuts and feasibility-based tightening. Section 5 presents results on the algorithm’s computational performance as well as a comprehensive comparison against the use of five different state-of-the-art global optimization solvers. Finally, we conclude our paper with some remarks in Sect. 6.

## 2 Problem definition

In this work, we focus on the following problem (2).

$$\begin{aligned}
 & \underset{x \in \mathbb{R}^n}{\text{minimize}} && c^\top x \\
 & \text{subject to} && f_k(x) \leq 0 \quad \forall k \in \mathcal{K} \\
 & && x_i^2 + x_j^2 \geq \rho_{ij}^2 \quad \forall (i, j) \in \mathcal{M} \\
 & && x^L \leq x \leq x^U,
 \end{aligned} \tag{2}$$

where  $f_k(x) : \mathbb{R}^n \mapsto \mathbb{R}$  denotes a convex function, for each applicable index  $k \in \mathcal{K}$ , while  $x_i^2 + x_j^2 \geq \rho_{ij}^2$  represents a circle-circle non-overlapping constraint, for each applicable ordered pair  $(i, j) \in \mathcal{M}$ .<sup>1</sup> Without loss of generality, we assume  $x_i^L \leq x_i^U$ , for all  $i \in \{1, 2, \dots, n\}$ , where these variable bounds are not necessarily finite; that is, we allow for  $x_i^L \rightarrow -\infty$  and/or  $x_i^U \rightarrow +\infty$ .

Since non-overlapping constraints induce a non-convex solution space, problem (2) is a global optimization problem. In this work, we aim to develop exact, custom-built methods for solving this problem to provable global optimality. We remark that this paper focuses on the case where  $\rho_{ij}$  are generally different from each other, i.e., we are interested in packing unequal, a.k.a. *non-identical*, circles. Of course, the methodologies developed herein can still be applied with instances featuring identical circles, though in that case the use of specialized algorithms for packing identical circles is advisable for better overall tractability. We also remark that this paper is dedicated to the case where  $\rho_{ij}$  are constant. The setting where  $\rho_{ij}$  become variables (presumably to be penalized in the objective) constitutes an interesting generalization, and whereas some of the ideas contemplated in this paper, such as the purpose-designed branching scheme and the use of intersection cuts, would be straightforward to adapt, other aspects, such as the feasibility-based tightening techniques, would require further exploration as future work.

<sup>1</sup> For ease of exposition, we simplify the notation of Eq. (1), as follows:  $x_i \leftarrow a_{i'} - a_{j'}$ ,  $x_j \leftarrow b_{i'} - b_{j'}$  and  $\rho_{ij} \leftarrow r_{i'} + r_{j'}$ , where  $(a_\ell, b_\ell)$  and  $r_\ell$  represent the center coordinates and radii, respectively, of two circles  $\ell \in \{i', j'\}$ .

### 3 Solution approach

Our approach is based on the construction of a branch-and-bound tree where linear programming (LP) relaxations are solved at each node. Specifically, the convex domain defined by  $f_k(x) \leq 0$  in (2) is outer-approximated by linear inequalities. The main challenge arises from the non-convexities introduced by the non-overlapping constraints,  $x_i^2 + x_j^2 \geq \rho_{ij}^2$ , which we address in the remainder of this section.

#### 3.1 Customized model relaxation

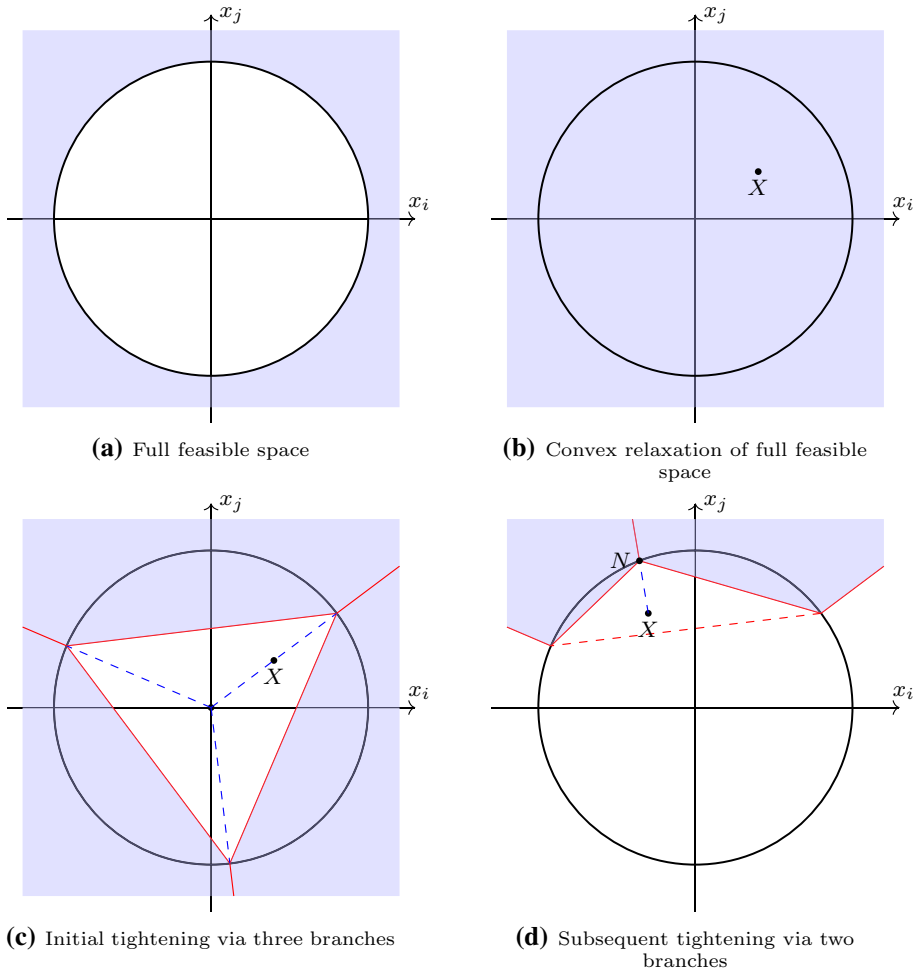
Here, we shall discuss how a suitable linear relaxation is constructed as the basis of the BB search algorithm. First, we adopt the branching scheme from our previous work [34], applied on a non-overlapping constraint from the set  $(i, j) \in \mathcal{M}$ . Let  $\bar{x}$  represent the current optimal solution of the LP relaxation at some BB node, and let  $D_{ij}$  denote a disk that is centered at the origin  $(0, 0)$  and has a radius of  $\rho_{ij}$ . Furthermore, let  $\theta_{ij}$  be the angle between the positive  $x_i$ -axis and the point given by the coordinates  $(\bar{x}_i, \bar{x}_j)$ , and let  $[\theta_{ij}^L, \theta_{ij}^U]$  represent its feasible interval, which is originally equal to  $(0, 2\pi)$ .

In the  $x_i$ - $x_j$  space, the non-overlapping constraint requires that all feasible points project either outside or exactly on the circumference of disk  $D_{ij}$  (Fig. 1a). Initially, dropping this constraint from consideration results in a convex relaxation that is defined by the full space (Fig. 1b). Feasibility of the non-convex constraint can be then be gradually enforced by branching on the implicit domain of variable  $\theta_{ij}$  variable, and by tightening the relaxation in the  $x_i$ - $x_j$  space, as follows.

Let  $X$  denote a solution of the relaxed problem with coordinates  $(\bar{x}_i, \bar{x}_j)$  such that it violates the original constraint. Using this solution as a guide, we can split the circumference of  $D_{ij}$  into three parts (each representing an arc of angle  $2\pi/3$  radians), and we can define a set of three linear constraints (corresponding to one secant and two boundary lines) to form the convex relaxation of the original non-overlapping constraint in each of the three resulting subdomains. As long as they are properly oriented with respect to  $X$  (Fig. 1c), it can be guaranteed that all of the three subdomains exclude the parent solution.

In subsequent violations of the same constraint  $(i, j) \in \mathcal{M}$ , one could further dissect the applicable  $\theta_{ij}$  interval into two parts. Again, as long as the solution  $X$  is used as the guide to define the branching point, one can guarantee that this parent solution is excluded from both resulting subdomains, defining a proper branching strategy to be employed in the course of the BB search process. In particular, one can analytically identify a point  $N$  on the circumference of disk  $D_{ij}$  such that the Euclidean distances between  $X$  and two resulting secant lines are equal, in the hope of producing a more balanced BB tree. Let  $\theta_{ij}^*$  denote the angle in radians between the positive  $x_i$ -axis and the point  $N$ , then the resulting proposed split is  $[\theta_{ij}^L, \theta_{ij}^*]$  and  $[\theta_{ij}^*, \theta_{ij}^U]$  (Fig. 1d).

We highlight that the aforementioned branching scheme proceeds from the perspective of splitting the space induced by the non-convex non-overlapping constraints, which distinguishes our work from the classic idea of branching on each variable's interval. By using the novel scheme, we enforce non-convex constraint feasibility in a direct manner, which is essential in global optimization, yet difficult to achieve in general-purpose global optimization codes for all but the simplest constraint functional forms.



**Fig. 1** The non-overlapping constraint and its dynamically tightened relaxation (adapted from [34])

Given the above relaxation strategy, Eqs.(3)–(7) constitute a relaxed LP formulation to be solved at each node of our BB search process.

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad c^\top x \tag{3}$$

$$\text{subject to} \quad f_k(\tilde{x}) + \nabla f_k(\tilde{x})^\top (x - \tilde{x}) \leq 0 \quad \forall \tilde{x} \in \mathcal{H}_k, \forall k \in \mathcal{K} \tag{4}$$

$$\alpha_{ij\ell} x_i + \beta_{ij\ell} x_j \geq \gamma_{ij\ell} \quad \forall \ell \in \{1, 2, 3\}, \forall (i, j) \in \tilde{\mathcal{M}} \tag{5}$$

$$\phi_t^\top x \geq \xi_t \quad \forall t \in \mathcal{T} \tag{6}$$

$$x^L \leq x \leq x^U \tag{7}$$

In the above formulation, constraints (4) constitute outer-approximation inequalities for  $f_k(x) \leq 0, k \in \mathcal{K}$ , where  $\tilde{x}$  are the points of linearization, noting that such outer-approximation is superfluous for any affine  $f_k(x)$ . The parameters  $\alpha_{ij\ell}, \beta_{ij\ell}$  and  $\gamma_{ij\ell}$  in

constraints (5) are suitably chosen in each case to define the linear inequalities that relax the original non-overlapping constraints discussed above, while the set  $\tilde{\mathcal{M}} \subseteq \mathcal{M}$  denotes the set of circle pairs for which the corresponding  $\theta_{ij}$  interval has been branched at least once. Constraints (6) generically denote strengthening cuts that we shall discuss in Sect. 4, while variable bounds are provided in constraints (7).

Note how, after a branch is applied, the parent node relaxation is tightened by adding elements to the set  $\tilde{\mathcal{M}}$ , when a new circle pair is branched upon for the first time, or by updating the coefficients  $\alpha/\beta/\gamma$ , when the circle pair is branched upon further. The relaxation is also tightened by appending and/or updating the set of strengthening cuts  $\mathcal{T}$ , while the outer-approximation sets  $\mathcal{H}_k$  are also expanded as appropriate (see details later).

### 3.2 The branch-and-bound algorithm

We shall now focus on the implementation of the customized BB algorithm. At the root node, the set  $\mathcal{H}_k$  is initialized with points  $wx^L + (1 - w)x^U$ , where  $w \in \{0.25, 0.5, 0.75\}$ , while the sets  $\tilde{\mathcal{M}}$  and  $\mathcal{T}$  begin as empty sets. All of them will be dynamically expanded and updated as the algorithm proceeds.

After solving the LP relaxation (3)–(7) at each node, we check whether each of the convex constraints (set  $\mathcal{K}$ ) as well as each of the non-overlapping constraints (set  $\mathcal{M}$ ) are satisfied within some predefined tolerance  $\varepsilon$  (we use  $\varepsilon = 10^{-5}$ ) at the current LP solution  $\bar{x}$ .<sup>2</sup> We first focus on the convex constraints, and if  $f_k(\bar{x}) > \varepsilon$  for some  $k \in K$ , we add an outer-approximation cut by setting  $\mathcal{H}_k \leftarrow \mathcal{H}_k \cup \{\bar{x}\}$ . Note that, at each iteration, we only add such cuts for the (up to) 5 most violated convex constraints  $k$ . If any outer-approximation cuts are added, we resolve the LP relaxation; otherwise, we proceed with checking the feasibility of the non-overlapping constraints, as follows. For each non-overlapping constraint  $x_i^2 + x_j^2 \geq \rho_{ij}^2$ ,  $(i, j) \in \mathcal{M}$ , we define  $V_{ij} := \rho_{ij} - \sqrt{\bar{x}_i^2 + \bar{x}_j^2}$ , and we consider this constraint to be violated, if  $V_{ij} > \varepsilon$ . If no violated non-overlapping constraints exist, then a feasible solution  $\bar{x}$  has been obtained; this solution is considered as a possible new incumbent, and the node is fathomed. However, if at least one non-overlapping constraint is violated, the node must be either tightened or branched, eliminating the current LP solution in either case.

Deferring the discussion of tightening techniques to Sect. 4, we now focus on our branching strategy, namely how we choose which implicit  $\theta_{ij}$  variable (element of set  $\mathcal{M} : \{V_{ij} > \varepsilon\}$ ) to branch upon. Recalling that further branching of a previously branched  $\theta_{ij}$  variable produces fewer child nodes in the BB tree than the initial branching (two versus three subdomains), the violation of currently unbranched non-overlapping constraints is first discounted by a factor of  $\tau^{unbr}$  (we use  $\tau^{unbr} = 0.7$ ); that is,  $V_{ij} \leftarrow (1 - \tau^{unbr}) V_{ij}$ , for all  $(i, j) \in \mathcal{M} \setminus \tilde{\mathcal{M}}$ . We then choose to branch on the implicit  $\theta_{ij}$  variable that corresponds to the highest violation after such modifications. Note that, according to the earlier discussion, the branching is implemented either as updating existing left-hand side coefficients or by adding new constraints, depending on whether  $(i, j) \in \tilde{\mathcal{M}}$  or not.

<sup>2</sup> For simplicity, we use here a common tolerance  $\varepsilon$  for all constraints in problem (2), but we remark that, in principle, one may use a set of constraint-specific feasibility tolerances that are appropriately scaled for each constraint.

## 4 Strengthening techniques

We consider two types of relaxation strengthening techniques, namely *intersection cuts* and *feasibility-based tightening*. Even though intersection cuts [6] are well-known for their use in integer programming, their applicability goes much beyond that area, being generic tools to deal with non-convexities that may stem from either integrality or continuous nonlinear constraints. However, to the best of our knowledge, only a few works in the literature to date have attempted to apply these techniques in the context of continuous nonlinear optimization. In this work, we propose to utilize *strengthened* intersection cuts to tighten our LP relaxations. Additionally, various feasibility-based tightening techniques are also introduced.

### 4.1 Intersection cuts

The idea of *intersection cuts* was originally proposed in [6] to construct valid cuts for integer programming. It has gradually received more and more attention in other areas such as reverse convex programming [15], bilevel optimization [13], polynomial programming [10], and factorable MINLP [28], among others. To the best of our knowledge, the literature always considers the case where variables are bounded from below by zero, while the more general case where variables are bounded from below and/or above by arbitrary values has been ignored. The challenge comes from the fact that, at the LP optimal solution, some non-basic variable might be at its upper bound, or some non-basic variable might be at its lower bound yet that bound not being 0. In such cases, the intersection cut formula from the seminal paper [6] is not immediately valid. Whereas this can be easily dealt with by explicitly adding to the model variable-bounding constraints (thus forcing all non-basic variables at an optimal LP solution to be slack variables), it is generally much more common and useful for general-purpose global optimization solvers to separate variable bounds from other constraints and to handle them separately. To this end, we present in Sect. 4.1.1 a complete derivation of intersection cuts for more general cases where variable bounds might be arbitrary numbers. Afterwards, in Sect. 4.1.3, we consider a strengthening method to further tighten these cuts.

#### 4.1.1 Generating intersection cuts

Consider the following problem:

$$\begin{aligned} & \underset{x \in \mathbb{R}^n}{\text{minimize}} && c^\top x \\ & \text{subject to} && x \in \mathcal{P} \cap \mathcal{Q}, \end{aligned} \quad (8)$$

where  $\mathcal{P} := \{x \in \mathbb{R}^n : Ax \leq b; x^L \leq x \leq x^U\}$ ,<sup>3</sup> with  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^m$ , while  $\mathcal{Q}$  represents a “complicating” set. In this work, we consider

$$\mathcal{Q} := \left\{ x \in \mathbb{R}^n : x_i^2 + x_j^2 \geq \rho_{ij}^2, \forall (i, j) \in \mathcal{M} \right\}.$$

Initially, we relax the feasible region by only considering  $x \in \mathcal{P}$ , and we introduce slack variables  $s \in \mathbb{R}_{\geq 0}^m$  to obtain a linear program in standard form.

Let  $t := (x, s)$  denote all variables, for convenience. Assume that the linear program is feasible and that its objective function is bounded; then, it can be solved via the simplex

<sup>3</sup> Structural variables  $x$  are not necessarily bounded, i.e.,  $x_i^L \rightarrow -\infty$  and/or  $x_i^U \rightarrow +\infty$ , for any and all  $i \in \{1, 2, \dots, n\}$ .



algorithm. Let  $\bar{t} = (\bar{x}, \bar{s})$  represent the current LP optimal solution. Without loss of generality, we assume that  $\bar{x} \notin \mathcal{Q}$ , since otherwise problem (8) has been solved. We assume that we are given a closed convex set  $\mathcal{S}$  whose interior contains  $\bar{x}$  but no feasible point; that is,  $\bar{x} \in \text{int}(\mathcal{S})$  and  $\text{int}(\mathcal{S}) \cap (\mathcal{P} \cap \mathcal{Q}) = \emptyset$ . In the following, we shall discuss how to generate a valid intersection cut to eliminate  $\bar{x}$  using  $\mathcal{S}$  for the case where structural variables  $x$  might be bounded from below and/or above.

Let  $\mathcal{I}$  represent the index set of structural variables  $x$ . Let  $\mathcal{B}$  and  $\mathcal{N}$  denote the index sets associated with the basic and non-basic variables, respectively. Furthermore, let  $\mathcal{N}^L$  denote the set of non-basic variables that are currently at their lower bounds and  $\mathcal{N}^U := \mathcal{N} \setminus \mathcal{N}^L$  represent the set of non-basic variables at their upper bounds. Note that both structural and slack variables might be non-basic. We focus on the structural variables and, for each non-basic such variable  $x_i, i \in \mathcal{I} \cap \mathcal{N}$ , we add a trivial relation, setting  $x_i$  to itself. The modified simplex tableau is demonstrated in (9), where non-basic variable  $t_j$  might correspond to either a structural (i.e.,  $x_k$ , for some  $k$ ) or a slack (i.e.,  $s_k$ , for some  $k$ ) variable.

$$\begin{aligned}
 x_i &= \bar{x}_i - \sum_{j \in \mathcal{N}^L} \bar{a}_{ij}(t_j - t_j^L) + \sum_{j \in \mathcal{N}^U} \bar{a}_{ij}(t_j^U - t_j) & \forall i \in \mathcal{I} \cap \mathcal{B} \\
 x_i &= \bar{x}_i - (-1)(x_i - x_i^L) & \forall i \in \mathcal{I} \cap \mathcal{N}^L \\
 x_i &= \bar{x}_i + (-1)(x_i^U - x_i) & \forall i \in \mathcal{I} \cap \mathcal{N}^U
 \end{aligned} \tag{9}$$

For convenience, tableau (9) can also be represented in vector form, where  $\bar{a}_j$  are suitably defined vectors:

$$x = \bar{x} - \sum_{j \in \mathcal{N}^L} \bar{a}_j(t_j - t_j^L) + \sum_{j \in \mathcal{N}^U} \bar{a}_j(t_j^U - t_j). \tag{10}$$

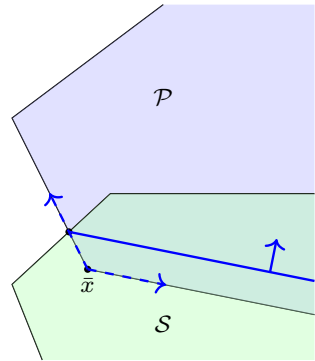
Associated with the simplex tableau (10) is a conic relaxation of  $\mathcal{P}$ , called the LP-cone, whose apex is  $\bar{x}$  and whose facets are defined by  $n$  hyperplanes that form the basis of  $\bar{x}$ . Thus, the LP-cone has  $n$  extreme rays, each formed by an extreme direction  $r^j$  emanating from the apex  $\bar{x}$ , where  $r^j = -\bar{a}_j$  if  $j \in \mathcal{N}^L$ , or  $r^j = \bar{a}_j$  if  $j \in \mathcal{N}^U$ . Following the extreme ray  $\bar{x} + \lambda_j r^j$  (where  $\lambda_j \geq 0$ ), we seek its intersecting point with the boundary of the set  $\mathcal{S}$ . More specifically, for each extreme ray  $\bar{x} + \lambda_j r^j, j \in \mathcal{N}$ , we seek to solve

$$\lambda_j^* := \underset{\lambda_j \geq 0}{\text{maximize}} \left\{ \lambda_j : \bar{x} + \lambda_j r^j \in \mathcal{S} \right\}. \tag{11}$$

Since  $\bar{x} \in \text{int}(\mathcal{S})$ , problem (11) is always feasible (e.g.,  $\lambda_j = 0$  is a feasible solution). If  $r^j$  is a recessive direction of  $\mathcal{S}$  (i.e.,  $r^j \in \text{rec}(\mathcal{S})$ , where  $\text{rec}(\mathcal{S})$  denotes the recession cone of  $\mathcal{S}$ ), there is no intersection point between the extreme ray  $\bar{x} + \lambda_j r^j$  and the boundary of the set  $\mathcal{S}$ . Thus, in this case, the objective in (11) is unbounded from above and we set  $\lambda_j^* = +\infty$ . Otherwise, we obtain a unique solution  $\lambda_j^*$  and the intersection point is  $\bar{x} + \lambda_j^* r^j$ .

Let  $\mathcal{N}^1 := \{j \in \mathcal{N} : r^j \in \text{rec}(\mathcal{S})\}$  and  $\mathcal{N}^2 := \mathcal{N} \setminus \mathcal{N}^1$  for convenience. The intersection cut that cuts off  $\bar{x}$  is defined to be the halfspace whose boundary contains each intersection point  $\bar{x} + \lambda_j^* r^j, j \in \mathcal{N}^2$  and which is parallel to each extreme direction  $r^j, j \in \mathcal{N}^1$  (Fig. 2). The validity of intersection cuts is intuitive from a geometric perspective. Thus, we skip its proof and instead focus on the derivation of the intersection cut formula.

**Fig. 2** Generation of intersection cuts



**Proposition 1** *The intersection cut can be represented in non-basic space (e.g., using non-basic variables  $t_j$ ) as follows:*

$$\sum_{j \in \mathcal{N}^L} \frac{t_j - t_j^L}{\lambda_j^*} + \sum_{j \in \mathcal{N}^U} \frac{t_j^U - t_j}{\lambda_j^*} \geq 1. \tag{12}$$

**Proof** We only need to prove that this halfspace passes through all intersection points  $\bar{x} + \lambda_j^* r^j$ ,  $j \in \mathcal{N}^2$  and that it is parallel to extreme directions  $r^j$ ,  $j \in \mathcal{N}^1$ . We consider the following two cases.

- If  $j \in \mathcal{N}^2$ , the intersection point  $\bar{x} + \lambda_j^* r^j$  is obtained via moving from  $\bar{x}$  along the direction  $r^j$  by a distance of  $\lambda_j^*$ . If  $j \in \mathcal{N}^2 \cap \mathcal{N}^L$ , in the non-basic space we have  $t_j = t_j^L + \lambda_j^* \cdot 1$ , and the other non-basic variables  $t_{j'}$  ( $j' \in \mathcal{N} \setminus \{j\}$ ) will remain at their lower or upper bounds at this intersection point. Therefore,

$$\sum_{j' \in \mathcal{N}^L} \frac{t_{j'} - t_{j'}^L}{\lambda_{j'}^*} + \sum_{j' \in \mathcal{N}^U} \frac{t_{j'}^U - t_{j'}}{\lambda_{j'}^*} = \frac{t_j^L + \lambda_j^* - t_j^L}{\lambda_j^*} = 1.$$

In other words, the intersection cut passes through the intersection point  $\bar{x} + \lambda_j^* r^j$ . This also applies in the case  $j \in \mathcal{N}^2 \cap \mathcal{N}^U$ .

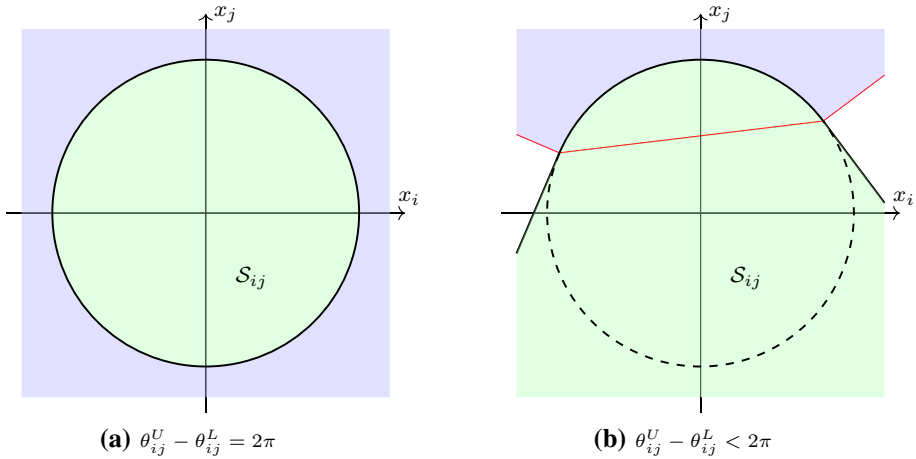
- If  $j \in \mathcal{N}^1$ , we only need to prove that the intersection cut is parallel to  $r^j$ . Moving along the direction  $r^j$  from the point  $\bar{x}$ , if  $j \in \mathcal{N}^1 \cap \mathcal{N}^L$ , we have  $t_j = t_j^L + \lambda \cdot 1$  (where  $\lambda \geq 0$ ), and the other non-basic variables  $t_{j'}$  ( $j' \in \mathcal{N} \setminus \{j\}$ ) will remain unchanged. Therefore,

$$\sum_{j' \in \mathcal{N}^L} \frac{t_{j'} - t_{j'}^L}{\lambda_{j'}^*} + \sum_{j' \in \mathcal{N}^U} \frac{t_{j'}^U - t_{j'}}{\lambda_{j'}^*} = \frac{t_j^L + \lambda - t_j^L}{\lambda_j^*} = \frac{\lambda}{+\infty} = 0.$$

The parallel relationship is proved. This also applies in the case  $j \in \mathcal{N}^1 \cap \mathcal{N}^U$ . □

As mentioned in [6],  $\lambda_j^*$  should be approximated from below for numerical validity. Readers are referred to that work for further details, with the observation however that the intersection cut formula (12) is more generic than the one presented in [6], reducing to the latter when structural variables  $x$  are only bounded from below by zero.

Note that, given a mathematical model, one has to identify a suitable set  $S$  to derive valid intersection cuts. We highlight that such a convex set  $S$  generally exists, if some constraint



**Fig. 3** Valid convex sets  $S_{ij}$  for generating intersection cuts

in the model imposes a reverse convex region. In fact, from problem (11), one can infer that deeper cuts will be produced from larger sets  $S$  whose boundary intersects with extreme rays at intersections points further away from the apex,  $\bar{x}$ .

### 4.1.2 Generating intersection cuts for non-overlapping constraints

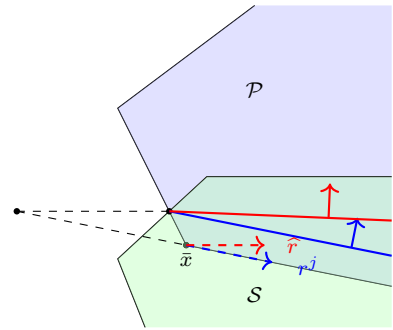
In our context, we can easily identify a valid sets  $S$  for generating intersection cuts. Considering the fact that the non-overlapping constraint  $x_i^2 + x_j^2 \geq \rho_{ij}^2$  represents a reverse convex region [15], we can define  $S_{ij} := \{x \in \mathbb{R}^n : x_i^2 + x_j^2 \leq \rho_{ij}^2\}$  (Fig. 3a) such that  $\bar{x} \in \text{int}(S_{ij})$ . Furthermore, recalling that enlarging the set whenever possible leads to stronger cuts, we can enlarge  $S_{ij}$  using the following formula (Fig. 3b) when  $\theta_{ij}^U - \theta_{ij}^L < 2\pi$ :

$$S_{ij} \leftarrow S_{ij} \cup \left\{ x \in \mathbb{R}^n \left| \begin{array}{l} \cos(\theta)x_i + \sin(\theta)x_j \leq \rho_{ij}, \quad \forall \theta \in \{\theta_{ij}^L, \theta_{ij}^U\} \\ \cos\left(\frac{\theta_{ij}^L + \theta_{ij}^U}{2}\right)x_i + \sin\left(\frac{\theta_{ij}^L + \theta_{ij}^U}{2}\right)x_j \leq \rho_{ij} \cos\left(\frac{\theta_{ij}^U - \theta_{ij}^L}{2}\right) \end{array} \right. \right\}. \tag{13}$$

The enlarged set  $S_{ij}$  remains convex and does not contain any feasible solution in its interior; therefore, intersection cuts generated from  $S_{ij}$  will be valid. We highlight that enlarging the convex set and generating stronger intersection cuts becomes possible due to the chosen scheme to branch upon non-overlapping constraints.

In our implementation, we define  $S_{ij}$  for every  $(i, j) \in \mathcal{M}$ . When  $\theta_{ij}^U - \theta_{ij}^L < 2\pi$ , we use the enlarged version (13). We use these sets to generate intersection cuts for which the corresponding non-overlapping constraint is violated, but only the best 5 cuts with Euclidean distances from  $\bar{x}$  larger than tolerance  $\delta = 10^{-2}$  are added to the model. Furthermore, whenever  $\theta_{ij}^L$  or  $\theta_{ij}^U$  is updated due to branching or tightening, we always enlarge the set  $S_{ij}$  and attempt to compute new  $\lambda_j^*$  values from (12) towards strengthening the related intersection cuts that have already been generated.

**Fig. 4** Strengthened intersection cuts



### 4.1.3 Strengthening intersection cuts

Figure 4 demonstrates a simple case where the intersection cut (blue solid line) generated from Sect. 4.1.1 can be further tightened. The generated intersection cut passes through an intersection point and is parallel to the extreme direction  $r^j$  (since  $r^j \in \text{rec}(S)$ ). In this case, we can identify another cut  $\widehat{IC}$  (red solid line), which also passes through the same intersection point, but which is parallel to some other recessive direction  $\widehat{r} \in \text{rec}(S)$ . Apparently,  $\widehat{IC}$  is a valid cut that dominates  $IC$ .

The validity of  $\widehat{IC}$  deduces from the fact that, by convexity of  $S$ , any point satisfying  $IC$  but not  $\widehat{IC}$  lies within the interior of  $S$  and is thus outside  $\mathcal{P} \cap \mathcal{Q}$ . The stronger cut  $\widehat{IC}$  can be regarded as a halfspace that passes through the existing intersection point and a new one residing at the negative part of the extreme ray  $\bar{x} + \lambda r^j$  (where  $\lambda \geq 0$ ). In order to maintain the validity of the new intersection cut, as the work of [10] suggested, one has to guarantee that  $\widehat{r} := (\bar{x} + \lambda_{j'}^* r^{j'}) - (\bar{x} + \lambda_j r^j)$ , the direction from the new intersection point,  $(\bar{x} + \lambda_j r^j)$ , to every intersection point,  $(\bar{x} + \lambda_{j'}^* r^{j'})$ , is a recessive direction of  $S$  (i.e.,  $\widehat{r} \in \text{rec}(S)$ ) for every  $j' \in \mathcal{N}^2$ . It is clear from (12) that increasing  $\lambda_j$  while  $\lambda_j \leq 0$  leads to a stronger cut. Thus, for each extreme direction  $r^j$ ,  $j \in \mathcal{N}^1$ , we define the following problem:

$$\lambda_j^* := \underset{\lambda_j \leq 0}{\text{maximize}} \left\{ \lambda_j : \lambda_{j'}^* r^{j'} - \lambda_j r^j \in \text{rec}(S), \forall j' \in \mathcal{N}^2 \right\}. \tag{14}$$

If problem (14) is infeasible, we set  $\lambda_j^* = -\infty$ ; otherwise, a unique solution  $\lambda_j^*$  is obtained, resulting in a new intersection point  $(\bar{x} + \lambda_j^* r^j)$ . With this, we arrive at Proposition 2.

**Proposition 2** *The intersection cut defined by (12) and using  $\lambda_j^*$  from (11), for  $j \in \mathcal{N}^2$ , and  $\lambda_j^*$  from (14), for  $j \in \mathcal{N}^1$ , is valid.*

Given a convex set  $S$ , the problem (14) is in general hard to solve because no closed-form formula for  $\text{rec}(S)$  is available. However, in our context, the recession cone  $\text{rec}(S_{ij})$  can be easily identified. More specifically, we distinguish three cases:

- (i) when  $\theta_{ij}^U - \theta_{ij}^L > \pi$ , the projection of  $S_{ij}$  onto the  $x_i - x_j$  space is a bounded area, and hence its recession cone is

$$\text{rec}(S_{ij}) = \{x \in \mathbb{R}^n : x_i = 0, x_j = 0\},$$

projecting to a singleton on the  $x_i - x_j$  space;

(ii) when  $\theta_{ij}^U - \theta_{ij}^L = \pi$ , the recession cone becomes

$$rec(\mathcal{S}_{ij}) = \left\{ x \in \mathbb{R}^n : x_i = -\lambda \sin(\theta_{ij}^L), x_j = \lambda \cos(\theta_{ij}^L), \forall \lambda \geq 0 \right\},$$

projecting to a ray on the  $x_i-x_j$  space;

(iii) when  $\theta_{ij}^U - \theta_{ij}^L < \pi$ , it is easy to show that

$$\begin{aligned} rec(\mathcal{S}_{ij}) &= rec \left( \left\{ x \in \mathbb{R}^n : \cos(\theta)x_i + \sin(\theta)x_j \leq \rho_{ij}, \forall \theta \in \{\theta_{ij}^L, \theta_{ij}^U\} \right\} \right) \\ &= \left\{ x \in \mathbb{R}^n : \cos(\theta)x_i + \sin(\theta)x_j \leq 0, \forall \theta \in \left\{ \theta_{ij}^L, \theta_{ij}^U \right\} \right\}. \end{aligned} \tag{15}$$

We remark that, when case (i) applies, the problem (14) is almost always infeasible, due to insufficient degrees of freedom. Furthermore, case (ii) is unlikely to be relevant in the context of double-precision arithmetic, due to inability to detect that  $\theta_{ij}^U - \theta_{ij}^L$  equals  $\pi$  exactly, and hence, we did not consider this case in our implementation. It is only under case (iii) that the closed-form formula (15) can be plugged into (14) and a value for  $\lambda_j^*$  be analytically identified for every  $j \in \mathcal{N}^1$ . Therefore, in our implementation, we generate intersection cuts using Proposition 1 in cases (i) and (ii), and we attempt to generate a strengthened version of such cuts using Proposition 2 only in case (iii).

### 4.2 Feasibility-based tightening

Feasibility-based tightening is a common technique in global optimization to reduce variable bounds, and it is usually implemented via interval arithmetic [25]. In our context, since explicit structural variable domains are not branched upon, the interval arithmetic technique would not tighten variable bounds effectively. Considering that our strategy is to branch on the feasible intervals of implicit variables  $\theta_{ij}$ , our feasibility-based tightening approach will instead focus on reducing the domains for these variables.

From a geometric perspective, the non-overlapping constraint,  $x_i^2 + x_j^2 \geq \rho_{ij}^2$ , requires the solution’s projection on the  $x_i-x_j$  space to lie either outside or exactly on the circumference of disk  $D_{ij}$  (Fig. 1). Let  $FR_{ij}$  denote the feasible region of the  $x_i-x_j$  space. At the root node of the BB tree, we have  $FR_{ij} = conv \left( \left\{ (x_i, x_j) : x_i^L \leq x_i \leq x_i^U, x_j^L \leq x_j \leq x_j^U \right\} \setminus int(D_{ij}) \right)$ , and this region gradually reduces due to branching or tightening. We emphasize that the feasible spaces  $FR_{ij}$ ,  $(i, j) \in \mathcal{M}$  are not explicitly enforced in the relaxation model, (3)–(7), because they are generally implied by other constraints. We also note that  $FR_{ij}$  is always a bounded polyhedron (assuming that valid lower and upper bounds for  $x_i, i \in \{1, 2, \dots, n\}$ , can be extracted or deduced), and that this region might correspond to a reduced interval  $[\theta_{ij}^L, \theta_{ij}^U]$  for the implied variable  $\theta_{ij}$ . To that end, in our implementation, we attach our description of  $FR_{ij}$  to the BB node as important information to be used for tightening the  $\theta_{ij}$  intervals, as follows.

Assume that the projection of the current LP solution  $\bar{x}$  lies in the interior of  $D_{ij}$ , i.e.,  $X \in FR_{ij} \cap int(D_{ij})$ . Whenever  $FR_{ij}$  is reduced due to some tightening technique, we can revisit the geometric meaning of the non-overlapping constraint and obtain  $FR_{ij} \leftarrow conv (FR_{ij} \setminus int(D_{ij}))$  (Fig. 5). From that, we can infer possibly tighter  $[\theta_{ij}^L, \theta_{ij}^U]$  bounds, and whenever the latter are indeed tightened, we first update the relaxations for its corresponding non-overlapping constraint and then check whether  $X$  is cut off by the halfspace formed by the new secant line. Next, we discuss three opportunities to reducing  $FR_{ij}$ .

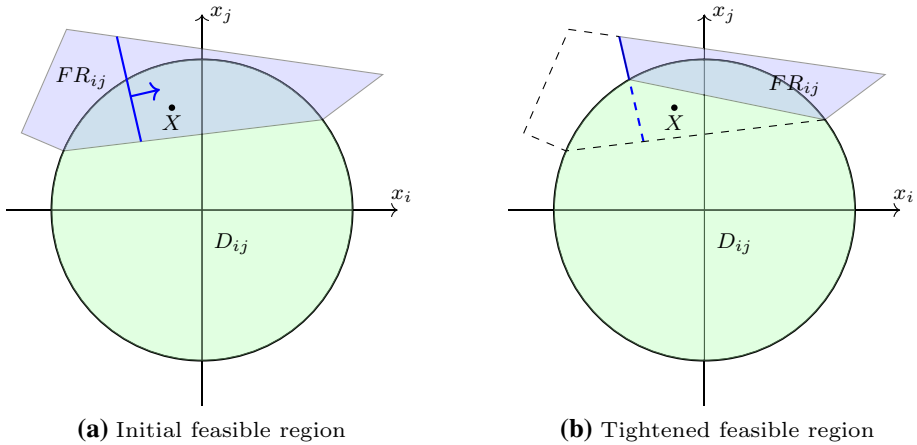


Fig. 5 Feasibility-based tightening

### 4.2.1 Generating concave envelopes

Let us begin by defining  $h(x_i, x_j) := x_i^2 + x_j^2$ , where  $(x_i, x_j) \in FR_{ij}$ . We seek the *concave envelope*, i.e., the tightest possible over-estimator, of  $h(x_i, x_j)$  over its domain  $FR_{ij}$ , which we shall denote as  $conc_{FR_{ij}}h(x_i, x_j)$ . It is clear that the function  $h(x_i, x_j)$  has a *vertex polyhedral* concave envelope over the bounded polyhedral domain  $FR_{ij}$  [31]; therefore, its concave envelope coincides with the concave envelope of its restriction to the vertices of  $FR_{ij}$  and consists of finitely many hyperplanes.<sup>4</sup>

As a result, we obtain the following constraints that are valid for our relaxation:

$$conc_{FR_{ij}}h(x_i, x_j) \geq \rho_{ij}^2. \tag{16}$$

We remark that constraints (16) are not directly added to the relaxation model, (3)–(7), rather used to reduce  $FR_{ij}$ . More specifically,  $FR_{ij} \leftarrow FR_{ij} \cap \left\{ (x_i, x_j) : conc_{FR_{ij}}h(x_i, x_j) \geq \rho_{ij}^2 \right\}$ . If  $FR_{ij}$  is indeed reduced in this manner, we then attempt to apply feasibility-based tightening as described in the preamble of this section to further tighten  $FR_{ij}$ . Note that this process can be applied recursively, since a smaller polytope  $FR_{ij}$  will induce a tighter over-estimator for  $h(x_i, x_j)$ , which might in turn further reduce  $FR_{ij}$ .

### 4.2.2 Calculating Minkowski sums

Another typical feasibility-based tightening technique in global optimization is *bound propagation* [25]. Adapting this idea into our context, we propose to apply *domain propagation*. For example, in the context of packing three circles  $i, j$  and  $k$ , one usually seeks to enforce non-overlapping constraints in a pairwise sense. In this case, any feasible point  $(a_i - a_j, b_i - b_j)$

<sup>4</sup> From our computational experience, the number of vertices of  $FR_{ij}$  is usually small (around 10), and thus identifying its concave envelope via enumeration is computationally efficient. Regardless, if the number of vertices of  $FR_{ij}$  is large, one can always properly relax the domain  $FR_{ij}$  and obtain a new bounded polyhedron that contains  $FR_{ij}$  and has a small number of vertices.

satisfying the non-overlapping constraint  $(a_i - a_j)^2 + (b_i - b_j)^2 \geq (r_i + r_j)^2$  must correspond to a point  $(a_i - a_k, b_i - b_k)$  in the domain  $FR_{ik}$  as well as to a point  $(a_k - a_j, b_k - b_j)$  in the domain  $FR_{kj}$ . This results from the simple observation that

$$(a_i - a_j, b_i - b_j) = (a_i - a_k, b_i - b_k) + (a_k - a_j, b_k - b_j). \quad (17)$$

Consequently, one can calculate the Minkowski sum of  $FR_{ik}$  and  $FR_{kj}$  to tighten  $FR_{ij}$ ; that is,  $FR_{ij} \leftarrow FR_{ij} \cap (FR_{ik} \oplus FR_{kj})$ . If  $FR_{ij}$  is indeed reduced, the feasibility-based tightening  $FR_{ij} \leftarrow \text{conv}(FR_{ij} \setminus \text{int}(D_{ij}))$  will be considered. Note that the Minkowski sum of two polytopes with  $n_1$  and  $n_2$  vertices, respectively, can be computed in  $O(n_1 + n_2)$  time [9]; thus, the proposed procedure is computationally efficient.

### 4.2.3 Solving LPs

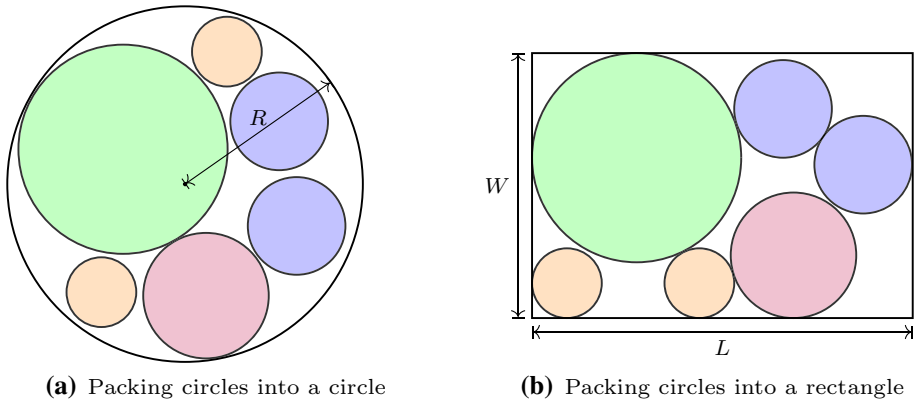
We can also refine feasible region  $FR_{ij}$  by using the projection of the feasible region of the relaxation model, (3)–(7), onto the  $x_i$ – $x_j$  space. Considering that computing the exact projection is not practical, we choose to outer-approximate it via linear inequalities defined in the  $x_i$ – $x_j$  space. More specifically, one can solve an LP with the objective of minimizing some linear function, e.g.,  $\mu x_i + \nu x_j$ , and with constraints (4)–(7), where  $\mu$  and  $\nu$  are properly chosen coefficients. Let  $\sigma$  be its optimal value, thus  $\mu x_i + \nu x_j \geq \sigma$  is a valid inequality to characterize the projection area and can be used to refine  $FR_{ij}$ ; that is,  $FR_{ij} \leftarrow FR_{ij} \cap \{(x_i, x_j) : \mu x_i + \nu x_j \geq \sigma\}$ . If  $FR_{ij}$  is successfully reduced in this manner, we then apply feasibility-based tightening as before.

In our context, we apply the above process for two separate objective functions, namely those that are parallel to the two linear boundaries of  $FR_{ij}$  that neighbor the secant line. These were specifically chosen due to their potential to immediately tighten the secant line, which might thus help to eliminate the current LP solution,  $\bar{x}$ .

## 4.3 Implementation details

We note the following details about our implementation.

- With regards to the protocol via which we employ the various tightening techniques, we note the following. We first apply intersection cuts, followed by calculating Minkowski sums, and finally by solving LPs. Tightening from concave envelopes is swiftly applied as soon as  $FR_{ij}$  is tightened during the latter two steps, and whenever  $\bar{x}$  is cut off, we skip the subsequent routines and turn our attention to resolving the LP relaxation (3)–(7).
- Since the intersection cuts are formulated in non-basic space, and since we do not have access to non-basic slack variables when enforcing a constraint in the LP solver, we first convert it to the structural space. After this step, we apply some presolve reductions to mitigate numerical difficulties [3]. In particular, we first scale the coefficients in a cut such that their largest absolute value is 1.0, and we then perform proper reductions on small coefficients (e.g., less than  $10^{-4}$ ) while maintaining the cut's validity.
- In order to alleviate tapering off from using intersection cuts, we stop the separation if the gap is decreased by less than 0.01% a total of 3 times.
- Outer-approximation and intersection cuts are removed from the LP model (3)–(7), if they are not active for the past 50 LP-solving rounds. We note that removed intersection cuts are still stored in the cut pool and will be strengthened when the corresponding convex



**Fig. 6** Two variants of circle packing problems

sets are enlarged, in the hope that they might be added back to the set  $\mathcal{T}$  if violated at some future point.

- Whenever  $FR_{ij}$  is reduced at some BB node, we recursively call this routine to further reduce the feasible region, and this continues until the reduction in area of  $FR_{ij}$  is less than 1%. Moreover, whenever the area of  $FR_{ij}$  is reduced by at least 5%, we attempt to apply domain propagation to tighten relevant domains. However, since solving LPs can add up to the overall computational time, we only consider the 10 pairs  $(i, j) \in \mathcal{M}$  with the largest violation values,  $V_{ij}$ .

## 5 Computational studies

In this section, we test our customized branch-and-bound approach and compare its performance against the state-of-the-art general purpose global optimization solvers, namely ANTIGONE 1.1 [24], BARON 19.7.13 [32], COUENNE 0.5 [8], LINDOGLOBAL 12.0 and SCIP 6.0 [2,33]. Our algorithm was implemented in C++ and the LP relaxation models were solved via CPLEX Optimization Studio 12.8.0 through the C application programming interface. Global solvers were called within the GAMS 28.2.0 environment. The absolute optimality tolerance was set to be  $10^{-4}$ . All computational experiments were conducted on a single thread of an Intel Xeon CPU E5-2689 v4 @ 3.10GHz with 32GB of RAM.

### 5.1 Circle packing

#### 5.1.1 Problem definitions

Let  $\mathcal{C} = \{1, 2, 3, \dots\}$  denote a set of circles. For each circle  $i \in \mathcal{C}$ , let  $r_i$  denote its radius. Without loss of generality, we assume a circle ordering such that  $r_i \geq r_j$ , when  $i < j$ , for all  $i, j \in \mathcal{C}$ . We aim to identify a feasible configuration of these circles within (i) a larger circle, and (ii) a sheet of fixed width  $W$ , such that no circles overlap and the size of the circle (radius  $R$ ) or the sheet (length  $L$ ) is minimized. We refer to these settings as *packing circles into a circle* and *packing circles into a rectangle*, respectively. The latter is often also referred to in the literature as the circular open dimension problem (CODP).



### 5.1.2 Mathematical modeling

Let  $(a_i, b_i)$  denote the final coordinates for the center of circle  $i \in C$ . Assuming that the center of the containing circle is fixed to the point  $(0, 0)$ , the problem of packing circles into a circle can be formulated as the non-convex model (18)–(21).

$$\underset{a_i, b_i, R}{\text{minimize}} \quad R \tag{18}$$

$$\text{subject to} \quad a_i^2 + b_i^2 \leq (R - r_i)^2 \quad \forall i \in C \tag{19}$$

$$(a_i - a_j)^2 + (b_i - b_j)^2 \geq (r_i + r_j)^2 \quad \forall (i, j) \in C \times C : i < j \tag{20}$$

$$R \geq \max_{i \in C} \{r_i\} \tag{21}$$

Constraints (19) enforce that all circles lie within the circular container, constraints (20) guarantee that there is no overlap among the circles, while constraint (21) specifies the applicable lower bound on the containing circle’s radius. Additionally, as also pointed out in [27], two types of symmetry-breaking constraints can be added to the above model:

- i. Identical circles: When two circles  $i$  and  $j$ ,  $i < j$ , are identical (i.e.,  $r_i = r_j$ ), then  $a_i \leq a_j$  breaks this symmetry. In our implementation, we equivalently enforce this by tightening  $[\theta_{ij}^L, \theta_{ij}^U] = [\pi/2, 3\pi/2]$  at the root node.
- ii. Rotational symmetry and reflection: Rotational symmetry arises due to the fact that an equivalent packing solution can always be obtained via rotating the entire configuration by any angle; hence, one can enforce that  $a_1 \leq 0$  and  $b_1 = 0$ . Furthermore, the reflection of a feasible configuration through the horizontal axis results in an equivalent solution; hence, one can further impose  $b_2 \geq 0$ .

The mathematical formulation for packing circles into a rectangle is similar. Assuming that the bottom left corner of the rectangle is fixed to the point  $(0, 0)$ , the non-convex model (22)–(25) applies.

$$\underset{a_i, b_i, L}{\text{minimize}} \quad L \tag{22}$$

$$\text{subject to} \quad r_i \leq a_i \leq L - r_i \quad \forall i \in C \tag{23}$$

$$r_i \leq b_i \leq W - r_i \quad \forall i \in C \tag{24}$$

$$(a_i - a_j)^2 + (b_i - b_j)^2 \geq (r_i + r_j)^2 \quad \forall (i, j) \in C \times C : i < j \tag{25}$$

In this case, applicable constraints to break symmetry from reflection are: (i)  $a_1 \leq L/2$ , and (ii)  $b_1 \leq W/2$ .

### 5.1.3 Benchmark instances

In order to evaluate the performance of our approach, we consider instances of packing circles into a circle from [1]. Each instance is defined by two numbers,  $p$  and  $N$ , and the size of circles to be packed in every instance is defined by  $r_i = i^p$ , where  $i = 1, 2, \dots, N$ . We consider  $N \in \{5, 6, \dots, 11\}$  and  $p \in \{-2/3, -1/2, -1/5, +1/2, +1\}$ ; thus, in total we have 35 instances of difference sizes.

For packing circles into a rectangle, we generate a suite of instances using data from the thirty-circle example in [29]. We consider  $N \in \{6, 7, \dots, 10\}$ , and using the first  $N$  circles in each case, we generate all possible packing instances with  $N - 1$  circles. Therefore, we

generate a total of 40 instances, namely 6 five-circle, 7 six-circle, 8 seven-circle, 9 eight-circle and 10 nine-circle instances. The rectangle width  $W$  is chosen to be 9.5, as per the literature reference [29].

## 5.2 Computational results

The goals of our computational study are to (i) assess the effect of strengthened intersection cuts and feasibility-based tightening on the BB tree, and (ii) conduct a comprehensive comparison between our proposed algorithm and state-of-the-art global solvers. These two goals are covered in Sects. 5.2.1 and 5.2.2, respectively. To ensure a fair comparison, all algorithms (including the global solvers) were initialized with heuristic solutions, which were obtained after running BARON heuristics with a time limit of 1 hour.

### 5.2.1 Effect of node relaxation tightening

We first analyze the effect of using strengthened intersection cuts on the branch-and-bound algorithm. We then enable our feasibility-based tightening techniques and assess the additional tractability gains from doing so. Overall, we consider three versions of our algorithm: (i) the rudimentary branch-and-bound algorithm (denoted by “BB”); (ii) “BB” enhanced with strengthened intersection cuts (denoted by “BB+SIC”); (iii) “BB+SIC” enhanced with feasibility-based tightening (denoted by “BB+SIC+FBT”). For each version, we adopt the best-bound first search node selection strategy during the branch-and-bound process.

Tables 1 and 2 present the computational results for packing circles into a circle and a rectangle, respectively. The first two columns in these tables list the input size (number of circles) and the number of instances of a given input size, for a respective total of 35 and 40 instances, as described above. The tables also present the number of instances that were solved to provable optimality within a given time limit of 1 hour, as well as the geometric means of solution time and number of branch-and-bound nodes explored; for the remaining instances for which optimality could not be proven, we present the average residual gap, defined as  $(UB - LB)/UB$ , at the time limit.

Table 1 shows that BB could solve 20 out of 35 instances optimally, while both BB+SIC and BB+SIC+FBT solved 23 and 26 of them to optimality, respectively, including one of the largest instances featuring the packing of eleven circles. Putting aside the fact that it allowed us to solve three more instances within the allotted time limit, the addition of strengthened intersection cuts resulted in a noticeable improvement in average solution times, number of nodes explored and residual gaps (when applicable). The BB+SIC+FBT version performed even better in terms of these metrics, especially for instances of larger input size, such as instances with ten or eleven circles.

Turning our attention to Table 2, we observe that all three variants can easily address instances featuring up to seven circles. The BB+SIC improved upon the baseline approach in terms of being able to prove optimality within the allotted time limit for a number of the eight-circle instances, with the BB+SIC+FBT approach solving the majority of them while also demonstrating a noticeable reduction in the average solution time to do so. Instances with nine circles remained elusive to all methods, with a significant average residual gap above 15%; for this problem size category, enabling feasibility-based tightening did not seem to improve the results already obtained via the use of only intersection cuts.

Considering all instances across both datasets, we conclude that the utilization of both strengthened intersection cuts and feasibility-based tightening has an overall positive impact

**Table 1** Effect of strengthened intersection cuts and feasibility-based tightening on solving instances of packing circles into a circle

# circles	# inst.	BB			BB + SIC			BB + SIC + FBT					
		# solved	Time (sec)	# nodes	Gap (%)	# solved	Time (sec)	# nodes	Gap (%)	# solved	Time (sec)	# nodes	Gap (%)
5	5	5	0.3	903	–	5	0.3	408	–	5	0.2	17	–
6	5	5	3.6	9388	–	5	2.3	2657	–	5	0.6	56	–
7	5	5	23	45,175	–	5	13	11,077	–	5	2.0	213	–
8	5	3	149	183,638	5.4	4	68	52,731	5.2	4	17	2622	3.1
9	5	1	122	202,293	6.7	2	249	160,174	5.3	4	223	22,536	12.1
10	5	1	222	323,408	10.0	1	194	118,791	7.7	2	120	11,901	9.2
11	5	0	–	–	12.0	1	2,489	762,566	10.6	1	967	158,403	9.9

**Table 2** Effect of strengthened intersection cuts and feasibility-based tightening on solving instances of packing circles into a rectangle

# circles	# inst.	BB				BB + SIC				BB + SIC + FBT			
		# solved	Time (sec)	# nodes	Gap (%)	# solved	Time (sec)	# nodes	Gap (%)	# solved	Time (sec)	# nodes	Gap (%)
5	6	6	0.1	143	-	6	0.1	95	-	6	0.1	28	-
6	7	7	0.2	874	-	7	0.2	252	-	7	0.2	69	-
7	8	8	2.6	21,049	-	8	3.5	3636	-	8	2.7	693	-
8	9	0	-	-	10.8	3	3052	770,170	5.1	7	1264	126,822	9.6
9	10	0	-	-	23.6	0	-	-	15.9	0	-	-	16.5

on performance of the algorithm. Therefore, we adopt approach BB+SIC+FBT during the comparative study that follows.

## 5.2.2 Comparison with global optimization solvers

In order to evaluate the competitiveness of our proposed approach against general-purpose global optimization solvers, we solve every benchmark instance independently with each of five state-of-the-art solvers, namely ANTIGONE, BARON, COUENNE, LINDOGLOBAL and SCIP, imposing the same time limit of 1 hour. The consolidated results are presented in Tables 3 and 4, using similar format as before, while the detailed results are presented in Tables 5 and 6.

Considering all five global solvers, BARON performs the best in terms of number of solved instances and solution time. The smallest average residual gaps for unsolved instances were obtained by SCIP, though possibly this results from aggressive branching, rather than from tight relaxations constructed at its branch-and-bound nodes. This is corroborated by the observation that the number of branch-and-bound nodes in SCIP runs was significantly larger than in other solvers. Processing a large branch-and-bound tree takes more time, which might explain why SCIP could only solve fewer instances to guaranteed optimality within the time limit, performing generally worse than the other four global solvers.

More specifically, for the problem of packing circles in a circle, BARON was able to prove optimality in 21 out of 35 instances, with the largest solvable input size being a single nine-circle instance. In contrast, our proposed algorithm (Table 1) could in addition solve a handful of nine-circle, ten-circle and eleven-circle instances, pushing the state-of-the-art in terms of what is considered solvable for this class of packing problems. Furthermore, the solution time of our approach is generally much less than that of the global optimization solvers. Turning our attention to the application of packing circles into a rectangle, BARON again performs the best among the five general-purpose global solvers, being able to solve to confirmed optimality 29 out of 40 instances. While our algorithm (Table 2) solved 28 instances with comparable solution time, it achieved a much smaller average residual gaps for the remaining instances. Notably, across both problem datasets, our proposed approach was able to close the gap in a total of 6 instances for which none of the global optimization solvers was able to prove optimality. The competitiveness of our approach against these solvers can also be inferred from rigorous performance profiles [12], which we present in Fig. 7.

## 6 Conclusions

In this work, we focused on a class of reverse convex constraints called circle-circle non-overlapping constraints, which are popular in many cutting and packing optimization models. Adapting a custom-built branch-and-bound algorithm that we had previously developed to address irregular shape nesting problems [34], we proposed strengthened intersection cuts and various feasibility-based tightening techniques to expedite the search based on direct branching upon the set of non-overlapping constraints. To this end, we first generalized the intersection cut formula from the seminal paper of [6] to more generic cases where variables can be bounded by arbitrary values, and used the fact that a non-overlapping constraint represents a reverse convex domain to incorporate such intersection cuts into our relaxations. In addition, we proposed feasibility-based tightening techniques that differ from the ones

**Table 3** Computational results for global solvers on solving instances of packing circles into a circle

# circles	# inst.	ANTIGONE			BARON			COUENNE			LINDOGLOBAL			SCIP		
		# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)
5	5	5	0.1	–	5	0.2	–	5	0.4	–	5	1.4	–	5	6.5	–
6	5	5	5.1	–	5	1.3	–	5	5.3	–	5	5.5	–	4	34	4.2
7	5	2	0.2	3.1	5	13	–	5	145	–	5	36	–	1	7.0	3.4
8	5	1	54	11.4	4	202	24.0	2	587	11.1	3	836	38.1	1	270	7.9
9	5	0	–	13.1	2	1387	26.6	0	–	12.8	1	2394	31.5	0	–	9.5
10	5	0	–	15.4	0	–	18.9	0	–	15.8	0	–	30.1	0	–	12.3
11	5	0	–	19.6	0	–	21.3	0	–	18.5	0	–	34.4	0	–	13.5

**Table 4** Computational results for global solvers on solving instances of packing circles into a rectangle

# circles	ANTIGONE			BARON			COUENNE			LINDOGLOBAL			SCIP		
	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)
5	6	2.6	-	6	0.2	-	6	11	-	6	9.0	-	6	14	-
6	7	13	-	7	0.5	-	6	83	11.7	7	114	-	7	256	-
7	8	114	-	8	6.7	-	7	866	26.7	5	1415	5.8	5	633	2.1
8	9	0	7.2	8	1033	35.5	0	-	22.0	0	-	33.6	0	-	9.0
9	10	0	22.3	0	-	44.2	0	-	32.4	0	-	46.4	0	-	18.8

**Table 5** Detailed results for packing circles into a circle

<i>N</i>	<i>p</i>	ANTIGONE	BARON	COUENNE	LINDOGLOBAL	SCIP	This work
5	-2/3	0.1	0.1	0.1	0.8	0.7	0.1
5	-1/2	0.1	0.3	0.8	1.5	4.0	0.2
5	-1/5	7.0	0.9	3.0	3.7	54	0.9
5	+1/2	1.0	0.2	0.7	1.3	32	0.1
5	+1	0.1	0.1	0.1	0.9	2.3	0.1
6	-2/3	0.1	0.1	0.2	0.9	1.0	0.1
6	-1/2	24	4.1	19	8.9	103	2.8
6	-1/5	518	31	106	47	[4.2%]	13
6	+1/2	28	3.8	24	13	2,905	1.4
6	+1	1.0	0.1	0.4	1.0	4.4	0.1
7	-2/3	0.1	0.1	1.2	1.0	7.0	0.1
7	-1/2	[0.7%]	63	311	86	[5.4%]	18
7	-1/5	[6.5%]	312	1482	367	[2.4%]	101
7	+1/2	[2.0%]	30	614	120	[4.3%]	2.6
7	+1	4.0	6.8	187	15	[1.4%]	0.1
8	-2/3	54	32	96	148	270	18
8	-1/2	[8.0%]	694	[5.8%]	1597	[4.5%]	138
8	-1/5	[21.4%]	[24.0%]	[16.4%]	[23.6%]	[9.5%]	[3.1%]
8	+1/2	[10.6%]	695	[11.1%]	2,469	[13.6%]	31
8	+1	[5.6%]	106	3,598	[52.6%]	[4.2%]	1.2



Table 5 continued

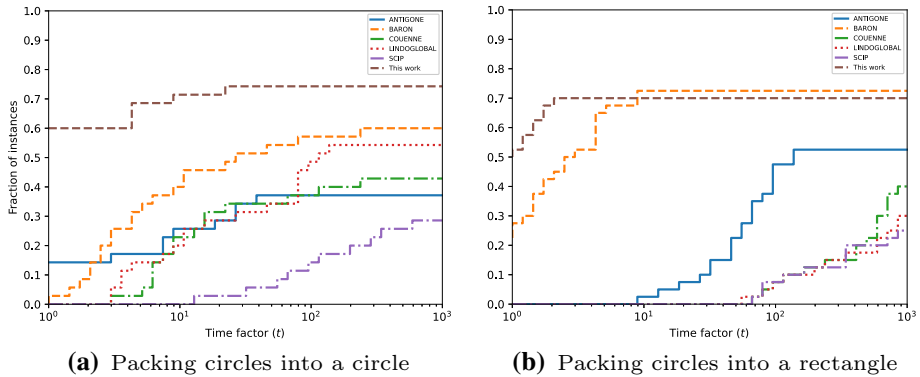
<i>N</i>	<i>p</i>	ANTIGONE	BARON	COUENNE	LINDOGLOBAL	SCIP	This work
9	-2/3	[0.5%]	621	[0.4%]	2,394	[0.4%]	143
9	-1/2	[9.2%]	[10.0%]	[10.1%]	[10.1%]	[6.6%]	2099
9	-1/5	[24.1%]	[45.3%]	[23.9%]	[36.9%]	[16.7%]	[12.1%]
9	+1/2	[20.0%]	[24.4%]	[17.6%]	[24.6%]	[13.1%]	724
9	+1	[11.5%]	3099	[11.9%]	[54.5%]	[10.5%]	11
10	-2/3	[1.0%]	[1.3%]	[1.0%]	[1.3%]	[1.1%]	1,307
10	-1/2	[11.3%]	[13.6%]	[12.2%]	[13.6%]	[10.6%]	[5.3%]
10	-1/5	[28.5%]	[32.4%]	[28.2%]	[46.0%]	[22.5%]	[19.8%]
10	+1/2	[22.1%]	[32.7%]	[22.1%]	[32.3%]	[16.2%]	[2.6%]
10	+1	[14.1%]	[14.4%]	[15.3%]	[57.2%]	[11.0%]	11
11	-2/3	[1.5%]	[1.8%]	[1.5%]	[1.8%]	[2.6%]	[1.1%]
11	-1/2	[12.2%]	[12.9%]	[13.2%]	[14.6%]	[11.3%]	[8.2%]
11	-1/5	[32.3%]	[37.4%]	[31.6%]	[53.0%]	[26.3%]	[22.9%]
11	+1/2	[34.4%]	[34.5%]	[26.1%]	[45.0%]	[21.1%]	[7.5%]
11	+1	[17.8%]	[19.9%]	[20.1%]	[57.7%]	[6.3%]	967
# solved (/35)		13	21	17	19	11	26
Time (sec)		0.8	7.4	12	19	17	4.9
[Gap]		[13.4%]	[21.8%]	[14.9%]	[32.8%]	[9.4%]	[9.2%]

**Table 6** Detailed results for packing circles into a rectangle

<i>N</i>	Omitted circle #	ANTIGONE	BARON	COUENNE	LINDOGLOBAL	SCIP	This work
5	1	2.0	0.1	6.2	7.6	5.6	0.1
5	2	3.0	0.2	109	9.1	19	0.2
5	3	1.0	0.1	5.8	7.7	35	0.1
5	4	3.0	0.1	7.0	7.8	5.7	0.1
5	5	2.0	0.2	5.0	7.6	21	0.1
5	6	9.0	0.2	13	17	14	0.2
6	1	10	0.3	74	92	43	0.1
6	2	25	0.8	[11.7%]	101	89	0.5
6	3	10	0.7	73	94	602	0.1
6	4	13	0.4	109	141	890	0.2
6	5	10	0.3	66	94	291	0.1
6	6	21	0.7	85	132	427	0.1
6	7	11	0.5	102	165	283	0.1
7	1	51	3.4	669	1275	[1.3%]	0.8
7	2	1677	178	[26.7%]	[12.8%]	[4.0%]	411
7	3	57	1.6	628	1335	1567	0.7
7	4	70	3.7	975	[0.1%]	1877	2.0
7	5	57	2.8	620	1365	285	0.8
7	6	75	3.2	1039	1587	281	0.7
7	7	251	22	1644	[4.7%]	[1.1%]	17
7	8	76	5.9	838	1,537	431	0.6

Table 6 continued

N	Omitted circle #	ANTIGONE	BARON	COUENNE	LINDOGLOBAL	SCIP	This work
8	1	[3.8%]	473	[16.7%]	[30.8%]	[9.6%]	683
8	2	[20.6%]	[35.5%]	[43.1%]	[39.2%]	[16.3%]	[15.4%]
8	3	[0.4%]	555	[17.0%]	[30.9%]	[6.7%]	710
8	4	[5.5%]	863	[19.4%]	[32.1%]	[7.3%]	1492
8	5	[0.9%]	553	[16.8%]	[30.9%]	[7.5%]	592
8	6	[7.1%]	1451	[20.2%]	[32.0%]	[6.2%]	1832
8	7	[16.5%]	3550	[23.4%]	[36.9%]	[13.0%]	[3.9%]
8	8	[4.3%]	1301	[21.5%]	[31.8%]	[6.7%]	2472
8	9	[6.1%]	1545	[19.8%]	[37.7%]	[8.0%]	2655
9	1	[17.9%]	[42.9%]	[30.2%]	[45.6%]	[15.5%]	[13.7%]
9	2	[28.6%]	[45.0%]	[44.9%]	[48.2%]	[22.3%]	[24.3%]
9	3	[23.3%]	[42.1%]	[30.4%]	[45.6%]	[16.4%]	[14.5%]
9	4	[19.0%]	[42.7%]	[31.7%]	[45.5%]	[17.6%]	[15.3%]
9	5	[20.8%]	[42.1%]	[30.1%]	[45.7%]	[18.4%]	[13.5%]
9	6	[23.3%]	[44.8%]	[31.1%]	[44.2%]	[21.3%]	[16.5%]
9	7	[24.3%]	[46.0%]	[32.4%]	[47.8%]	[20.7%]	[18.2%]
9	8	[24.2%]	[46.0%]	[31.1%]	[44.4%]	[18.9%]	[17.5%]
9	9	[21.3%]	[46.7%]	[30.4%]	[50.1%]	[17.2%]	[16.4%]
9	10	[20.6%]	[43.3%]	[31.6%]	[47.2%]	[19.1%]	[15.1%]
# solved (/40)		21	29	19	18	18	28
Time (sec)		19	6.6	104	99	124	2.8
[Gap]		[15.2%]	[43.4%]	[26.7%]	[35.6%]	[12.5%]	[15.4%]



**Fig. 7** Performance profiles across all benchmark instances of each problem variant. In both graphs, “This work” refers to the performance of our proposed algorithm using strengthened intersection cuts and feasibility-based tightening (BB+SIC+FBT). For each curve, the value at  $t = 0$  provides the fraction of benchmark instances for which the corresponding solver/algorithm is fastest, while the limiting value at  $t \rightarrow \infty$  provides the fraction of instances that could be solved within the time limit of 1 hour

commonly used in global optimization, in the sense that we sought to reduce the feasible domain enforced by non-overlapping constraints directly, rather than rely on domain reduction from variable bounds. Our extensive computational studies on 75 instances for unequal circle packing elucidated the effectiveness of tightening the relaxations in terms of speeding up the branch-and-bound process, and showcased that the purposed-build search approach performs favorably as compared to using state-of-the-art, yet general-purpose global optimization solvers.

Despite these advances, however, there remains a considerable gap between our ability to globally optimize instances that solely feature identical circles and those involving non-identical circles as well. Indeed, how to adapt many of the techniques utilized in specialized solvers for the former setting, such as employing optimality-based cuts to tighten the relaxations, in the case of unequal circle packing and general shape nesting constitute interesting avenues for future research.

**Acknowledgements** Akang Wang gratefully acknowledges financial support from the James C. Meade Graduate Fellowship and the H. William and Ruth Hamilton Prengle Graduate Fellowship at Carnegie Mellon University.

## References

1. Specht, E.: <http://www.packomania.com/>. Accessed 23 Aug 2020
2. Achterberg, T.: Scip: solving constraint integer programs. *Math Program Comput* **1**(1), 1–41 (2009)
3. Achterberg, T., Bixby, R.E., Gu, Z., Rothberg, E., Weninger, D.: Presolve reductions in mixed integer programming. *INFORMS J Comput* **32**(2), 473–506 (2019)
4. Alvarez-Valdés, R., Parreño, F., Tamarit, J.M.: Reactive grasp for the strip-packing problem. *Computer Op Res* **35**(4), 1065–1083 (2008)
5. Anstreicher, K.M.: On convex relaxations for quadratically constrained quadratic programming. *Math Program* **136**(2), 233–251 (2012)
6. Balas, E.: Intersection cuts – a new type of cutting planes for integer programming. *Op Res* **19**(1), 19–39 (1971)
7. Bao, X., Sahinidis, N.V., Tawarmalani, M.: Multiterm polyhedral relaxations for nonconvex, quadratically constrained quadratic programs. *Optim Method Softw* **24**(4–5), 485–504 (2009)

8. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex minlp. *Optim Method Softw* **24**(4–5), 597–634 (2009)
9. Berg, M.D., Cheong, O., Kreveld, M.V., Overmars, M.: *Computational geometry: algorithms and applications*. Springer-Verlag TELOS, Berlin (2008)
10. Bienstock, D., Chen, C., Munoz, G.: Outer-product-free sets for polynomial optimization and oracle-based cuts. arXiv preprint [arXiv:1610.04604](https://arxiv.org/abs/1610.04604) (2016)
11. Castillo, I., Kampas, F.J., Pintér, J.D.: Solving circle packing problems by global optimization: numerical results and industrial applications. *Eur J Op Res* **191**(3), 786–802 (2008)
12. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. *Math Program* **91**(2), 201–213 (2002)
13. Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: On the use of intersection cuts for bilevel optimization. *Math Program* **172**(1–2), 77–103 (2018)
14. Hifi, M., M'hallah, R.: A literature review on circle and sphere packing problems: models and methodologies. *Adv. Oper. Res.* **2009** (2009)
15. Hillestad, R.J., Jacobsen, S.E.: Reverse convex programming. *Appl Math Optim* **6**(1), 63–78 (1980)
16. Horst, R., Tuy, H.: *Global optimization: Deterministic approaches*. Springer Science and Business Media, NY (2013)
17. Jones, D.R.: A fully general, exact algorithm for nesting irregular shapes. *J Global Optim* **59**(2–3), 367–404 (2014)
18. Khajavirad, A.: Packing circles in a square: a theoretical comparison of various convexification techniques (2017). Preprint available via Optimization Online at [http://www.optimization-online.org/DB\\_FILE/2017/03/5911.pdf](http://www.optimization-online.org/DB_FILE/2017/03/5911.pdf)
19. Locatelli, M., Raber, U.: Packing equal circles in a square: a deterministic global optimization approach. *Discret Appl Math* **122**(1–3), 139–166 (2002)
20. López, C.O., Beasley, J.E.: A heuristic for the circle packing problem with a variety of containers. *Eur J Op Res* **214**(3), 512–525 (2011)
21. Markót, M.C.: Interval methods for verifying structural optimality of circle packing configurations in the unit square. *J Comput Appl Math* **199**(2), 353–357 (2007)
22. Markót, M.C., Csendes, T.: A new verified optimization technique for the “packing circles in a unit square” problems. *SIAM J Optim* **16**(1), 193–219 (2005)
23. Markót, M.C., Csendes, T.: A reliable area reduction technique for solving circle packing problems. *Computing* **77**(2), 147–162 (2006)
24. Misener, R., Floudas, C.A.: Antigone: algorithms for continuous/integer global optimization of nonlinear equations. *J Global Optim* **59**(2–3), 503–526 (2014)
25. Puranik, Y., Sahinidis, N.V.: Domain reduction techniques for global nlp and minlp optimization. *Constraints* **22**(3), 338–376 (2017)
26. Rebennack, S.: Computing tight bounds via piecewise linear functions through the example of circle cutting problems. *Math Method Op Res* **84**(1), 3–57 (2016)
27. Scheithauer, G.: *Introduction to Cutting and Packing Optimization: Problems, Modeling Approaches, Solution Methods*, vol. 263. Springer, NY (2017)
28. Serrano, F.: Intersection cuts for factorable minlp. In: *International Conference on Integer Programming and Combinatorial Optimization*, pp. 385–398. Springer (2019)
29. Stoyan, Y.G., Yas'kov, G.: A mathematical model and a solution method for the problem of placing various-sized circles into a strip. *Eur J Op Res* **156**(3), 590–600 (2004)
30. Szabó, P.G., Markót, M.C., Csendes, T.: Global optimization in geometry—circle packing into the square. In: *Essays and Surveys in Global Optimization*, pp. 233–265. Springer, NY (2005)
31. Tardella, F.: On the existence of polyhedral convex envelopes. In: *Frontiers in global optimization*, pp. 563–573. Springer, NY (2004)
32. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. *Math Program* **103**(2), 225–249 (2005)
33. Vigerske, S., Gleixner, A.: Scip: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. *Optim Method Softw* **33**(3), 563–593 (2018)
34. Wang, A., Hanselman, C.L., Gounaris, C.E.: A customized branch-and-bound approach for irregular shape nesting. *J Global Optim* **71**(4), 1–21 (2018)