A computational comparison of handling distance constraints in MINLP

Christopher Hojny[1] and Leo Liberti[2]

[1]*Eindhoven University of Technology, Department of Mathematics and Computer Science, PO Box 513, 5600 MB Eindhoven, The Netherlands, c.hojny@tue.nl*
[2]*LIX CNRS Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France, leo.liberti@polytechnique.edu*

April 30, 2026

Abstract

Minimum distance constraints (minDCs) appear in many geometric optimization problems. They pose major challenges for mixed-integer nonlinear programming (MINLP) due to their reverse-convexity. We develop new algorithms for tightening variable bounds in general MINLPs with minDCs. Because many such problems exhibit substantial symmetry, we further introduce a practical approach for handling rotation symmetries via separation of lexicographic constraints induced by Givens rotations. In a computational study, we examine the performance of the various methods and determine the scenarios in which each approach demonstrates superiority.

Keywords minimum distance constraints • rotation symmetry • bound tightening • cutting planes

1 Introduction

We consider mixed-integer nonlinear programs (MINLP)

$$\min_{x\in\mathbb{R}^n}\{f(x) : g(x) \leq 0 \text{ and } x_i \in \mathbb{Z} \text{ for } i \in I\},$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$, $g\colon \mathbb{R}^n \to \mathbb{R}^m$, $I \subseteq [n] \coloneqq \{1,\dots,n\}$, and some constraints are minimum distance constraints. A *minimum distance constraint (minDC)* is a constraint $\|y-z\|_2^2 \geq \delta$, where $y, z \in \mathbb{R}^d$ are d-dimensional subvectors of x and δ is either a variable or real number. Minimum distance constraints require that two points in d-dimensional space have Euclidean distance at least $\sqrt{\delta}$. They arise in many applications, e.g., the sphere packing problem [15], kissing number problem [11], or obnoxious facility location problem [12]. Since minDCs are highly nonconvex, solving MINLP problems involving such constraints is extremely difficult. For example, for the circle packing problem, where the goal is to find the largest radius such that k nonoverlapping circles fit into a unit square, the MINLP solver `SCIP` only finds a provably optimal solution within two hours when $k \leq 8$.

For many applications, the effective treatment of minDCs has been investigated. Most prominently, many variants of the circle packing problem have been studied. While many heuristic solution approaches have been suggested, see, e.g., [13] for an excellent overview for the problem of packing circles into a bigger circle, few exact methods exist for circle packing. One of the first such methods is [15], which suggests a branch-and-bound (BB) algorithm where the variables are the center points of the circles. If $D_y, D_z \subseteq \mathbb{R}^2$ are bounding boxes for the center points y and z in a minDC, it was observed that the box D_y can be replaced by $D' \subseteq D_y$ if the points in $D_y \setminus D'$ are "too close" to D_z. This observation is used to tighten variable bounds and thus accelerate the BB algorithm. Since this approach is vulnerable to numerical errors, [18] adapts the ideas

of [15] to interval arithmetic, in order to find guaranteed feasible solutions. Extending on the ideas of [15], [3, 4] derive structural properties of solutions of circle and point packing problems which enable a more effective pruning within BB. In [24], an alternative BB algorithm is suggested, where branching is not performed on the y- and z-variables, but on the differences $\Delta_i = y_i - z_i$, $i \in \{1, 2\}$. Exploiting bounds on Δ_i, intersection cuts and three feasibility-based bound tightening (FBBT) techniques are derived.

For general MINLPs with minDCs, however, the literature is scarcer and the previously mentioned results, except for the bounding box idea of [15], can not be applied. To handle minDCs in general MINLPs, the authors of [12] considered the case of minDCs $\|y - z\|_2 \geq \delta$ where one vector, say z, is a fixed point p. When there are multiple minDCs $\|y - p^1\|_2 \geq \delta_1, \ldots, \|y - p^k\|_2 \geq \delta_k$ and y is contained in a bounding box $D \subseteq \mathbb{R}^2$, they showed how to efficiently compute the convex hull of all $y \in D$ satisfying the k minDCs and exploit facet defining inequalities within spatial branch-and-bound.

In this article, we complement these results by considering the case when both y and z are variable vectors. Our aim is to devise new algorithms for handling minDCs and to computationally investigate scenarios in which the different methods demonstrate superiority. To this end, Section 2 explains the method given in [15] in more detail and suggests new methods for handling minDCs. Since many problems involving minDCs also admit a large amount of symmetries, it is inevitable to handle these symmetries in branch-and-bound because otherwise the performance of MINLP solvers deteriorates drastically, cf. [17,21]. While many sophisticated symmetry handling methods exist, handling the *rotation symmetries* arising in applications like the kissing number problem or packing spheres into a bigger sphere, have not yet been investigated to the best of our knowledge. We therefore complement our results on minDCs by developing novel cutting planes to handle rotation symmetries (Section 3). Section 4 concludes this article by a numerical comparison of the different algorithms.

Notation. For a variable x, we denote by $\underline{x}$ and $\bar{x}$ lower and upper bounds on x, respectively. Moreover, for $r > 0$ and $p \in \mathbb{R}^d$, the set $\mathcal{B}_r(p) = \{x \in \mathbb{R}^n : \|x - p\|_2 < r\}$ denotes the open Euclidean ball of radius r centered at p. By $\partial\mathcal{B}_r(p)$ we denote the boundary of $\mathcal{B}_r(p)$. The convex hull of a set $S \subseteq \mathbb{R}^n$ is denoted by $\operatorname{conv}(S)$.

2 Handling minimum distance constraints

Throughout this section, we consider minimum distance constraints

$$\sum_{i=1}^{d} (y_i - z_i)^2 \geq \delta^2, \tag{1}$$

where $y, z \in \mathbb{R}^d$ are variables with respective domains $D_y, D_z \subseteq \mathbb{R}^d$, and δ is a nonnegative variable. When solving MINLPs via spatial branch-and-bound, variable domains are typically real intervals. We therefore assume that the domains D_y and D_z are hyperrectangles, which we refer to as boxes for brevity.

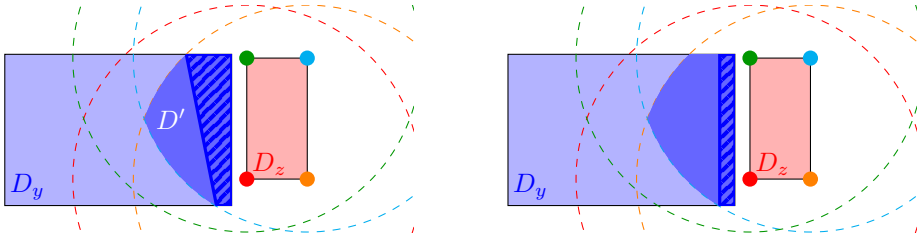
In the following, we explain the ideas of Locatelli and Raber [15] for shrinking the box domains D_y, D_z based on minDCs (Section 2.1.1). Afterwards, we describe a simpler algorithm for shrinking box domains (Section 2.1.2), present simple cutting planes (Section 2.1.3), and develop novel algorithms for shrinking box domains based on pairs of minDCs (Section 2.2).

2.1 A single minimum distance constraints

2.1.1 Locatelli and Raber's algorithm [15]

The approach of [15] proceeds in two steps.

1. Weaken the minDC (1) to $\sum_{i=1}^{d} (y_i - z_i)^2 \geq \underline{\delta}^2$, i.e., replace the variable right-hand side δ^2 by the constant lower bound $\underline{\delta}^2$.
2. Find a region $D' \subseteq D_z$ such that, for every $y \in D'$, one has $\max_{z \in D_z} \|y - z\|_2^2 < \underline{\delta}^2$.



(a) D_y becomes general polyhedron. (b) D_y remains a box.

Figure 1: Illustration of the method by Locatelli and Raber to generate convex domains D_y. In Fig. 1(a), D' corresponds to the hatched area.

Clearly, every $(y, z) \in D' \times D_z$ violates the weakened constraint (1) (i.e., with δ replaced by $\underline{\delta}$), and therefore D_y can be replaced by $D_y \setminus D'$. The crucial step of this procedure is Step 2. Although Locatelli and Raber discuss it only for $d = 2$, their results immediately apply to the general case $d \geq 2$.

To explain their ideas, let V_y and V_z denote the sets of vertices of D_y and D_z, respectively. For every $y \in D_y$, note that $\max_{z \in D_z} \|y - z\|_2^2 = \max_{z \in V_z} \|y - z\|_2^2$ because $\|y - z\|_2^2$ is convex in z. A point $y \in D_y$ can thus only be removed from D_y if its Euclidean distance to all $z \in V_z$ is less than $\underline{\delta}$, i.e., $y \in \bigcap_{z \in V_z} \mathcal{B}_{\underline{\delta}}(z) =: C(D_z, \underline{\delta})$. Consequently, the largest region D' that can be removed from D_y is

$$D' = D_y \cap C(D_z, \underline{\delta}),$$

see Fig. 1 for an illustration.

Since $D_y \setminus D'$ can be nonconvex (the light blue region in Fig. 1), Locatelli and Raber suggest to use a smaller set D', which is constructed as follows.

(i) For every $z \in V_z$, one computes the set J_z consisting of all intersection points of $\mathcal{B}_{\underline{\delta}}(z)$ with the edges of D_y. Let $J = V_y \cup \bigcup_{z \in V_z} J_z$

(ii) Select $D' = \operatorname{conv}(J')$, where $J' = \{x \in J : \max_{z \in V_z} \|x - z\|_2 < \underline{\delta}\}$ contains all vertices of D_y and intersection points of $C(D_z, \underline{\delta})$ with the edges of D_y that are at distance less than $\underline{\delta}$ to each vertex of D_z.

Although this choice guarantees that the set $D_y \setminus D'$ is convex and remains polyhedral, it has two downsides: maintaining an exact representation of $D_y \setminus D'$ potentially requires a lot of memory, and computing a facet description of $D_y \setminus D'$ is costly in arbitrary dimension. Locatelli and Raber therefore suggest replacing D' by a box that has a common facet with D_y, see Fig. 1(b). This approach ensures that $D_y \setminus D'$ remains a box.

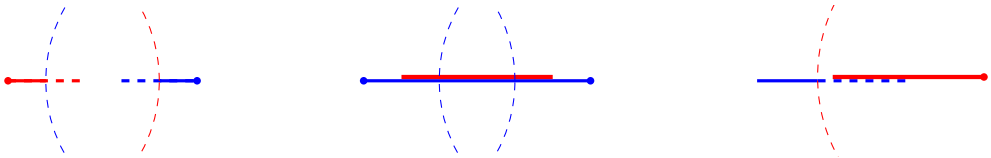
Observation 2.1. *Since $|V_y|, |V_z| \in O(2^d)$ and a d-dimensional box has $d2^{d-1}$ edges, the set J has size $|J| \in O(|V_z| \cdot d2^{d-1} + |V_y|) = O(d2^{2d-1})$. Checking containment of $x \in J$ in J' takes $O(d|V_z|)$ time because computing $\|x - z\|_2$ requires $O(d)$ time for a fixed $z \in D_z$. Consequently, the entire procedure of Locatelli and Raber takes $O(|J| \cdot d|V_z|) = O(d^2 2^{3d-1})$ time.*

2.1.2 A simple algorithm for shrinking box domains

Since the running time of the algorithm by Locatelli and Raber grows exponentially in d, it is only practical for moderate values of d. We therefore present a much simpler algorithm that scales linearly in the dimension d. While the algorithm of Locatelli and Raber provably finds the largest box D' that can be removed from D_y such that $D_y \setminus D'$ remains a box, our algorithm comes without such a guarantee.

For all $i \in [d]$, let $\operatorname{dist}_i = \max_{(y,z) \in D_y \times D_z} \|y_i - z_i\|_2$. For all $j \in [d]$, let $\Delta_j = \underline{\delta}^2 - \sum_{i \in [d] \setminus \{j\}} \operatorname{dist}_i^2$.

Lemma 2.2. *With respect to Eq. (1), we have $(y_j - z_j)^2 \geq \Delta_j^2$ for every $j \in [d]$.*



(a) Case 1: dashed parts can be removed from the domains.

(b) Case 2: the blue interval is not too close to any of the endpoints of the red interval.

(c) Case 3: dashed parts can be removed from the domain.

Figure 2: Illustration of the three different cases of Proposition 2.3. The red and blue lines correspond to the interval domain of y_j and z_j, respectively.

Proof. For any dimension index $j \in [d]$, consider Eq. (1) in the following form:

$$\sum_{i \in [d] \setminus \{j\}} (y_i - z_i)^2 + (y_j - z_j)^2 \geq \delta^2.$$

This yields $(y_j - z_j)^2 \geq \delta^2 - \sum_{i \in [d] \setminus \{j\}} (y_i - z_i)^2$. Now, we have $\underline{\delta}^2 \leq \delta^2$, as well as $(y_i - z_i)^2 \leq \mathrm{dist}_i^2$, which implies that

$$\delta^2 - \sum_{i \in [d] \setminus \{j\}} (y_i - z_i)^2 \geq \underline{\delta}^2 - \sum_{i \in [d] \setminus \{j\}} \mathrm{dist}_i^2 = \Delta_j,$$

as claimed. □

Based on this observation, we can devise some simple rules for reducing variable domains, see Fig. 2 for an illustration.

Proposition 2.3. *Consider a single minDC (1) with variable domains D_y and D_z, and let $j \in [n]$. Let $(y', z') \in D_y \times D_z$ satisfy (1). Then,*

1. *if $\bar{y}_j \leq \underline{z}_j$, we have*

$$y'_j \leq \min\{\bar{y}_j, \bar{z}_j - \Delta_j\},$$
$$z'_j \geq \max\{\underline{z}_j, \underline{y}_j + \Delta_j\},$$

   and symmetrically if $\bar{z}_j \leq \underline{y}_j$;

2. *if $\underline{z}_j \leq \underline{y}_j \leq \bar{y}_j \leq \bar{z}_j$, then $\underline{y}_j - \underline{z}_j \geq \Delta_j$ or $\bar{z}_j - \underline{y}_j \geq \Delta_j$, and symmetrically if $\underline{y}_j \leq \underline{z}_j \leq \bar{z}_j \leq \bar{y}_j$;*

3. *if $\underline{z}_j < \underline{y}_j \leq \bar{z}_j < \bar{y}_j$, we have*

$$\bar{z}_j - \underline{y}_j < \Delta_j \text{ and } \bar{y}_j - \bar{z}_j < \Delta_j \implies z'_j \leq \min\{\bar{z}_j, \bar{y}_j - \Delta_j\};$$
$$\underline{y}_j - \underline{z}_j < \Delta_j \text{ and } \bar{z}_j - \underline{y}_j < \Delta_j \implies y'_j \geq \max\{\underline{y}_j, \underline{z}_j + \Delta_j\},$$

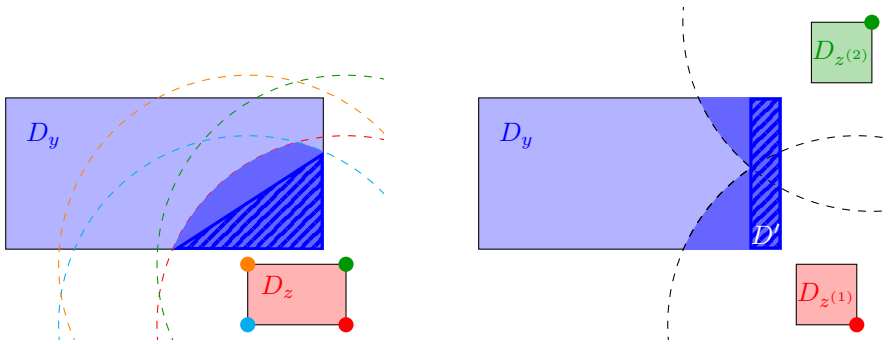
   and symmetrically if $\underline{y}_j < \underline{z}_j \leq \bar{y}_j < \bar{z}_j$.

Proof. Recall that, since D_y and D_z are boxes, the domain of variable y_j and z_j is an interval for each $j \in [d]$. Let $(y', z') \in D_y \times D_z$ adhere to (1). Then, Lemma 2.2 implies $(y'_j - z'_j)^2 \geq \Delta_j^2$. For each of the three cases, we discuss one variant as the other one follows by symmetry.

In the first case, $(y'_j - z'_j)^2 \geq \Delta_j^2$ yields $z'_j - y'_j \geq \Delta_j$ since $z'_j \geq y'_j$. Hence, $y'_j \leq z'_j - \Delta_j \leq \bar{z}_j - \Delta_j$. Together with the trivial inequality $y'_j \leq \bar{y}_j$, we conclude $y'_j \leq \min\{\bar{y}_j, \bar{z}_j - \Delta_j\}$.

In the second case, the domain of y_j is a subset of the domain of z_j. For the sake of contradiction, suppose $\underline{y}_j - \underline{z}_j < \Delta_j$ and $\bar{z}_j - \underline{y}_j < \Delta_j$. By convexity, this means that every point in the interval $[\underline{y}_j, \bar{y}_j]$ has distance less than Δ_j to both $\underline{z}_j$ and $\bar{z}_j$. Consequently, the distance between all points in $[\underline{y}_j, \bar{y}_j]$ and $[\underline{z}_j, \bar{z}_j]$ is less than Δ_j, contradicting Lemma 2.2.

For the last case, we only discuss the first subcase, because the second one follows analogously. By the same arguments as in Case 2, $\bar{z}_j - \underline{y}_j < \Delta_j$ and $\bar{y}_j - \bar{z}_j < \Delta_j$ implies that every point in $[\underline{y}_j, \bar{y}_j]$ has distance less than Δ_j to $\bar{z}_j$. Hence, $z'_j \leq \bar{y}_j - \Delta_j$ needs to hold. Together with the trivial inequality $z'_j \leq \bar{z}_j$, we obtain $z'_j \leq \min\{\bar{z}_j, \bar{y}_j - \Delta_j\}$. □



(a) Cutting plane for one minDC. (b) Shrinking box domains for two minDCs.

Figure 3: Illustration of domain reductions derived via cutting planes and two minDCs.

For each $i \in [d]$, the value dist_i can be computed in constant time, because $(x_i - y_i)^2$ is a convex function and its maximum over the domain $D = [\underline{y}_i, \bar{y}_i] \times [\underline{z}_i, \bar{z}_i]$ is attained at one of the (at most) four vertices of D. The variable bounds of Proposition 2.3 can therefore be computed in $O(d)$ time.

2.1.3 Simple cutting planes

The algorithms presented in the previous section can only shrink the domain D_y if there is a facet of D_y that is contained in $\mathcal{B}_{\underline{\delta}}(z)$ for every $z \in V(D_z)$. In the situation of Fig. 3(a), one therefore needs to use cutting planes to cut off parts of $D_y \cap C(D_z, \underline{\delta})$. Since the method by Locatelli and Raber for constructing the convex hull of $D_y \cap C(D_z, \underline{\delta})$ is rather expensive, we suggest a simpler set of cutting planes.

Let us assume that D_y is full-dimensional and $y' \in V(D_y) \cap C(D_z, \underline{\delta})$. To find a cutting plane that cuts off y', let E be the set of d (polyhedral) edges of D_y that are incident with y'. For $e \in E$, let $p(e) = \text{argmax}\{\|y' - p\|_2 : p \in e \cap C(D_z, \underline{\delta})\}$, i.e., $p(e)$ is the point on e that is farthest away from y' while still contained in $C(D_z, \underline{\delta})$. Since $y' \in C(D_z, \underline{\delta})$, the simplex $\text{conv}\{y'\} \cup \{p(e) : e \in E\}$ is contained in $C(D_z, \underline{\delta})$ and the facet defined by $\{p(e) : e \in E\}$ can be used as a cutting plane to cut off y', see Fig. 3(a).

Observation 2.4. *For fixed $y' \in V(D_y)$ and $e \in E$, $p(e)$ can be found by computing the intersection points of e and $\partial\mathcal{B}_{\underline{\delta}}(z)$ for every $z \in V(D_z)$. Computing the intersections of e with a single sphere involves $O(d)$ numbers; a single intersection can thus be computed in $O(d)$ time. As D_z has $O(2^d)$ vertices, the set $\{p(e) : e \in E\}$ can be found in $O(d2^d)$ time. Finding the hyperplane spanned by $\{p(e) : e \in E\}$ requires the solution of a system of d linear equations. The time for computing a single hyperplane is thus dominated by the time to find $\{p(e) : e \in E\}$ and is therefore $O(d2^d)$.*

2.2 Handling pairs of minimum distance constraints

Consider the situation of Fig. 3(b), where we are given two minDCs

$$\|y - z^{(1)}\|_2^2 \geq \delta_1^2 \qquad \text{and} \qquad \|y - z^{(2)}\|_2^2 \geq \delta_2^2 \tag{2}$$

that constrain a common variable vector $y \in \mathbb{R}^d$. In this example, the method of Locatelli and Raber fails to find a smaller box domain for y because every facet of D_y contains a point that is at distance at least δ_1 (resp. δ_2) to a point in $D_{z^{(1)}}$ (resp. $D_{z^{(2)}}$). By considering both minDCs simultaneously, however, the hatched area D' can be removed from D_y because every point in D' has distance below δ_1 to a point in $D_{z^{(1)}}$ or distance below δ_2 to a point in $D_{z^{(2)}}$. In the following, we show how to systematically exploit this observation. To this end, we proceed in two steps. First, we discuss how to decide whether a given box domain D' can be removed from D_y when we are given two minDCs as in (2). Second, we show how to (heuristically) find candidates for such domains D'.

Deciding whether D' can be removed. Suppose we have guessed a box domain D' that we want to remove. Then, the following proposition shows how to verify whether (2) implies $y \in D_y \backslash D'$ in case both D_{z^1} and D_{z^2} are singletons.

Proposition 2.5. *Let $D_y, D' \subseteq \mathbb{R}^d$ be boxes, let $D_{z^{(1)}} = \{p\}, D_{z^2} = \{q\} \subseteq \mathbb{R}^d$, and let $\delta_1, \delta_2 > 0$. Then, $D' \subseteq \mathcal{B}_{\delta_1}(p) \cup \mathcal{B}_{\delta_2}(q)$ if and only if*

- *the vertices of D' are contained in $\mathcal{B}_{\delta_1}(p) \cup \mathcal{B}_{\delta_2}(q)$, and*
- *for every edge e of D', we have $e \cap \partial\mathcal{B}_{\delta_1}(p) \subseteq \mathcal{B}_{\delta_2}(q)$.*

Proof. First, assume $D' \subseteq \mathcal{B}_{\delta_1}(p) \cup \mathcal{B}_{\delta_2}(q) =: B$. Then, the vertices of D' are obviously contained in B. Moreover, B can only cover D' when $\mathcal{B}_{\delta_2}(q)$ contains $D' \cap \partial\mathcal{B}_{\delta_1}(p)$.

Second, assume that the two properties hold. Let V_1 and V_2 be the vertices of D' that are contained in $\mathcal{B}_{\delta_1}(p)$ and $\mathcal{B}_{\delta_2}(q)$, respectively. Moreover, let I be the intersection points of $\partial\mathcal{B}_{\delta_1}(p)$ and the edges of D'. We need to show that $\mathcal{B}_{\delta_2}(q)$ contains $A := D' \setminus \mathcal{B}_{\delta_1}(p)$. Since A is a reverse convex set, $A \subseteq \operatorname{conv}(I \cup V_2)$. Thus, $A \subseteq \mathcal{B}_{\delta_2}(q)$ because $\operatorname{conv}(I \cup V_2) \subseteq \mathcal{B}_{\delta_2}(q)$ due to the convexity of $\mathcal{B}_{\delta_2}(q)$. □

When $D_{z^{(1)}}$ and $D_{z^{(2)}}$ are not singletons, then the same arguments as in Sec. 2.1.1 can be used to show that D' can be removed from D_y: for every pair of vertices $(p, q) \in V(D_{z^{(1)}}) \times V(D_{z^{(2)}})$, one needs to check whether Prop. 2.5 applies.

Observation 2.6. *For fixed $p \in V(D_{z^{(1)}})$ and $q \in V(D_{z^{(2)}})$, checking $V(D') \subseteq \mathcal{B}_{\delta_1}(p) \cup \mathcal{B}_{\delta_2}(q)$ takes $O(d2^d)$ time, because $|V(D')| \in O(2^d)$ and computing the distance between two points requires linear time. Finding the intersection of $\partial\mathcal{B}_{\delta_1}(p)$ with an edge of D' and computing the distance to q takes $O(d)$ time. The entire procedure for fixed p and q thus takes $O(d2^d + d^2 2^{d-1}) = O(d^2 2^d)$ time, because D' has $O(d2^{d-1})$ edges. Running this algorithm for all $O(2^{2d})$ pairs (p, q) of vertices therefore takes $O(d^2 2^{3d})$ time.*

Finding D'. To find a candidate box D' to remove, we propose two heuristic approaches. Since removing a box D' from D_y results in changing a variable lower or upper bound on some variable y_i, $i \in [d]$, the first approach selects the new bound via bisection on the interval $[\underline{y}_i, \bar{y}_i]$ and checks whether the removed box satisfies the properties of Proposition 2.5. If not, the size of the removed box is decreased; otherwise, the size of the removed box is increased. Since the time needed to check the requirements of Proposition 2.5 is rather high, we restrict the number of iterations in the bisection to three in our implementation. Moreover, to increase the chance of finding a reduction, we split the interval with a ratio of 1:9, where D' gets 10 % of the original interval length.

The second approach is restricted to the case $d \in \{2, 3\}$. It selects a coordinate y_i, $i \in [d]$, and uses geometric ideas to improve a variable bound, say $\bar{y}_i$. Motivated by Fig. 3(b), the improved upper bound for y_i for $d = 2$ is defined by the intersection of two spheres or the intersection of a sphere and an edge of D_y. Let E be the edges of D_y that are orthogonal to the facet defined by $y_i = \bar{y}_i$. For each $p \in V(D_{z^{(1)}})$, $q \in V(D_{z^{(2)}})$, and $e \in E$, we collect the intersection points of e with $\partial\mathcal{B}_{\delta_1}(p)$ and $\partial\mathcal{B}_{\delta_2}(q)$. Let I be the set of all such intersection points. Moreover, for $d = 2$, compute the intersections of the spheres $\partial\mathcal{B}_{\delta_1}(p)$ and $\partial\mathcal{B}_{\delta_2}(q)$. Let J_2 be the set of all intersections for which the i-th coordinate lies in the interval $[\underline{y}_i, \bar{y}_i]$. The guess for the box D' is then given by reducing the upper bound $\bar{y}_i$ to $\max\{y_i : y \in I \cup J_2\}$ and keeping all other bounds identical.

When $d = 3$, we also compute the set I. But since the intersection of two spheres can be a circle, we check the heights y_i at which these circles intersect the facets of D_y that are incident with the facet defined by $y_i = \bar{y}_i$. Let J_3 be the set of all intersections for which the i-th coordinate lies in the interval $[\underline{y}_i, \bar{y}_i]$. Then, as for $d = 2$, our guess for the box D' is given by reducing the upper bound $\bar{y}_i$ to $\max\{y_i : y \in I \cup J_3\}$ and keeping all other bounds identical.

3 Handling rotation symmetries

In many geometric optimization problems, the task is to find an optimal arrangement of n points in d-dimensional Euclidean space. For example, in the sphere packing problem [15], the goal is to decide whether n identical spheres fit into a d-dimensional box, and a variant of the kissing number

problem [11] asks whether n unit spheres can be placed in $\mathbb{R}^d$ such that each touches a common unit sphere. Both problems can be reduced to locating the center points of the n spheres in $\mathbb{R}^d$ subject to appropriate constraints. The center points can be modeled as a matrix $X \in \mathbb{R}^{n\times d}$, where row $i \in [n]$ of this matrix contains the i-th point. In the following, we denote the i-th row of X by X^i, whereas the j-th column is denoted by X_j.

Since the n spheres are identical, both problems possess inherent symmetries: every exchange of two spheres in a solution yields a symmetric one. Moreover, in the sphere packing problem, whenever two box dimensions coincide, exchanging these two dimensions in an arrangement results in a symmetric solution. In the kissing number problem, any rotation around the center of the common sphere also produces a symmetric assignment.

When solving these problems with spatial branch-and-bound algorithms, it is well known that such symmetries must be addressed; otherwise, symmetric regions of the search space are explored repeatedly, leading to unnecessarily long computation times [14, 17, 21]. While permutation symmetries arising from exchanging spheres or dimensions have been studied extensively and many sophisticated symmetry handling methods exist, see, e.g., [9, 16, 20, 22], rotational symmetries have, to the best of our knowledge, received significantly less attention. We therefore describe a mechanism for handling rotation symmetries.

When all spheres (resp. dimensions) are symmetric, a common approach to handle symmetries is to sort the rows (resp. columns) of X lexicographically, i.e.,

$$X^i \geq_{\text{lex}} X^{i+1} \text{ for all } i \in [n-1] \text{ and } X_j \geq_{\text{lex}} X_{j+1} \text{ for all } j \in [d-1]. \tag{3}$$

cf. [6]. Moreover, when the box D is centered at the origin and symmetric w.r.t. reflections along the standard hyperplanes $x_j = 0$, $j \in [d]$, these reflection symmetries carry over to symmetries of the matrix X. Usually, this is handled by enforcing that some variables can only take nonnegative values and can be used in conjunction with (3), see [6, 10].

In the presence of rotation symmetries $\rho\colon \mathbb{R}^d \to \mathbb{R}^d$, such a symmetry acts on an assignment X by applying ρ row-wise. Due to [22], a valid symmetry handling approach is to enforce that a solution adheres to

$$(X^1, X^2, \dots, X^n) \coloneqq X \geq_{\text{lex}} \rho(X) \coloneqq (\rho(X^1), \rho(X^2), \dots, \rho(X^n)), \tag{4}$$

i.e., the rows of X are sorted lexicographically w.r.t. any rotation ρ. In particular, this approach is compatible with the previously described methods because all require solution vectors to be sorted lexicographically. The result of [22], however, is purely theoretical and does not explain how (4) can be enforced.

In the following, we show how rotation symmetries can be handled for the special class of Givens rotations ρ, i.e., there exist two distinct coordinates $j, j' \in [d]$, $j < j'$, and an angle $\alpha \in [0, 2\pi)$ such that ρ acts like an α-rotation in the j-j'-plane and keeps the remaining dimensions invariant. By restricting the lexicographic comparison $X \geq_{\text{lex}} \rho(X)$ to the first entry, one can derive

$$(X_{1,j}, X_{1,j'}) \geq_{\text{lex}} (\cos(\alpha)X_{1,j} - \sin(\alpha)X_{1,j'}, \sin(\alpha)X_{1,j} + \cos(\alpha)X_{1,j'}). \tag{5}$$

Although explicitly adding the simple constraints (5) for all $\alpha \in [0, 2\pi)$ is impossible as there are infinitely many, it is still possible to separate these constraints. In our current implementation, we only focus on the first part of the lexicographic comparison, i.e., the condition

$$X_{1,j} \geq \cos(\alpha)X_{1,j} - \sin(\alpha)X_{1,j'} \qquad \Longleftrightarrow \qquad (1 - \cos(\alpha))X_{1,j} + \sin(\alpha)X_{1,j'} \geq 0.$$

To separate these inequalities, we compute $\alpha \in [0, 2\pi)$ that minimizes the left-hand side expression for a fixed solution X to be separated. This can be done by basic calculus techniques in constant time.

Moreover, we can apply these cuts also to rows with an index $i > 1$ if $X_{i',j} = X_{i',j'} = 0$ for all $i' \in [i-1]$. That is, if $X_{i',j} = X_{i',j'} = 0$ for all $i' \in [i-1]$, a valid symmetry handling inequality is given by

$$(X_{i,j}, X_{i,j'}) \geq_{\text{lex}} (\cos(\alpha)X_{i,j} - \sin(\alpha)X_{i,j'}, \sin(\alpha)X_{i,j} + \cos(\alpha)X_{i,j'}).$$

Furthermore, one can combine Givens rotations and reflection symmetries, i.e., one derives analogous inequalities from the condition $(X_{i,j}, X_{i,j'}) \geq_{\text{lex}} \rho(\pm X_{i,j}, \pm X_{i,j'})$.

4 Numerical experience

To evaluate the impact of the different algorithms for finding domain reductions based on minDCs, we have implemented all techniques in the global MINLP solver `SCIP`. The experiments have been conducted for three different test sets: general benchmark instances from the library MINLPLib [2] for which our implementation could automatically detect minDCs (MINLP), instances of the obnoxious facility location (OFL) problem from EuclidLib[1], and variants of the kissing number problem and the problem of packing identical spheres into a bigger sphere (GEOM). The dimension d of minDCs for the MINLP test set is $d \in \{1, \dots, 5\}$, for OFL $d = 2$, and for GEOM $d \in \{2, 3\}$. Moreover, for GEOM we also test the impact of handling rotation symmetries via our proposed cutting planes.

Hardware and software specifications. All experiments have been conducted on a Linux Cluster with Intel Xeon E5-1620 v4 3.5 GHz quad core processors and 32 GB memory. The code was executed single-threaded with a time limit of 2 h. We use `SCIP` 10.0.0 [7] as branch-and-bound framework; LP relaxations are solved using `SoPlex` 8.0.0 and nonlinear problems are solved by `Ipopt` 3.14.20 [23]. Symmetries are detected by `sassy` 1.1 [1] and `Nauty` 2.8.8 [19]. We use `SCIP`'s default parameters except for the optimality gap, which we set to 0.5 %.

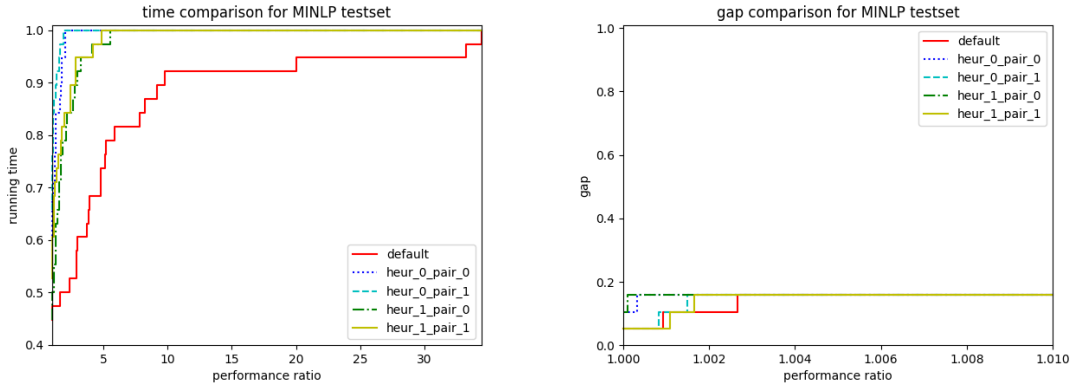
Results for MINLP and OFL. Due to space restrictions, we discuss the performance of the different algorithms mainly using performance profiles [5] and refer the reader to an online supplement [8] for detailed numerical results and our code, which is also available on GitHub[2]. For all test sets, we compare the performance of the different settings based on time and gap at termination. For all test sets, we compare the performance of the different settings based on time and gap at termination. The profiles comparing the running time are based on all instances of the respective test set that could be solved by at least one setting within the time limit; the profiles comparing gap at termination are based on the instances that could not be solved by any setting within the time limit. Figs. 4 and 5 show the results for the MINLP and OFL test sets, respectively, comparing our different domain reduction techniques and cutting planes disabled. The *default* setting refers to `SCIP`'s default configuration without any of our techniques; *heur_0_pair_0* and *heur_1_pair_0* refer to the techniques from Secs. 2.1.1 and 2.1.2, respectively; and *heur_0_pair_1* and *heur_1_pair_1* refer to the algorithm of Sec. 2.2 with the geometrically inspired techniques and bisection, respectively. Since the variants using *pair_1* can be expensive, we inspect at each node of the branch-and-bound tree which variables have a changed domain in comparison to the parent node. The algorithms for pairs of minDCs are then only executed for pairs with at least one changed variable domain.

Concerning running time, for MINLP, all techniques for handling minDCs substantially improve `SCIP`'s running time, with the settings involving *heur_0* performing better in general. For OFL, however, there is a clear distinction between *pair_0* and *pair_1*. While *pair_0* outperforms the *default* setting, *pair_1* performs worse on most instances. The reason is that the OFL instances contain far more minDCs than the MINLP instances, making the *pair_1* algorithms substantially more expensive. This behavior is also reflected in the profiles describing the gap at termination: for the more difficult instances, the algorithms using *pair_1* are too expensive to effectively close the gap.

Overall, *heur_0_pair_0* is the most performant configuration. Despite its simplicity, the method proposed in Sec. 2.1.2 solves for both MINLP and OFL as many instances as the *heur_0_pair_0* setting. In particular, the performance difference is smaller for the OFL instances than for the MINLP instances. We conclude that, especially in the presence of many minDCs, it is advantageous to use more inexpensive algorithms for handling minDCs that can be invoked frequently during the spatial branch-and-bound process to benefit from locally updated variable bounds.

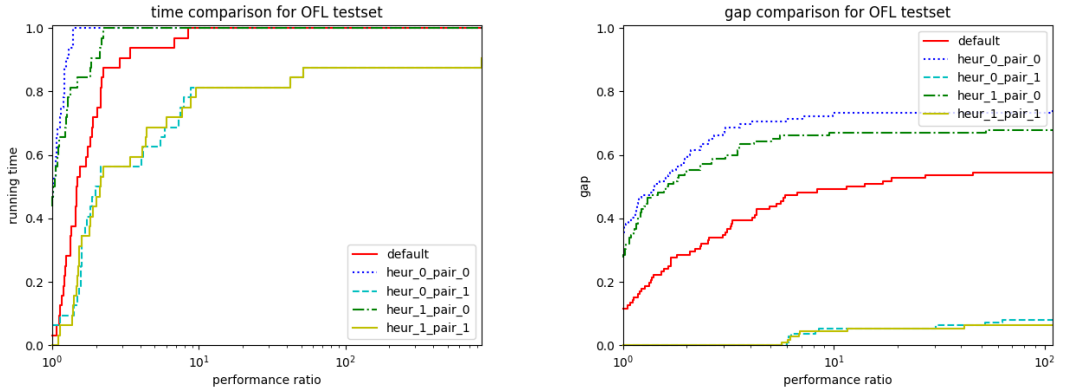
We have also conducted experiments to investigate the impact of the cutting planes discussed in Sec. 2.1.3. To this end, we either disabled the separation of these cutting planes or only separate them at every k-th level of the branch-and-bound tree with $k \in \{1, 10\}$. In general, we observed that the less frequent cutting planes are separated, the better the performance of the overall algorithm. We therefore disabled the separation of cutting planes in the following experiments.

[1] https://github.com/anatoliy-kuznetsov/EuclidLib
[2] https://github.com/christopherhojny/distance-constraints



(a) Time (b) Gap

Figure 4: Performance profiles for the MINLP test set.

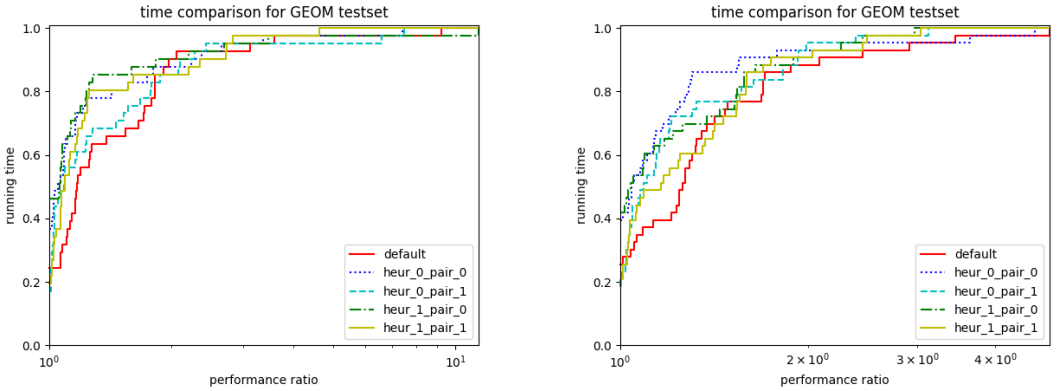


(a) Time (b) Gap

Figure 5: Performance profiles for OFL test set.

Results for GEOM. The GEOM test set consists of 2- and 3-dimensional versions of the following problems. Finding the maximum radius such that n identical spheres fit into a sphere of radius 1, and finding the maximum radius such that the kissing number problem for n spheres has a solution. For the former, we adapt the MINLP model (1) from [10] for packing spheres into a box, where $n \in \{4, \ldots, 20\}$ when $d = 2$ and $n \in \{4, \ldots, 10\}$ when $d = 3$. For the kissing number problem we use Models (1)–(5) and (6) from [11]. The first model uses $n \in \{4, \ldots, 20\}$ when $d = 2$ and $n \in \{4, \ldots, 10\}$ when $d = 3$; the second model uses $n \in \{4, \ldots, 20\}$ when $d = 2$ and $n \in \{4, \ldots, 16\}$ when $d = 3$.

Fig. 6 compares the results of the different settings and additionally illustrates the impact of handling rotation symmetries. As for the other test sets, handling minDCs greatly improves SCIP's performance. When rotation symmetries are not handled (Fig. 6(a)), the settings with *heur_1* dominate those with *heur_0*. In particular, the inexpensive method from Sec. 2.1.2 is the most performant on most instances. Since most of the constraints of the GEOM instances are minDCs, this indicates that inexpensive heuristic methods are essential for successfully handling minDCs, as the bound-reduction techniques are invoked frequently during the branch-and-bound search. It is therefore important to derive good reductions quickly rather than investing substantial time into computing potentially only marginally better variable domain reductions. When rotation symmetries are also handled (Fig. 6(b)), the situation changes. Here, *heur_0_pair_0* is faster in general. A possible explanation is that handling rotation symmetries leads to a smaller branch-and-bound tree. With a smaller tree, stronger domain reductions from handling minDCs can accelerate



(a) Without handling of rotation symmetries.

(b) With handling of rotation symmetries.

Figure 6: Performance profiles for GEOM test set.

the solution process. For the more difficult instances, however, *heur_0_pair_0* performs worse than *heur_1_pair_0*. In this regime, a larger branch-and-bound tree must be explored, and it becomes more important to use an inexpensive domain reduction algorithm.

Since the performance profiles do not allow to assess the overall running time, we also present some numbers to highlight the effect of handling rotation symmetries. For a fair comparison, we restrict the comparison to the subset of 35 instances that could be solved by all methods within the time limit. Here, handling rotation symmetries improves the running time of both the *default* and *heur_0_pair_0* setting by 78–80 %. Since the coefficients of the cutting planes (5) are sine and cosine expressions, we noted that these inequalities are numerically difficult to handle. In fact, for six instances we observed inconsistencies between the optimal objective values reported by the settings that handle and do not handle rotation symmetries.

Conclusion. Our computational study demonstrates that efficiently handling minDCs is crucial for achieving good performance across all test sets. Simple and inexpensive domain reduction techniques, in particular the heuristic from Sec. 2.1.2, consistently perform very well and can outperform more elaborate methods on harder instances. For instances with many minDCs, such as the OFL and GEOM test sets, lightweight reductions that are not based on pairs of minDCs are especially beneficial, as they can be applied frequently during the spatial branch-and-bound process without incurring excessive overhead. Handling rotation symmetries further improves performance on the GEOM instances.

In our experiments, we had to add methods for handling rotation symmetries manually, as we currently do not know how to detect such symmetries automatically. Future research should therefore investigate how to automatically identify and exploit rotation symmetries to further enhance the performance of spatial branch-and-bound algorithms. Moreover, an important direction is to derive methods for handling rotation symmetries that are more stable numerically.

Acknowledgments. LL was partly sponsored by the ANR "Evariste" project ANR-24-CE23-1621.

References

[1] M. Anders, P. Schweitzer, and J. Stieß. Engineering a Preprocessor for Symmetry Detection. In L. Georgiadis, editor, *21st International Symposium on Experimental Algorithms (SEA 2023)*, volume 265 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:21, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[2] M. Bussieck, A. Drud, and A. Meeraus. MINLPLib — A collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing*, 15(1), 2003.

[3] A. Costa. Valid constraints for the point packing in a square problem. *Discrete Applied Mathematics*, 161(18):2901–2909, 2013.

[4] A. Costa, P. Hansen, and L. Liberti. On the impact of symmetry-breaking constraints on spatial branch-and-bound for circle packing in a square. *Discrete Applied Mathematics*, 161:96–106, 2013.

[5] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[6] C. Hojny. Detecting and handling reflection symmetries in mixed-integer (nonlinear) programming and beyond. *Mathematical Programming Computation*, 18:31–78, 2025.

[7] C. Hojny, M. Besançon, K. Bestuzheva, S. Borst, J. Dionísio, J. Ehls, L. Eifler, M. Ghannam, A. Gleixner, A. Göß, A. Hoen, J. von Holly-Ponientzietz, R. van der Hulst, D. Kamp, T. Koch, K. Kofler, J. Lentz, M. Lübbecke, S. J. Maher, P. M. Meinhold, G. Mexi, T. Mohr, E. Mühmer, K. K. Patel, M. E. Pfetsch, S. Pokutta, C. R. Groba, F. Serrano, Y. Shinano, M. Turner, S. Vigerske, M. Walter, D. Weninger, and L. Xu. The SCIP Optimization Suite 10.0. Technical report, arXiv 2511.18580, 2025.

[8] C. Hojny and L. Liberti. Online supplement for "A computational comparison of handling distance constraints in MINLP". https://www.doi.org/10.5281/zenodo.19914665.

[9] V. Kaibel and M. E. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 114(1):1–36, 2008.

[10] A. Khajavirad. The circle packing problem: A theoretical comparison of various convexification techniques. *Operations Research Letters*, 57:107197, 2024.

[11] S. Kucherenko, P. Belotti, L. Liberti, and N. Maculan. New formulations for the kissing number problem. *Discrete Applied Mathematics*, 155(14):1837–1841, 2007.

[12] A. Kusnetsov and N. V. Sahinidis. Simultaneous convexification for the planar obnoxious facility location problem. *Journal of Global Optimization*, 92:1–20, 2025.

[13] X. Lai, J.-K. Hao, D. Yue, Z. Lü, and Z.-H. Fu. Iterated dynamic thresholding search for packing equal circles into a circular container. *European Journal of Operational Research*, 299(1):137–153, 2022.

[14] L. Liberti. Symmetry in mathematical programming. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *IMA Series*, pages 263–286. Springer, New York, 2012.

[15] M. Locatelli and U. Raber. Packing equal circles in a square: a deterministic global optimization approach. *Discrete Applied Mathematics*, 122(1):139–166, 2002.

[16] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94(1):71–90, 2002.

[17] F. Margot. Symmetry in integer linear programming. In M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors, *50 Years of Integer Programming*, pages 647–686. Springer, 2010.

[18] M. C. Markót and T. A. Csendes. A reliable area reduction technique for solving circle packing problems. *Computing*, 77:147–162, 2006.

[19] B. D. McKay and A. Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014.

[20] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. *Mathematical Programming*, 126(1):147–178, 2011.

[21] M. E. Pfetsch and T. Rehn. A computational comparison of symmetry handling methods for mixed integer programs. *Mathematical Programming Computation*, 11(1):37–93, 2019.

[22] J. van Doornmalen and C. Hojny. A unified framework for symmetry handling. *Mathematical Programming*, 212:217–271, 2025.

[23] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

[24] A. Wang and C. E. Gounaris. On tackling reverse convex constraints for non-overlapping of unequal circles. *Journal of Global Optimization*, 80:357–385, 2021.