On the impact of symmetry-breaking constraints on spatial Branch-and-Bound for circle packing in a square[☆]

Alberto Costa[a,*], Pierre Hansen[a,b], Leo Liberti[a]

[a] *LIX, École Polytechnique, 91128 Palaiseau, France*
[b] *GERAD, HEC, 3000 chemin de la Côte-S.te-Catherine, H3T 2A7 Montreal, Canada*

Abstract

We study the problem of packing equal circles in a square from the mathematical programming point of view. We discuss different formulations, we analyse formulation symmetries, we propose some symmetry breaking constraints and show that not only do they tighten the convex relaxation bound, but they also ease the task of local NLP solution algorithms in finding feasible solutions. We solve the problem by means of a standard spatial Branch-and-Bound implementation, and show that our formulation improvements allow the algorithm to find very good solutions at the root node.

Keywords: symmetry, reformulation, narrowing, circle packing, nonconvex NLP

1. Introduction

Circle packing is a classic problem in mathematics [3, 4]. Applications include cutting problems [5, 6, 7] and container loading [8, 9]; for an application-driven survey see [10]. We consider the following decision problem:

> Packing Equal Circles in a Square (PECS). Given an integer $n > 0$ and a rational radius $r > 0$, can n circles of radius r be packed in a unit square in such a way that the interiors of the circles have pairwise empty intersection?

The optimization version of the PECS asks for the maximum radius r allowing a packing of n circles in the unit square (the acronym PECS was used in [11] to indicate the equivalent problem of packing n unit circles in the smallest possible square).

[☆] This paper extends the extended abstracts [1, 2]. Financial support by grants: Digiteo 2009-14D "RMNCCO", Digiteo 2009-55D "ARM" is gratefully acknowledged.

[*] Corresponding author

Email addresses: costa@lix.polytechnique.fr (Alberto Costa), pierre.hansen@gerad.ca (Pierre Hansen), liberti@lix.polytechnique.fr (Leo Liberti)

1.1. Solution approaches

Many different approaches were proposed to solve PECS, stemming from global optimization and geometry. The classical formulation of the PECS is as a quadratically constrained problem [12, 13, 14, 15], but it can also be formulated as a d.c. (difference of convex functions) program [16]. A geometric Branch-and-Bound (BB) method is introduced in [15], together with some characterizations of optimal solutions: (i) there is an optimal solution such that at each vertex v of the square either a circle is adjacent to both edges e_1, e_2 incident to v, or two adjacent circles are adjacent to e_1 and e_2 respectively; (ii) there is an optimal solution such that the maximum distance between two circle adjacency points on each edge does not exceed $4r$. An interval BB described in [4] is used to find guaranteed optimal packings whilst verifying floating point computations. Another approach consists in finding a relationship between the number of circles and the structure of the packings (patterns): if these patterns can be found, it is easy to determine the coordinates of the centers of the circles; some experiments in this direction were performed in [17, 18].

Heuristics include minimization of energy function, where the circle centers are considered as electrical charges repulsing each other [17], billiard simulation method [18], perturbation method [19], TAMSASS-PECS (Threshold Accepting Modified Single Agent Stochastic Search for Packing Equal Circles in a Square) [20], some techniques based on the pattern finding [17, 18], simulation of the movement of smooth elastic discs in a container [21]. In [11], a formulation-based multi-start heuristic with a combinatorial element (circles get moved to the largest vacant area of the current configuration before calling a local optimization procedure) is proposed for the PECS. Monotonic basin hopping heuristics have been proposed for packing equal and unequal circles in a square [22] and in a containing circle [23]. For more information, we refer to the book [4] and the surveys [12, 24].

1.2. Complexity: an unclear status

The PECS belongs to at least two classes of **NP**-hard problems: the Quadratically Constrained Quadratic Problem (QCQP) [25] and the Circle Packing Problem (CPP), where one is given a sequence of n radii $r_1, \ldots, r_n$ and must decide whether n circles with respective radii can fit in a unit square; the CPP was recently shown to be **NP**-hard [26]. The proof employs different radii and therefore does not seem applicable to PECS. Because the YES certificates of PECS instances might involve irrational numbers, it is unclear whether PECS is in **NP**.

One might wonder whether considering the contact structures of the circles on the plane might yield a more treatable problem to reduce to, but this line of thought does not seem promising either: let $(r_1, \ldots, r_n)$ be a YES instance of the CPP, and $\mathcal{C} = ((x_i, y_i) \mid i \leq n)$ be a certificate (i.e., the sequence of circle centers). The *coin graph* of $\mathcal{C}$ is an undirected graph $G = (V, E)$ such that $V = \{1, \ldots, n\}$ and for all $u, v \in V$ we have $\{u, v\} \in E$ if $\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2} = r_u + r_v$. It is known that a graph is a coin graph if and only if it is finite, simple

and planar [27]. Determining whether a given graph is a coin graph with unit edge lengths is **NP**-hard [28, 29], but this does not take into account the PECS constraint that all circles should be contained in a square; furthermore, the instance for the PECS is simply a pair of numbers rather than a whole graph.

Many papers simply declare circle packing problems to be **NP**-hard (sometimes without stating any reference). As an example, [21] presents a heuristic for packing equal circles in an equilateral triangles: the authors state that the problem is **NP**-hard and refer to [30, 31, 32]. The authors of [30] state in their introduction that:

> *For larger combinatorial [packing] problems these [simple] techniques become inefficient due to the vast number of possible solutions and the computation time grows exponentially. These problems are said to be* **NP***-complete,*

a definitely questionable definition of **NP**-completeness; in the conclusion they also mention that "most packing problems are **NP**-complete". Garey and Johnson [31] only discuss set and bin packing problems, but not circle packing in the plane. The authors of [32] present polynomial-time approximation schemes for square covering, disc covering and square packing in a rectilinear region, but not disc packing; they cite [33, 34] for **NP**-completeness of the square packing problem. The authors of [33] exhibit a proof that packing equal boxes in a given region R of the plane is **NP**-complete (and they say that the proof can be extended to the case of equal discs). However, they work under the hypothesis that in R there is only a finite number of box (disc) positions which might be required by an optimal packing. More precisely, they consider the graph $\mathcal{R}$ whose vertex set is R and whose edge set includes pairs of points in R which are closer than $2r$, so that equal disc packings then correspond to stable sets in $\mathcal{R}$; but they assume $\mathcal{R}$ to be finite, which does not seem to be the case if R is the unit square as in the PECS. In his **NP**-completeness column [34], Johnson reports the results of [33] as packing equal squares in a rectilinear polygon such that the squares are parallel to the axes, but omits to mention the disc packing result.

In summary, to the best of our knowledge, there is no proof in the literature that offers a polynomial reduction from an **NP**-hard problem to PECS.

1.3. Contents and contributions

In Sect. 2 we introduce two PECS formulations, the BB algorithm we use for solving the PECS, and justify the need for breaking formulation symmetries. In Sect. 3 we introduce some formulation symmetry concepts, determine the structure of the PECS formulation group, and describe some classes of symmetry-breaking constraints. In Sect. 4 we discuss the impact of our symmetry-breaking reformulation on the local NLP subsolver in the BB method. Sect. 5 discusses computational results.

The heart of our mathematical contribution is in Sect. 3. We also regard Sect. 4 as an important contribution: for the first time a symmetry-breaking

reformulation is shown to have an impact on a local NLP solver (usually symmetry breaking techniques help tighten a relaxation bound [35, 36, 37]). Our computational results do not improve the state of the art [4] *if we run the BB method to its completion*. We are nonetheless able to emphasize the striking root-node performance of our symmetry-breaking reformulation, as shown by column r_{r} of Table 4.

2. The PECS formulation

We employ the following Mathematical Programming (MP) formulation for the optimization version of the PECS:

$$\max r \tag{1}$$
$$\forall i < j \leq n \quad (x_i - x_j)^2 + (y_i - y_j)^2 \geq 4r^2 \tag{2}$$
$$\forall i \leq n \quad x_i, y_i \in [r, 1-r] \tag{3}$$
$$r \geq 0. \tag{4}$$

The objective function (1) aims to maximize the radius r; the distance constraints (2) make sure the circle interiors are pairwise disjoint; the bound constraints (3) make sure the circles are within the square.

The PECS formulation given above is a quadratically constrained Nonlinear Programming (NLP) problem. The only nonconvexities are given by the reverse convex constraints (2). A simple multi-start approach, where a local NLP solver (such as SNOPT [38]) is deployed from a variety of randomly chosen starting points, will quickly convince the reader that the PECS formulation has several different local optima. The most widespread method for solving nonconvex NLPs is the spatial Branch-and-Bound (sBB) algorithm.

2.1. Equivalent formulations and choice thereof

As remarked in [15], the PECS is equivalent to the following problem:

> Point Packing in a Square (PPS). Given an integer $n > 0$ and a rational $\alpha > 0$, can n points be determined in the unit square in such a way that their squared minimum pairwise distance is greater or equal to α?

A well-known mathematical programming formulation for the optimization version of the PPS is the following:

$$\max \alpha \tag{5}$$
$$\forall i < j \leq n \quad (x_i - x_j)^2 + (y_i - y_j)^2 \geq \alpha \tag{6}$$
$$\forall i \leq n \quad x_i, y_i \in [0, 1] \tag{7}$$
$$\alpha > 0, \tag{8}$$

where $\alpha = 4r^2$. Here is a reduction from PPS to PECS: (a) every NO instance of the PPS is a NO instance of the PECS; (b) if a YES instance of the PPS is such

that $r \geq \frac{\sqrt{\alpha}}{2+2\sqrt{\alpha}}$ then it is also a YES instance of the PECS (the inequality can be verified easily by scaling the PPS configuration down so that it allows enough space to arrange circles wholly contained within the square); (c) otherwise, it is a NO instance of the PECS (Ch. 2 in [4]). Thus, given an instance of the PPS with its YES/NO decision, a YES/NO decision can be taken in constant time for the PECS. A similar transformation from PECS to PPS also holds.

Although the two problems are equivalent, the corresponding formulations are not. Specifically, the PECS formulation involves both r and r^2, whereas the PPS formulation only involves a linear term α which replaces $4r^2$ (given an optimal α, the corresponding r can be recovered in constant time). This formulation difference has an impact on sBB performance with implementations such as Couenne [39]: Table 1 shows that there is no clear efficiency domination on a per-instance basis. The time to solve a node is lower for PPS, but the total number of nodes is greater; however, the impact of the number of nodes is the most relevant for big instances. In the rest of the paper, we shall employ the

	PECS		PPS	
n	CPU	nodes	CPU	nodes
2	0.03	0	0.04	0
3	0.06	0	0.07	0
4	0.12	0	0.10	0
5	0.19	2	0.20	2
6	14.30	94	3.18	220
7	17.11	614	9.77	2360
8	57.25	6952	41.94	9160
9	553.62	69172	1334.82	339804

Table 1: Comparing sBB on PECS and on PPS.

PECS formulation (1)-(4).

2.2. The spatial Branch-and-Bound algorithm

The sBB algorithm is an ε-approximation algorithm for solving nonconvex NLPs and Mixed-Integer Nonlinear Programs (MINLP); several variants exist, among which [40, 41, 42, 43, 44, 39]. Given a constant $\varepsilon > 0$, the sBB recursively generates a binary search tree, some leaf node of which contains a feasible point (x^*, y^*, r^*) for which r^* differs by at most ε from the globally optimal value of r. A generic node a of the sBB tree contains a formulation restricted to some box $B^a \subsetneq [0,1]^{2n+1}$ as well as an upper bound value $\bar{r}_a$ relative to the parent node. All along the sBB run, the following data are maintained:

- the search tree, encoded in some efficiently accessible form;
- the best solution (x^*, y^*, r^*) so far (also called the *incumbent*).

The following steps are performed at each node a.

1. *Range tightening* techniques such as Optimization-Based Bounds Tightening (OBBT) [44] and Feasibility-Based Bounds Tightening (FBBT) [45] in order to attempt to reduce the width of B^a in view to obtain a tighter upper bound.

2. *Computation of an upper bound* $\bar{r}_a$ (given by the solution $(\bar{x}_a, \bar{y}_a)$) for the restriction to B^a of the PECS formulation. This is done by means of solving a linear relaxation thereof (see below).

3. *Pruning by bound*: if $\bar{r}_a \leq r^*$ then the box B^a cannot contain optima better than the incumbent. Go to Step 8.

4. *Computation of a lower bounding solution* (x', y', r'), obtained using a local NLP solver on the problem at the node, with $(\bar{x}_a, \bar{y}_a, \bar{r}_a)$ as a starting point.

5. *Incumbent evaluation*: if $r' > r^*$ then let $(x^*, y^*, r^*) \leftarrow (x', y', r')$.

6. *Pruning by optimality*: if $\bar{r}_a - r' < \varepsilon$, then x' is an ε-approximate global optimum within the box B^a; further refinements will not yield better optima. Go to Step 8.

7. *Branching*. Select a variable and a value for branching: this consists in creating two subnodes a_1, a_2 of a, one with the subproblem where the branching variable is constrained between its lower range end and the branching value, and the other between the branching value and its upper range end; several heuristics exist for selecting branching variable and value [39].

8. *Choice of next node*: again, several heuristic methods exist. The most popular seems to be the choice of the node with the highest associated upper bound, insofar as it intuitively offers the best promise of improving the incumbent.

A proof of finite convergence of the sBB to an ε-approximation of a global optimum is given in [46].

2.2.1. Linear relaxation of the PECS formulation

The linear relaxation employed by most sBB solvers is constructed automatically from the problem formulation in the following way:

- replace all nonlinear terms $T(x, y)$ by an added variable w_T;
- compute lower and upper linear bounding functions $\check{T}(x, y), \hat{T}(x, y)$ to $T(x, y)$ on the node box B^a;
- adjoin constraints $\check{T}(x, y) \leq w_T \leq \hat{T}(x, y)$ to the formulation.

The distance constraints (2) are the only ones that need to be relaxed, as they are the only nonconvex ones. The relaxation we obtain at the root node (where $B = [0,1]^{2n+1}$) is:

$$\begin{aligned}
& & \max \quad & r & & (9)\\
\forall i < j \leq n \quad & (X_i + X_j - 2W_{ij}) & + \quad & (Y_i + Y_j - 2Z_{ij}) \geq 4R & & (10)\\
\forall i \leq n \quad & x_i, y_i & \in \quad & [r, 1-r] & & (11)\\
& r & \geq \quad & 0 & & (12)\\
\forall i \leq n \quad & X_i & \in \quad & [0, x_i] & & (13)\\
\forall i \leq n \quad & Y_i & \in \quad & [0, y_i] & & (14)\\
\forall i < j \leq n \quad & W_{ij} & \leq \quad & \min\{x_i, x_j\} & & (15)\\
\forall i < j \leq n \quad & W_{ij} & \geq \quad & \max\{0, x_i + x_j - 1\} & & (16)\\
\forall i < j \leq n \quad & Z_{ij} & \leq \quad & \min\{y_i, y_j\} & & (17)\\
\forall i < j \leq n \quad & Z_{ij} & \geq \quad & \max\{0, y_i + y_j - 1\} & & (18)\\
& R & \in \quad & [0, r], & & (19)
\end{aligned}$$

where, for each $i \leq n$, $X_i \in [0, x_i]$ are lower/upper bounding relaxations of $X_i = x_i^2$ on $x_i \in [0,1]$ (the same holds for Y_i and R), and for all $i < j \leq n$ constraints (15)-(16) are lower and upper bounding relaxations for $x_i x_j$ on $[0,1] \times [0,1]$ (the same holds for constraints (17)-(18)).

Proposition 1. *All optimal solutions of the PECS relaxation* (9)-(18) *have* $r = x = y = \frac{1}{2}$.

Proof. First, $r = \frac{1}{2}$ is the globally maximal value of the PECS relaxation, as any larger value would make (11) infeasible. Secondly, by (11), $r = \frac{1}{2}$ implies $x_i = y_i = \frac{1}{2}$ for all $i \leq n$. Any value of W, Z, R in $[0, \frac{1}{2}]$ consistent with (10) (e.g. $W = Z = R = 0$) yields a feasible solution with maximum objective function value. □

Although the situation changes at lower level nodes, relaxations yielding $x_i = y_i$ for several values of i is typical for several high-level nodes.

2.3. Motivations for exploiting symmetry

The PECS has solution symmetries that stem from the geometry of the configurations (rotations and reflections of the square), as well as from the formulation itself (permutations of axes labels and point indices). The sBB tree is a rooted plane binary tree whose leaves contain globally optimal solutions (or rather, ε-approximations thereof). Intuitively, a formulation with fewer optimal solutions yield fewer leaves, smaller sBB trees and faster convergence. If a set of different global optima can be obtained by symmetry from just one global optimum, we should aim to only keep one sBB branch leading to a single optimum, whilst discarding the other (symmetric) branches. One way to do this — the way we shall follow in this paper — consists in reformulating the PECS so that some symmetric solutions become infeasible. In other words, we

adjoin some constraints to the formulation which are feasible with at least one global optimum, but might make several symmetric optima infeasible. Such constraints are called *Symmetry Breaking Constraints* (SBC) [36] (also called *Static Symmetry Breaking Inequalities* (SSBI) [35, 2]), and the corresponding reformulation is called a *narrowing* [47]. The sBB applied to the proposed narrowing tightens the bound in Step 2 more effectively and thus solves the problem in less CPU time. Experiments indicate that the bound tightening occurs relatively late in the search [36], which means that letting sBB terminate naturally is still practically too CPU-expensive.

An important motivation for this work is based on the empirical observation that good PECS solutions are found earlier when using an SBC-based narrowing. Since the incumbent is found by the local NLP solver in Step 4, this means that the narrowing somehow "eases" local ascent towards good optima. Consider the PECS with $n = 2$: since the root node relaxation solution has all components set to $\frac{1}{2}$ by Prop. 1, at Step 4 the local NLP solver will use the central point of the square as a starting point to perform local ascent from. Since there are four symmetric optima at exactly the same distance from the starting point, the local solution algorithm will have to consider four different ascent vectors (shown as the arrows in Fig. 1) whose sum is the zero vector, making it difficult to identify an ascent direction. Adjoining the SBC $x_1 \leq x_2$, for example, and assuming circle 1 is filled in Fig. 1, would make the two leftmost configurations infeasible. This will make the sum of the ascent vectors nonzero, thereby easing the task of the local NLP solver. The benefits brought by SBCs to local NLP

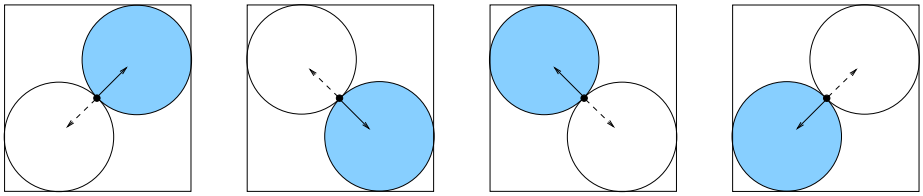

Figure 1: Four symmetric optima with $n = 2$: the sum of the four ascent directions from the central starting point towards the four optima is zero, both for solid and for dashed coordinates. If the two leftmost optima are infeasible (e.g. by means of the constraint $x_1 \leq x_2$) the sum of the ascent directions becomes nonzero: positive (for the dashed coordinates) and negative (for the solid coordinates).

solvers will be further discussed in Sect. 4.

3. Detection and exploitation of PECS symmetry

A method for automatically detecting formulation symmetries of MINLPs was described in [36] (a more compact explanation was provided in [1]). It basically consists in encoding the MINLP instance as a Directed Acyclic Graph (DAG) and in finding the graph automorphisms group of this DAG. The group generators can then be "projected" on the set of variable indices, thus obtaining

a set of generators for the group G_P of variable permutations which keep the formulation of the MINLP P invariant.

The following group structures were (automatically) obtained for the PECS formulation:

n	G_{PECS}
2	$C_2 \times S_2$
3	$C_2 \times S_3$
4	$C_2 \times S_4$
5	$C_2 \times S_5$

(the experiments were conducted on many more instances than are listed here). This allowed us to conjecture that the group of the PECS formulation is $C_2 \times S_n$. Intuitively, this is reasonable: C_2 corresponds to permuting the symbols x with the symbols y, and S_n corresponds to permuting the variable indices. The hardest part of proving the conjecture consists in showing that there are no other formulation symmetries for a generic n. Let $\mathcal{G}(\mathrm{P})$ be the set of global optima of problem P.

Theorem 1. *The formulation group of the PECS is isomorphic to $C_2 \times S_n$.*

Proof. Let G_{PECS} be the formulation group of PECS. For all $i < j \leq n$ call the constraints $(x_i - x_j)^2 + (y_i - y_j)^2 \geq 4r^2$ the *distance* constraints (2). Let $(x, y, r) \in \mathcal{G}(\mathrm{PECS})$; the following claims are easy to establish.

1. The permutation $\tau = \prod_{i \leq n}(x_i, y_i)$ is in G_{PECS}; ($\langle\tau\rangle \cong C_2$).
2. For any $i \leq n-1$, the permutation $\sigma_i = (x_i, x_{i+1})(y_i, y_{i+1})$ is in G_{PECS}; notice that $\langle\sigma_i \mid i < n\rangle \cong S_n$.
3. Any permutation moving r to one of the variables $\notin G_{\mathrm{PECS}}$.
4. If $\pi \in G_{\mathrm{PECS}}$ such that $\pi(x_i) = y_i$ for some $i \leq n$ then $\pi(x_i) = y_i$ for all $i \leq n$, as otherwise the term $x_i x_j + y_i y_j$ (appearing in the distance constraints) would be mapped to a term not appearing in the problem.
5. For any $i < n$, if $\pi \in G_{\mathrm{PECS}}$ such that $\pi(x_i) = x_{i+1}$ *or* $\pi(y_i) = y_{i+1}$, then $\pi(x_i) = x_{i+1}$ *and* $\pi(y_i) = y_{i+1}$; if not the term $x_i x_{i+1} + y_i y_{i+1}$ (appearing in some of the distance constraints) would be mapped to a term not appearing in the problem.

Let $K = \langle\tau\rangle$ and $H_n = \langle\sigma_i \mid i \leq n-1\rangle$. Claims (1)-(2) imply that $K, H_n \leq G_{\mathrm{PECS}}$. It is tedious but not too hard to check that $KH_n = H_nK$; it follows that $KH_n \leq G_{\mathrm{PECS}}$ and hence K, H_n are normal subgroups of KH_n. Since $K \cap H_n = \{e\}$, we have $KH_n \cong K \times H_n \cong C_2 \times S_n \leq G_{\mathrm{PECS}}$.

Now suppose $\pi \in G_{\mathrm{PECS}}$ with $\pi \neq e$. By Claim (3), π cannot move r so it must map x_i to y_j for some $i < j \leq n$; the action $i \to j$ on the circles indices can be decomposed into a product of transpositions $i \to i+1, \ldots, j-1 \to j$. Thus, by Claim (5) (resp. 4), π involves a certain product γ of τ and σ_i's; furthermore,

since by definition γ maps x_i to y_j, any permutation in G_{PECS} (including π) can be obtained as a product of these elements γ; hence π is an element of KH_n, which shows $G_{\mathrm{PECS}} \leq KH_n$, implying $G_{\mathrm{PECS}} \cong C_2 \times S_n$. □

3.1. Breaking symmetries

Once G_{PECS} is known, we aim to find a narrowing Q which ensures that at least one symmetric optimum of PECS is in $\mathcal{G}(Q)$. An SBC for a problem P with respect to a permutation $\pi \in G_P$ is a system of constraints $h(x) \leq 0$ such that there is a global optimum $y \in \mathcal{G}(P)$ with $h(\pi y) \leq 0$. This definition simply ensures that a narrowing does not make all global optima of P infeasible, but does not attempt to quantify the extent of the symmetry breaking; definitions in this sense, but limited to Integer Linear Programs (ILP) can be found in [48]. Adjoining SBCs to P yields a narrowing of P [36].

We present three different narrowings of the PECS obtained by using three different classes of SBCs.

1. Weak SBCs:
$$\forall i \leq n \quad x_1 \leq x_i. \tag{20}$$
These SBCs are based on the fact that we can always choose a an arbitrary index (e.g., 1) such that the circle corresponding to that index is leftmost. One might alternatively choose to employ $\forall i \leq n\ y_1 \leq y_i$.

2. Strong SBCs:
$$\forall i \in \{2, \ldots, n\} \quad x_{i-1} \leq x_i. \tag{21}$$
These SBCs are based on the fact that the circles can be ordered on the horizontal axis. Again, one might alternatively choose to employ $\forall i \in \{2, \ldots, n\}\ y_{i-1} \leq y_i$.

3. Mixed SBCs: designed to constrain both sets of coordinates at the same time. These are discussed in more depth in Sect. 3.2.

3.2. Mixed SBCs

In the mixed SBCs, we remove some of the strong SBCs in x and replace them with compatible SBCs in y. More precisely, let $L \in \{1, \ldots, \lfloor \frac{n}{2} \rfloor\}$, and consider the strong SBCs. For each $i \in \left\{1, 2, \ldots, \lceil \frac{n}{L} \rceil - 1\right\}$ we replace the constraints $x_{iL} \leq x_{iL+1}$ with $y_{1+(i-1)L} \leq y_{1+iL}$.

In order to show that the mixed constraints are SBCs, we prove that the PECS formulation with the mixed constraints adjoined is a narrowing of the PECS formulation. We define the following index sets:

- $\mathcal{N} = \{1, \ldots, n\}$
- $\mathcal{N}' = \{1, \ldots, n-1\}$
- $\mathcal{N}'' = \{1, L+1, 2L+1, \ldots, (\lceil n/L \rceil - 2)L + 1\}$,

the following sets of constraints (intended as lists of symbolic expressions representing the constraints, rather than sets of real vectors feasible with the constraints):

- $\mathscr{S} = \{x_i \leq x_{i+1} \mid i \in \mathcal{N}'\}$
- $\forall i \in \mathcal{N}'' \quad \mathscr{A}_i = \{x_h \leq x_{h+1} \mid h \in \mathcal{N}' \setminus \{i+L-1\}\}$
- $\forall i \in \mathcal{N}'' \quad \mathscr{C}_i = \{y_i \leq y_{i+L}\}$,

and the following formulations:

- $\text{PECS}' \equiv \text{PECS} \cup \mathscr{S}$ (i.e., the PECS formulation with strong constraints)
- $\forall i \in \mathcal{N}'' \quad \text{PECS}_i \equiv \text{PECS} \cup \mathscr{A}_i \cup \mathscr{C}_i$,
- $\text{PECS}'' \equiv \text{PECS} \cup \bigcup_{i \in \mathcal{N}''} (\mathscr{A}_i \cup \mathscr{C}_i)$.

Proposition 2. *For all $i \in \mathcal{N}''$, PECS$_i$ is a narrowing of PECS.*

Proof. Let $i \in \mathcal{N}''$ and $(\bar{x}, \bar{y}, \bar{r}) \in \mathcal{G}(\text{PECS})$. For a permutation $\pi \in S_n$ we assume $\pi(\bar{x}, \bar{y}, \bar{r}) = (\pi\bar{x}, \pi\bar{y}, \bar{r})$ where π acts on a vector in $\mathbb{R}^n$ by permuting the indices of its components; notice that since π is simply a reindexing of the circles, $\pi(\bar{x}, \bar{y}, \bar{r}) \in \mathcal{G}(\text{PECS})$. Furthermore, since PECS$'$ is known to be a narrowing of PECS, we can assume WLOG that $(\bar{x}, \bar{y}, \bar{r})$ satisfies $\mathscr{S}$. If $\bar{y}_i \leq \bar{y}_{i+L}$ the result holds, otherwise assume $\bar{y}_i > \bar{y}_{i+L}$. Consider the permutation $\sigma_i = \prod_{\ell=0}^{L-1}(i+\ell, i+L+\ell)$ in S_n; $\sigma_i(\bar{x}, \bar{y}, \bar{r})$ has the following properties: (a) by the action of the 2-cycle $(i, i+L)$ (appearing in σ_i when $\ell = 0$) we have $\bar{y}_i < \bar{y}_{i+L}$; (b) $\forall \ell \in \{0, \ldots, L-2\}$ we have $\sigma_i \bar{x}_{i+\ell} = \bar{x}_{i+L+\ell} \leq \bar{x}_{i+L+\ell+1} = \sigma_i \bar{x}_{i+\ell+1}$ and $\sigma_i \bar{x}_{i+L+\ell} = \bar{x}_{i+\ell} \leq \bar{x}_{i+\ell+1} = \sigma_i \bar{x}_{i+L+\ell+1}$; (c) $\forall h \in \mathcal{N}'$ such that $h \notin H_i = \{i, \ldots, i+2L-1\}$ we have $\sigma_i \bar{x}_h = \bar{x}_h \leq \bar{x}_{h+1} = \sigma_i \bar{x}_{h+1}$ because σ_i fixes all $h \notin H_i$. Thus $\sigma_i(\bar{x}, \bar{y}, \bar{r}) \in \mathcal{G}(\text{PECS})$ and satisfies the constraints of PECS$_i$. □

Lemma 1. *Let $t = \lceil n/L \rceil - 1$ and $\Sigma = \{\sigma_i \mid i \in \mathcal{N}''\}$. Then $\langle \Sigma \rangle \cong S_t$.*

Proof. Notice $\mathcal{N}'' = \{(j-1)L + 1 \mid 1 \leq j \leq t\}$, and define a map $\varphi((j-1)/L + 1) = j$, under which $\varphi(\Sigma) = \{(1,2), (2,3), \ldots, (t-1, t)\}$. This map induces a group homomorphism $\bar{\varphi} : \langle \Sigma \rangle \to S_t$ given by $\bar{\varphi}(\sigma_i) = (\varphi(i), \varphi(i)+1)$, which can be verified to be injective and surjective. □

Similarly, for all $h < k \in \mathcal{N}''$ we have $\langle \Sigma^{hk} \rangle = \langle \{\sigma_i \mid h \leq i < k\} \rangle \cong \text{Sym}(I^{hk})$, the symmetric group on the set $I^{hk} = \{\varphi(h), \ldots, \varphi(k)\}$. Thus, for all $h, k \in \mathcal{N}''$, the permutation $\tau_{hk} = \prod_{\ell=0}^{L-1}(h+\ell, k+\ell)$ can be obtained as a certain product of the σ_i's for $i \in \varphi^{-1}(I^{hk})$. More precisely, we have $\tau_{hk} = (\varphi(k)-1, \varphi(k))(\varphi(k)-2, \varphi(k)-1) \cdots (\varphi(h), \varphi(h)+1)(\varphi(h)+1, \varphi(h)+2) \cdots (\varphi(k)-1, \varphi(k))$.

Theorem 2. *PECS$''$ is a narrowing of PECS.*

Proof. Let $(\bar{x}, \bar{y}, \bar{r}) \in \mathcal{G}(\text{PECS})$, and consider the set $\mathscr{V}$ of all constraints $\mathscr{C}_i \equiv \{y_i \leq y_{i+L}\}$ violated by $(\bar{x}, \bar{y}, \bar{r})$. Let ψ be the (invertible) map given by $\psi(\mathscr{C}_i) = (\varphi(i), \varphi(i)+1)$; then $\psi(\mathscr{V})$ is a set of transpositions that can be partitioned into maximal non-disjoint subsets $S^{hk} = \{(\varphi(h), \varphi(h)+1), \ldots, (\varphi(k)-1, \varphi(k))\}$; let $\mathscr{T}$ be the set of pairs (h,k) for which S^{hk} is in the partition of $\psi(\mathscr{V})$. It is easy to verify that if $\pi_{hk} = \prod_{\substack{\ell \in I^{hk} \\ h+\ell L < k-\ell L}} \tau_{h+\ell L, k-\ell L}$ then $\pi_{hk}\bar{y}$ satisfies the constraints in $\psi^{-1}(S^{hk})$. Furthermore, by maximality of the S^{hk}, the permutations π_{hk} are disjoint. Now, if $\pi = \prod_{(h,k)\in\mathscr{T}} \pi_{hk}$, $\pi(\bar{x}, \bar{y}, \bar{r})$ is such that $\pi\bar{y}$ satisfies all constraints in $\mathscr{V}$ and $\pi\bar{x}$ satisfies all constraints in $\bigcup_{i\in\mathcal{N}''} \mathscr{A}_i$ by Prop. 2. Thus $\pi(\bar{x}, \bar{y}, \bar{r}) \in \mathcal{G}(\text{PECS}'')$. □

4. Why SBCs are good for the local solver

As mentioned in Sect. 2.3, we partially motivate this work on the experimental observation that good feasible solutions were found earlier in the search with SBCs rather than without.

All our experiments are conducted using the Couenne [39] sBB solver, which employs the IpOpt [49] subsolver as the local NLP solver used to find incumbents in Step 4 of the sBB algorithm given in Sect. 2.2. IpOpt actually solves the following PECS reformulation:

$$
\begin{aligned}
& -\min -r && && (22)\\
\forall i < j \leq n \quad & (x_i - x_j)^2 + (y_i - y_j)^2 - 4r^2 - s_{ij} && = \quad 0 && (23)\\
\forall i \leq n \quad & x_i - r - L_i^x && = \quad 0 && (24)\\
\forall i \leq n \quad & y_i - r - L_i^y && = \quad 0 && (25)\\
\forall i \leq n \quad & x_i + r - 1 + U_i^x && = \quad 0 && (26)\\
\forall i \leq n \quad & y_i + r - 1 + U_i^y && = \quad 0 && (27)\\
& x, r, s, L, U && \geq \quad 0, && (28)
\end{aligned}
$$

obtained by introducing slack variables for each inequality. The natural starting point for solving (22)-(28) in Step 4 is the solution of the relaxation in Step 2, which is $x = y = r = \frac{1}{2}$ at the root node by Prop. 1. Since this is infeasible w.r.t. (23), IpOpt starts with a feasibility restoration phase, converging to the starting point $x = y = \frac{1}{2}$, $r = 0$. It is long and tedious, but easy, to check that linear independence constraints qualification (LICQ) conditions [50] hold at this starting point, and that it is actually a KKT point. Thus, IpOpt simply confirms it is a local optimum — this is consistent with the results in Table 2 (column r in "no SBCs").

If, on the other hand, we adjoin SBCs to the formulation, positive ascent directions are found using IpOpt's Second Order Corrections (SOC) [49], as shown by the locally optimal r values in column r ("strong SBCs") of Table 2. This is consistent with the intuitive explanation given in Sect. 2.3. Another interesting phenomenon occurs: the CPU time taken by IpOpt is reduced for the PECS with SBCs (Table 2, CPU columns). This is due to the fact that

	no SBCs		strong SBCs	
n	r	CPU	r	CPU
4	4.5e-5	1.9	0.25	0.07
5	4.5e-5	2.1	0.196	0.02
6	5e-5	0.05	0.187	0.04
7	5e-5	0.06	0.174	0.04
8	5e-5	0.05	0.169	0.06
9	5e-5	0.06	0.166	0.04
10	5e-5	0.06	0.148	0.06
20	4.95e-5	0.24	0.109	0.27
50	4.89e-5	48.91	0.068	4.82

Table 2: IPOPT with starting point $x = y = 0.5$, $r = 0$ with and without strong SBCs.

interior point methods require primal variables to be have strictly positive values at each iteration [49], and $r = 0$ obviously fails to satisfy this requirement. A different local NLP solver, `snopt`, which is based on a Sequential Quadratic Programming (SQP) method, converges to a local optimum in roughly the same CPU time both with and without SBCs, but fails to find ascent directions for r, because it is a first-order method and does not exploit SOC.

Although the above discussion only holds at the root node, further experiments with random variable bounds have shown that SBCs yield better values for r at lower nodes too (although the marked difference in CPU time disappears).

5. Computational results

The computational results reported in [1] show (empirically) that the strong SBCs provide a narrowing whose convex relaxation at nodes in the lower sBB tree levels is tighter with respect to the weak SBCs. In turn, the computational results reported in [2] show that the mixed SBCs provide a better narrowing than the strong SBCs. In the sequel, we are only going to consider PECS narrowings derived using mixed SBCs. All our computational results have been obtained on a 2.4GHz Intel Xeon CPU of a computer with 24 GB RAM running Linux and the COUENNE [39] solver (trunk version dated November 2010) with the default configuration.

5.1. Choice of L for the mixed SBCs

The techniques given in Sect 3.2 rely on an arbitrary choice for the integer L. Fig. 2 shows the number of sBB tree nodes in function of L for the instances from $n = 4$ to $n = 9$. These experiments indicate that $L = 2$ is the best choice.

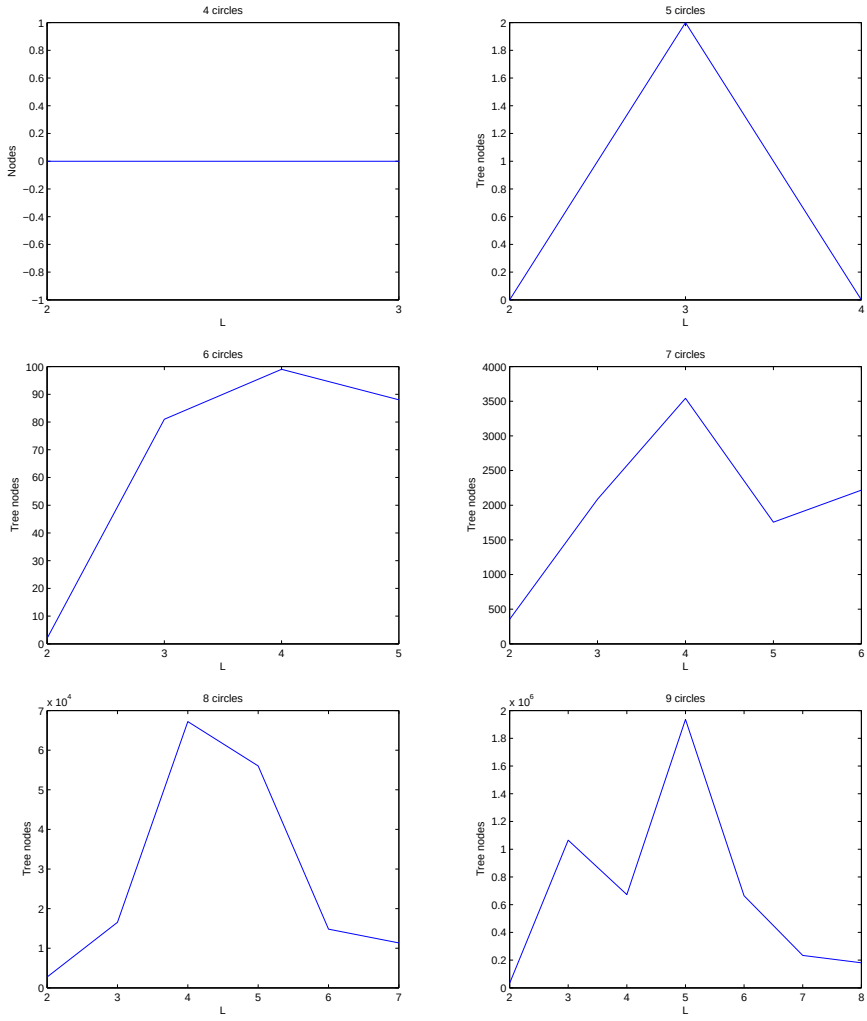

Figure 2: sBB tree nodes in function of L.

5.2. Effect of the narrowing on the upper bound

We first provide empirical evidence that the proposed SBCs tighten the upper bound in Step 2 of Sect. 2.2 by solving a set of small PECS instances to global optimality using Couenne. Table 3 reports the instance (n), the globally maximum possible radius r^* allowing a packing of n circles in the unit square, the number of sBB nodes and seconds of user CPU time taken by Couenne running to termination on the original formulation and on the narrowing.

n	r^*	Original formulation sBB nodes	Original formulation CPU time	Mixed SBC Narrowing sBB nodes	Mixed SBC Narrowing CPU time
2	0.292893	2	0.04	**0**	**0.02**
3	0.254333	2	0.15	**0**	**0.08**
4	0.25	282	1.85	**0**	**0.08**
5	0.207113	68710	69.24	**541**	**2.02**
6	0.187707	3087798	6176.05	**42850**	**90.84**

Table 3: sBB running to termination on small PECS instances.

5.3. Effect of the narrowing on the lower bound

We now exhibit the core of our results, i.e., the performance of COUENNE on the mixed SBC based narrowing with early termination based on two hours of user CPU time. In Table 4 we report the number of circles, the best known solution r^* (from `http://www.packomania.com`), the solution found at the root node r_{r}, the largest radius r' found by our method within the time limit, the tightest upper bound $\bar{r}$ on r' (which gives an idea of the optimality gap), the time $t(r')$ at which the solution r' was found and the number of nodes explored within the time limit.

n	r^*	r_{r}	r'	$\bar{r}$	$t(r')$	sBB nodes
20	0.111382	0.111382	0.111382	0.322063	16.45	441828
25	0.1	0.096852	0.1	0.250133	553.68	125632
30	0.091671	0.091671	0.091671	0.316273	86.24	90230
35	0.084290	0.082786	0.083766	0.351545	1495.31	46162
40	0.079186	0.078913	0.078913	0.2501	19.68	17116
45	0.074727	0.07444	0.07444	0.353325	357.90	12915
50	0.071377	0.070539	0.070539	0.250121	5429.88	2

Table 4: sBB running on large PECS instances.

Although we were not able to improve r for any tested PECS instance, our results show the validity of the proposed approach as a heuristic that finds excellent quality solutions already at the root node; since sBB always provides a relaxation bound, this heuristic also has the merit of yielding an approximation bound.

6. Conclusion

In this paper we discuss the use and impact of static symmetry breaking constraints in the problem of packing equal circles in a square. Our method exploits the formulation group of the problem in order to derive symmetry breaking constraints which both tighten the problem relaxation and — rather unexpectedly — eases the work of the local solver deployed at each node.

References

[1] A. Costa, P. Hansen, L. Liberti, Formulation symmetries in circle packing, in: R. Mahjoub (Ed.), Proceedings of the International Symposium on Combinatorial Optimization, Vol. 36 of Electronic Notes in Discrete Mathematics, Elsevier, Amsterdam, 2010, pp. 1303–1310.

[2] A. Costa, P. Hansen, L. Liberti, Static symmetry breaking in circle packing, in: U. Faigle (Ed.), Proceedings of the 9^{th} Cologne-Twente Workshop on Graphs and Combinatorial Optimization, University of Köln, 2010, pp. 47–50.

[3] K. Stephenson, Introduction To Circle Packing: The Theory of Discrete Analytic Functions, Cambridge University Press, Cambridge, 2005.

[4] P. Szabo, M. Markot, T. Csendes, E. Specht, L. Casado, I. Garcia, New Approaches to Circle Packing in a Square: With Program Codes, Springer, New York, 2007.

[5] M. Hifi, V. T. Paschos, V. Zissimopoulos, A simulated annealing approach for the circular cutting problem, European Journal of Operational Research 159 (2) (2004) 430 – 448.

[6] M. Hifi, R. M'Hallah, Approximate algorithms for constrained circular cutting problems, Computers & Operations Research 31 (5) (2004) 675 – 694.

[7] Y. Cui, Generating optimal t-shape cutting patterns for circular blanks, Computers & Operations Research 32 (1) (2005) 143 – 152.

[8] H. J. Fraser, J. A. George, Integrated container loading software for pulp and paper industry, European Journal of Operational Research 77 (3) (1994) 466 – 474.

[9] J. A. George, J. M. George, B. W. Lamar, Packing different-sized circles into a rectangular container, European Journal of Operational Research 84 (3) (1995) 693 – 712.

[10] I. Castillo, F. Kampas, J. Pintér, Solving circle packing problems by global optimization: Numerical results and industrial applications, European Journal of Operational Research 191 (2008) 786–802.

[11] H. Wenqi, T. Ye, Greedy vacancy search algorithm for packing equal circles in a square, Operations Research Letters 38 (2010) 378–382.

[12] P. Szabó, M. Markót, T. Csendes, Global optimization in geometry — Circle packing into the square, in: C. Audet, P. Hansen, G. Savard (Eds.), Essays and Surveys in Global Optimization, Springer, Berlin, 2005, pp. 233–265.

[13] U. Raber, Nonconvex all-quadratic global optimization problems: solution methods, application and related topics, Ph.D. thesis, University of Trier, Germany (1999).

[14] C. Maranas, C. Floudas, P. Pardalos, New results in the packing of equal circles in a square, Discrete Mathematics 142 (1-3) (1995) 287–293.

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

[16] R. Horst, N. V. Thoai, Dc programming: Overview, Journal of Optimization Theory and Applications 103 (1999) 1–43.

[17] K. J. Nurmela, P. R. J. Östergård, Packing up to 50 equal circles in a square, Discrete & Computational Geometry 18 (1) (1997) 111–120.

[18] R. L. Graham, B. D. Lubachevsky, Repeated patterns of dense packings of equal disks in a square, Electronic Journal of Combinatorics 3 (1) (1996) R16, 1–17.

[19] D. W. Boll, J. Donovan, R. L. Graham, B. D. Lubachevsky, Improving dense packings of equal disks in a square, Electronic Journal of Combinatorics 7 (1) (2000) R46, 1–9.

[20] L. Casado, I. Garcia, P. Szabó, T. Csendes, Equal circles packing in square ii: new results for up to 100 circles using the tamsass-pecs algorithm, in: F. Giannessi, P. Pardalos, T. Rapcsak (Eds.), Optimization Theory: Recent Developments from Mátraháza, Kluwer, Dordrecht, 2001, pp. 207–224.

[21] H. Wenqi, K. Yan, A heuristic quasi-physical strategy for solving disks packing problem, Simulation Modelling Practice and Theory 10 (2002) 195–207.

[22] B. Addis, M. Locatelli, F. Schoen, Disk packing in a square: a new global optimization approach, INFORMS Journal on Computing 20 (4) (2008) 516–524.

[23] A. Grosso, A. Jamali, M. Locatelli, F. Schoen, Solving the problem of packing equal and unequal circles in a circular container, Journal of Global Optimization 47 (2010) 63–81.

[24] M. Hifi, R. M'Hallah, A literature review on circle and sphere packing problems: Models and methodologies, Advances in Operations Research ID 150624 (2009) 1–22.

[25] S. Vavasis, Nonlinear Optimization: Complexity Issues, Oxford University Press, Oxford, 1991.

[26] E. Demaine, S. Fekete, R. Lang, Circle packing for origami design is hard, in: A. Peters (Ed.), Proceedings of the 5th International Conference on Origami in Science, Mathematics and Education, Singapore, to appear.

[27] H. Sachs, Coin graphs, polyhedra, and conformal maping, Discrete Mathematics 134 (1994) 133–138.

[28] P. Eades, N. Wormald, Fixed edge-length graph drawing is **np**-hard, Discrete Applied Mathematics 28 (1990) 111–134.

[29] H. Breu, D. Kirkpatrick, On the complexity of recognizing intersection and touching graphs of disks, in: F. Brandenburg (Ed.), Graph Drawing, Vol. 1027 of Lecture Notes in Computer Science, Springer, Berlin, 1996, pp. 88–98.

[30] E. Hopper, B. Turton, Application of genetic algorithms to packing problems - a review, in: Proceedings of the Second On-line World Conference of Soft Computing in Engineering Design and Manufacturing, Springer, Berlin, 1997, pp. 279–288.

[31] M. Garey, D. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, Freeman and Company, New York, 1979.

[32] D. Hochbaum, W. Maass, Approximation schemes for covering and packing problems in image processing and vlsi, Journal of the ACM 32 (1) (1985) 130–136.

[33] R. Fowler, M. Paterson, S. Tanimoto, Optimal packing and covering in the plane are **np**-complete, Information Processing Letters 12 (3) (1981) 133–137.

[34] D. Johnson, The **np**-completeness column: an ongoing guide, Journal of Algorithms 3 (1982) 182–195.

[35] F. Margot, Symmetry in integer linear programming, in: M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, L. Wolsey (Eds.), 50 Years of Integer Programming, Springer, Berlin, 2010, pp. 647–681.

[36] L. Liberti, Reformulations in mathematical programming: Automatic symmetry detection and exploitation, Mathematical Programming A (2010) 1–32.

[37] L. Liberti, Symmetry in mathematical programming, in: J. Lee, S. Leyffer (Eds.), Mixed Integer Nonlinear Programming, Vol. 154 of IMA, Springer, New York, 2012, pp. 263–286.

[38] P. Gill, User's guide for SNOPT version 7, Systems Optimization Laboratory, Stanford University, California (2006).

[39] P. Belotti, J. Lee, L. Liberti, F. Margot, A. Wächter, Branching and bounds tightening techniques for non-convex MINLP, Optimization Methods and Software 24 (4) (2009) 597–634.

[40] E. Smith, C. Pantelides, A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs, Computers & Chemical Engineering 23 (1999) 457–478.

[41] H. Tuy, Convex Analysis and Global Optimization, Kluwer Academic Publishers, Dordrecht, 1998.

[42] C. Floudas, Deterministic Global Optimization, Kluwer Academic Publishers, Dordrecht, 2000.

[43] C. Adjiman, S. Dallwig, C. Floudas, A. Neumaier, A global optimization method, αBB, for general twice-differentiable constrained NLPs: I. Theoretical advances, Computers & Chemical Engineering 22 (9) (1998) 1137–1158.

[44] L. Liberti, Writing global optimization software, in: L. Liberti, N. Maculan (Eds.), Global Optimization: from Theory to Implementation, Springer, Berlin, 2006, pp. 211–262.

[45] P. Belotti, S. Cafieri, J. Lee, L. Liberti, Feasibility-based bounds tightening via fixed points, in: D.-Z. Du, P. Pardalos, B. Thuraisingham (Eds.), Combinatorial Optimization and Applications, LNCS, Springer, New York, 2010, pp. 65–76.

[46] H. Tuy, D.c. optimization: Theory, methods and algorithms, in: R. Horst, P. Pardalos (Eds.), Handbook of Global Optimization, Vol. 1, Kluwer Academic Publishers, Dordrecht, 1995, pp. 149–216.

[47] L. Liberti, Reformulations in mathematical programming: Definitions and systematics, RAIRO-RO 43 (1) (2009) 55–86.

[48] L. Liberti, Automatic generation of symmetry-breaking constraints, in: B. Yang, D.-Z. Du, C. Wang (Eds.), COCOA Proceedings, Vol. 5165 of LNCS, Springer, Berlin, 2008, pp. 328–338.

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

[50] J. Nocedal, S. J. Wright, Numerical Optimization, Springer Series in Operations Research and Financial Engineering, Springer New York, 2006.