

## Rigorous packing of unit squares into a circle

Tiago Montanher · Arnold Neumaier ·  
Mihály Csaba Markót · Ferenc Domes ·  
Hermann Schichl

Received: date / Accepted: date

**Abstract** This paper considers the task of finding the smallest circle into which one can pack a fixed number of non-overlapping unit squares that are free to rotate. Due to the rotation angles, the packing of unit squares into a container is considerably harder to solve than their circle packing counterparts. Therefore, optimal arrangements were so far proved to be optimal only for one or two unit squares. By a computer-assisted method based on interval arithmetic techniques, we solve the case of 3 squares and find rigorous enclosures for every optimal arrangement of this problem. We model the relation between the squares and the circle as a constraint satisfaction problem (CSP) and found every box that may contain a solution inside a given upper bound of the radius. Due to symmetries in the search domain, general purpose interval methods are far too slow to solve the CSP directly. To overcome this difficulty, we split the problem into a set of subproblems by systematically adding constraints to the center of each square. Our proof requires the solution of 6, 43 and 12 subproblems with 1, 2 and 3 unit squares respectively. In principle, the method proposed in this paper generalizes to any number of squares.

**Keywords** Square packing into a circle · Interval branch-and-bound · Tiling constraints · Computer-assisted proof

**Mathematics Subject Classification (2000)** 52C15 · 90C26 · 65K05 · 65G30

---

This research was supported through the research grants P25648-N25 and P27891 of the Austrian Science Fund (FWF)

---

T. Montanher(✉)

E-mail: tiago.de.morais.montanher@univie.ac.at

A. Neumaier

E-mail: Arnold.Neumaier@univie.ac.at

M. C. Markót

E-mail: mihaly.markot@univie.ac.at

F. Domes

E-mail: ferenc.domes@univie.ac.at

H. Schichl

E-mail: hermann.schichl@univie.ac.at

Wolfgang Pauli Institute, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria and

Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

## 1 Introduction

Let  $S_1, \dots, S_n$  be  $n$  open unit squares and denote by  $C_r$  the closed circle of radius  $r$  centered at the origin. This paper deals with the problem of finding the smallest value of  $r$  such that one can pack  $S_1, \dots, S_n$  into  $C_r$  without overlapping. Formally, we can write the problem as

$$\begin{aligned} \min \quad & r \\ \text{s.t.} \quad & S_i \subseteq C_r \quad 1 \leq i \leq n \\ & S_i \cap S_j = \emptyset \quad 1 \leq i, j \leq n, \quad i \neq j. \end{aligned} \tag{1}$$

Packing identical objects into a container is an attractive part of geometrical optimization. The subject drew the attention of a considerable number of researchers, who contributed to problems similar to the one discussed in this paper.

The circle packing is the simplest packing problem in 2 dimensions in the sense that it does not involve the angles of the objects. Markót studied the packing of circles into a square from the interval analysis point of view in a series of papers [15, 16, 24]. In particular, he proved rigorous bounds for  $n = 28, 29$  and  $30$  circles. For a survey of the circle packing under the global optimization point of view, see [5]. The website *Packomania* [22] maintains an updated list of the best-known values for the packing of equal circles into several containers.

Kallrath [12] studied the packing of ellipses into rectangles using state-of-the-art complete global optimization solvers. He succeeded to find the global optimum for the case  $n = 3$  without rigor. For the packing of ellipsoids, see [2, 3].

Erdős and Graham [8] inaugurated the packing of unit squares into a square. They show that the wasted area in a container with side length  $l$  is  $O(l^{-\frac{7}{11}})$ . The proof relies on geometrical arguments and not on rigorous computations. Recent contributions in the packing of unit squares into a square include new bounds for the wasted area [6], the optimality proof for the cases  $n = 5, \dots, 10, 13$  and  $46$  [1, 10, 23] and the optimality proof for  $n - 2$  and  $n - 1$  whenever  $n$  is a square [19]. Again, none of these contributions rely on computer-assisted proofs. For a dynamic survey on the packing of unit squares in a square, see [10].

The packing of unit squares into general containers received considerably less attention than the circle or the unit square packing into a square. For example, Erich Friedman [9] maintains a list of proved and best-known values for the packing of unit squares into circles, triangles, L-shapes, and pentagons. In each case, only trivial arrangements are proved optimal. For the subject of interest in this paper, the packing of unit squares into a circle, the first open case is  $n = 3$ . For a list of figures of squares packed into a circle, see

<https://www2.stetson.edu/~efriedma/squincir/>.

### 1.1 Contribution and outline

This paper introduces a computer-assisted method for finding rigorous enclosures for  $r$  in Problem (1) and the corresponding optimal arrangements. The method is of theoretical interest since it proves optimality instead of only presenting a feasible arrangement. Therefore, it is suitable for small values of  $n$  only.

Our approach relies on the interval branch-and-bound framework. We implement the algorithm in *C++* using the forward-backward constraint propagation [21] to reduce the search domain. Section 2 introduces the solver. The code is available at

<http://www.mat.univie.ac.at/~montanhe/publications/n3.zip>.

Section 3 formulates Problem (1) as a constraint satisfaction problem (CSP). This paper uses the concept of sentinels [4, 18] to model non-overlapping conditions and the convexity of the circle to write containment constraints. Given an upper bound  $\bar{r}_n$  for  $r_n$ , the CSP asks for every feasible arrangement satisfying  $r \leq \bar{r}_n$ . Our software produces a list of small interval vectors with the property that every optimal arrangement of (1) belongs to at least one element in the list.

General purpose interval solvers are usually not capable of solving packing problems due to symmetries in the search domain. To overcome this difficulty, Section 4 shows how to split the original CSP into a set of subproblems by systematically adding constraints to the center of each square. We call them tiling constraints as the idea resembles the one proposed in [15, 16, 24]. The tiling divides the search domain into a set of isosceles triangles that must contain the center of at most one unit square. Then, one can replace the original CSP by a set of  $\binom{K}{n}$  subproblems, where  $K$  is the number of triangles in the tiling.

Our procedure iterates on the number of squares to avoid the exponential growth of subproblems. At the  $i$ -th iteration, we look at every possible combination of  $i$  triangles which can accommodate  $i$  unit squares into a circle with the radius at most  $\bar{r}_n$ . The rationale behind this strategy is twofold: (i) It allows us to discard a large number of hard subproblems by proving the infeasibility of more straightforward cases and (ii) It propagates the reduction on the search domain through the iterations. We also show that some combinations of triangles are symmetric by construction. Then one can discard them without any processing. This observation in addition to our iterative method reduces the number of hard cases considerably.

Section 5 illustrates the capabilities of our method. We find a mathematically rigorous enclosure for  $r_3$  and the corresponding optimal arrangement. If one set  $\bar{r} = \frac{5\sqrt{17}}{16}$  as pointed by Friedman [9], the tiling produces 36 triangles. Our approach requires the solution of 6 subproblems with one square, 43 with two and only 12 subproblems with 3 squares to conclude the proof. It is less than 1% of all possible  $\binom{36}{3} = 7140$  combinations. The method could also be used to find optimal configurations for higher values of  $n$  (e.g.,  $n = 4, 5, 6$ ).

## 1.2 Interval notation

This paper is an application of the interval branch-and-bound framework [11, 13]. We assume that the reader is familiar with concepts from interval analysis [20]. Let  $\underline{a}, \bar{a} \in \mathbb{R}$  with  $\underline{a} \leq \bar{a}$ . Then  $\mathbf{a} = [\underline{a}, \bar{a}]$  denotes the interval with  $\inf(\mathbf{a}) := \min(\mathbf{a}) := \underline{a}$  and  $\sup(\mathbf{a}) := \max(\mathbf{a}) := \bar{a}$ . We denote the width of the interval  $\mathbf{a}$  by  $\text{wid}(\mathbf{a}) := \bar{a} - \underline{a}$ .

The set of nonempty compact real intervals is given by

$$\mathbb{IR} := \{[\underline{a}, \bar{a}] \mid \underline{a} \leq \bar{a}, \underline{a}, \bar{a} \in \mathbb{R}\}.$$

Let  $S \subseteq \mathbb{R}$  be any set. Then the interval hull  $\square S$  of  $S$  is the smallest interval containing  $S$ .

An interval vector (also called box)  $\mathbf{x} := [\underline{x}, \bar{x}]$  is the Cartesian product of the closed real intervals  $\mathbf{x}_i := [\underline{x}_i, \bar{x}_i] \in \mathbb{IR}$ . We denote the set of all interval vectors of dimension  $n$  by  $\mathbb{IR}^n$ . We apply the width operator componentwise on vectors. Therefore  $\max(\text{wid}(\mathbf{x})) := \max(\text{wid}(\mathbf{x}_1), \dots, \text{wid}(\mathbf{x}_n))$ . Interval operations and functions are defined as in [13, 20]. The absolute value of the interval  $\mathbf{a}$  is given by

$$|\mathbf{a}| := \begin{cases} \mathbf{a} & \text{if } \inf(\mathbf{a}) \geq 0, \\ [0, \max(-\inf(\mathbf{a}), \sup(\mathbf{a}))] & \text{if } 0 \in \mathbf{a}, \\ -\mathbf{a} & \text{if } \sup(\mathbf{a}) \leq 0. \end{cases}$$

Let  $\mathbf{a}$  and  $\mathbf{b}$  be two intervals. The maximum of  $\mathbf{a}$  and  $\mathbf{b}$  is defined by

$$\max(\mathbf{a}, \mathbf{b}) := [\max(\inf(\mathbf{a}), \inf(\mathbf{b})), \max(\sup(\mathbf{a}), \sup(\mathbf{b}))].$$

Let  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$  be a function defined on  $\mathbf{x} \in \mathbb{IR}^n$  and let  $\mathbf{f} \in \mathbb{IR}^m$ . We denote the natural interval extension of the function  $F$  by  $\mathbf{F}$ . A constraint satisfaction problem (CSP) is the task of finding every point satisfying

$$F(x) \in \mathbf{f}, \quad x \in \mathbf{x}.$$

We call  $\mathbf{x}$  the search domain and the problem is said to be infeasible if there is no  $x \in \mathbf{x}$  satisfying  $F(x) \in \mathbf{f}$ . We also denote constraint satisfaction problems by the triplet  $(F, \mathbf{f}, \mathbf{x})$ .

## 2 The algorithm

This section describes the algorithm designed for solving the subproblems of form (1) using interval arithmetic [11, 13, 20]. The solver consists of two components, the memory, and the reducer. The former manages the branch-and-bound tree while the latter is responsible for processing the current box. There is also a post-processing step called cluster builder to group close boxes in the solution list.

The memory keeps the list of unprocessed boxes. It is also responsible for the box selector, and to split the boxes coming from the reducer that cannot be discarded or saved as a solution. In this paper, the selector is a depth-first search procedure while the splitter creates two boxes by dividing the input in the midpoint of the coordinate with maximum width.

The reducer contains a list of rigorous methods to reduce or discard boxes. This paper uses the forward-backward constraint propagation [21] and a feasibility verification method [7]. We consider a CSP of form  $(F, \mathbf{x}, \mathbf{f})$  in the next paragraphs to overview each method.

The forward-backward constraint propagation decomposes  $\mathbf{F}$  into a set of simple functions (like the exponential function or the sum of several elements) and displays the pieces in a graph. The forward step is a procedure to evaluate  $\mathbf{F}(\mathbf{x})$  systematically. In this case, the data flow from the decision variable nodes of the graph to the constraint nodes  $\mathbf{F}_1, \dots, \mathbf{F}_m$ . At the end of this step, each constraint node contains an enclosure of  $\mathbf{F}_i(\mathbf{x}) \cap \mathbf{f}_i$ . The backward step acts reversely. It starts from the constraint nodes  $\mathbf{F}(\mathbf{x}) \cap \mathbf{f}$  and walks the graph applying inverse functions until reaching  $\mathbf{x}_1, \dots, \mathbf{x}_n$ . At the end of the backward step, we have a new box  $\mathbf{x}' \subseteq \mathbf{x}$  with the reduced search domain.

This paper employs the following feasibility verification method. Let  $\mathbf{x}$  be a box and define the midpoint of  $\mathbf{x}$  as  $x^*$ . Then, we build a small box  $\mathbf{x}^*$  around the  $x^*$  and check its feasibility. The box  $\mathbf{x}^*$  is a feasible if  $\mathbf{F}(\mathbf{x}^*) \subseteq \mathbf{f}$ . We also save a box  $\mathbf{x}$  as solution if it satisfies  $\max(\text{wid}(\mathbf{x})) < \epsilon_x$  for a given  $\epsilon_x > 0$ .

The order into which we call the rigorous methods to process  $\mathbf{x}$  may influence the efficiency of the branch-and-bound procedure. In this paper, the methods follow the finite state machine described in Table 1.

Table 1: The finite state machine for the inner loop of the Algorithm 1.

Current state	Next state	Condition
Forward CP [21]	exit Backward CP	box $\mathbf{x}$ is infeasible otherwise
Backward CP [21]	Forward CP Feasibility verification	$G_{Rel}(\mathbf{x}, \mathbf{x}') > \epsilon_T$ otherwise
Feasibility Verification [7]	exit	true

The parameter  $\epsilon_T > 0$  is the threshold tolerance which controls the relative gain of the box  $\mathbf{x}' \subseteq \mathbf{x}$  with the help of the following function

$$G_{Rel}(\mathbf{x}, \mathbf{x}') := \max_{\substack{i=1, \dots, n; \\ \text{wid}(\mathbf{x}_i) > 0}} \left( \frac{\text{wid}(x'_i)}{\text{wid}(\mathbf{x}_i)} \right).$$

It is clear that the input of  $G_{Rel}$  at each iteration is the box  $\mathbf{x}$  and the outcome of the rigorous method,  $\mathbf{x}'$ .

After processing every box in the memory, we run a post-processing step to build clusters of solutions. This method supports the analysis of the solution list since it reduces the number of boxes on it. Given two intervals  $\mathbf{a}$  and  $\mathbf{b}$ , we define the gap between  $\mathbf{a}$  and  $\mathbf{b}$  by

$$\text{gap}(\mathbf{a}, \mathbf{b}) := \begin{cases} \inf(\mathbf{b}) - \sup(\mathbf{a}) & \text{if } \inf(\mathbf{b}) > \sup(\mathbf{a}), \\ \inf(\mathbf{a}) - \sup(\mathbf{b}) & \text{if } \inf(\mathbf{a}) > \sup(\mathbf{b}), \\ 0 & \text{if } \mathbf{a} \cap \mathbf{b} \neq \emptyset. \end{cases}$$

We save two boxes  $\mathbf{x}, \mathbf{y} \in \mathbb{I}\mathbb{R}^n$  in the same group if

$$\max_{i=1, \dots, n} \text{gap}(\mathbf{x}_i, \mathbf{y}_i) < \epsilon_C$$

where  $\epsilon_C$  is the cluster builder tolerance. After assignin a group to every box in the solution set, we return the interval hull of each group and conclude the procedure.

Algorithm 1 summarizes the interval branch-and-bound method. We implement the algorithm in *C++* using two interval arithmetic libraries, the *Filib* [14] and the *Moore* [17]. The user can choose any of these implementations in the verification of the proof. We report only results from the test with *Filib* in this paper. The supplementary material also reports the results utilizing *Moore*. They are consistent with each other.

**Algorithm 1** Simplified solver

---

**Input:** The CSP  $(F, \mathbf{f}, \mathbf{x})$ . The acceptance, threshold and cluster builder tolerances  $\epsilon_x > 0$ ,  $\epsilon_T > 0$  and  $\epsilon_C > 0$  respectively. The list  $M$  of rigorous methods to reduce the search space.

**Output:** The list  $\mathcal{L}$  of boxes satisfying the following property. If  $x \in \mathbf{x}$  is a feasible point of  $(F, \mathbf{f}, \mathbf{x})$  then there exists at least one box  $\mathbf{x}' \in \mathcal{L}$  such that  $x \in \mathbf{x}'$ .

- 1: Run the forward-backward constraint propagation procedure on  $(F, \mathbf{f}, \mathbf{x})$ . Save the reduced domain in  $\mathbf{x}'$ ;
- 2: **if**  $\mathbf{x}'$  is proved to be infeasible **then**
- 3:     Return  $\emptyset$ ;
- 4: **end if**
- 5: Start the list of boxes to be processed with  $\mathbf{x}'$ ;
- 6:  $\mathcal{L} \leftarrow \emptyset$ ;
- 7: **while** has boxes to process **do**
- 8:     Run the problem selector to obtain  $\mathbf{x}$ ;
- 9:      $i \leftarrow 1$ ;
- 10:    **while**  $i \leq |M|$  **do**
- 11:       Run the strategy  $M_i$  on  $(F, \mathbf{f}, \mathbf{x})$  to obtain the reduced box  $\mathbf{x}'$ ;
- 12:       **if**  $G_{Rel}(\mathbf{x}, \mathbf{x}') > \epsilon_T$  **then**
- 13:           $i \leftarrow 1$ ;
- 14:           $\mathbf{x} \leftarrow \mathbf{x}'$ ;
- 15:          continue;
- 16:       **end if**
- 17:        $i \leftarrow i + 1$ ;
- 18:        $\mathbf{x} \leftarrow \mathbf{x}'$ ;
- 19:    **end while**
- 20:    **if**  $\text{wid}(\mathbf{x}) \leq \epsilon_x$  **then**
- 21:        $\mathcal{L} \leftarrow \mathbf{x}$ ;
- 22:       continue;
- 23:    **end if**
- 24:    Run the splitter on  $\mathbf{x}$  and stack all subproblems in the memory;
- 25: **end while**
- 26: Run the cluster builder on  $\mathcal{L}$  with the cluster tolerance  $\epsilon_C$ ;
- 27: Return  $\mathcal{L}$ ;

---

### 3 The standard model

This section introduces the mathematical model for the containment and the non-overlapping conditions of (1). We call the resulting model the standard constraint satisfaction problem since it is the same for every subproblem. We assume that the squares have side length  $s$ . The inequalities for the containment condition follow from the convexity of the circle. On the other hand, non-overlapping constraints rely on the concept of sentinels [4, 18].

#### 3.1 Containment

Let  $C_r$  be the closed circle of radius  $r$  and centered at the origin. The convexity of the circle implies that  $c \in C_r$  for any point  $c$  in the segment of line  $\overline{ab}$  if  $a, b \in C_r$ . Then, a given square belongs to  $C_r$  if and only if its vertices belong to  $C_r$ .

Let  $S_{0,0}$  be the open square centered at the origin, with no rotation angle and side length  $s$ . Then

$$S_{0,0} := \left\{ x \in \mathbb{R}^2 \mid \max(|x_1|, |x_2|) - \frac{s}{2} < 0 \right\}.$$

We denote the closure of a set  $S$  by  $\overline{S}$ . The set of vertices of  $\overline{S}_{0,0}$  is given by

$$V_{0,0} := \{V^{NW}, V^{SW}, V^{NE}, V^{SE}\}$$

where

$$V^{NW} := \begin{pmatrix} -\frac{s}{2} \\ \frac{s}{2} \end{pmatrix}, V^{SW} := \begin{pmatrix} -\frac{s}{2} \\ -\frac{s}{2} \end{pmatrix}, V^{NE} := \begin{pmatrix} \frac{s}{2} \\ \frac{s}{2} \end{pmatrix}, V^{SE} := \begin{pmatrix} \frac{s}{2} \\ -\frac{s}{2} \end{pmatrix}.$$

For any  $c \in \mathbb{R}^2$  and  $\theta \in \mathbb{R}$ , we define the displacement operator as

$$h(c, \theta, x) := c + A_\theta x \quad (2)$$

where  $A_\theta$  is the rotation matrix

$$A_\theta := \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}.$$

The open square centered at  $c \in \mathbb{R}^2$ , with rotation angle  $\theta \in [0, \frac{\pi}{2})$  and side length  $s$  is the set given by

$$S_{c,\theta} := \{z \in \mathbb{R}^2 \mid z = h(c, \theta, x), x \in S_{0,0}\}. \quad (3)$$

The set of vertices of  $\overline{S}_{c,\theta}$ , denoted by  $V_{c,\theta}$ , is the union of the following points

$$V_{c,\theta}^P := c + A_\theta V^P, \quad P \in \{NW, SW, NE, SE\}.$$

Finally, we denote the circle of radius  $r$  and centered at the origin by  $C_r$ . Then

$$C_r := \{x \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq r^2\}.$$

**Proposition 1** *Let  $g_r(x) := x_1^2 + x_2^2 - r^2$  and consider the following inequalities*

$$g_r(V_{c,\theta}^P) \leq 0, \quad P \in \{NW, SW, NE, SE\} \quad (4)$$

Then

$$\overline{S}_{c,\theta} \subseteq C_r \quad \Leftrightarrow \quad (4) \text{ hold.}$$

*Proof* If  $\overline{S}_{c,\theta} \subseteq C_r$  then  $V_{c,\theta} \subseteq C_r$  and (4) hold. Conversely, since  $\overline{S}_{c,\theta}$  is a bounded polytope, it is given by the convex hull of the elements of  $V_{c,\theta}$ . The result follows from the convexity of the circle.  $\square$

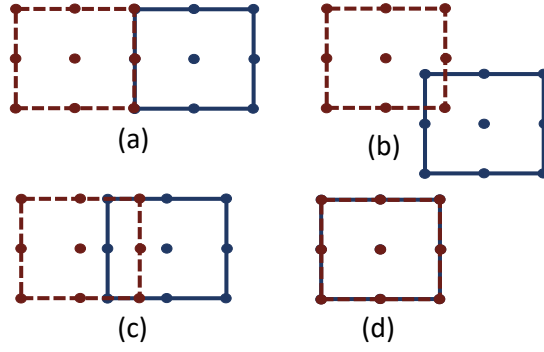


Fig. 1: (a) Non-overlapping squares. (b) Vertex sentinel violation. (c) Mid-point sentinel violation (d) Center sentinel violation.

### 3.2 Non-overlapping

This subsection shows that two squares  $S_{c_1, \theta_1}$  and  $S_{c_2, \theta_2}$  are non-overlapping if and only a set of nine points defined on  $S_{c_1, \theta_1}$  do not belong to  $S_{c_2, \theta_2}$  and vice-versa. We call such sets sentinels of a square. Figure 1 illustrates the need of the sentinels in the non-overlapping formulation.

The set of sentinels of  $S_{0,0}$  is given by

$$T_{0,0} := V_{0,0} \cup \{V^N, V^S, V^E, V^W, V^O\}$$

where

$$V^N := \begin{pmatrix} 0 \\ \frac{s}{2} \end{pmatrix}, V^S := \begin{pmatrix} 0 \\ -\frac{s}{2} \end{pmatrix}, V^E := \begin{pmatrix} \frac{s}{2} \\ 0 \end{pmatrix}, V^W := \begin{pmatrix} -\frac{s}{2} \\ 0 \end{pmatrix}, V^O := \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

We denote the set of sentinels of  $S_{c, \theta}$  by  $T_{c, \theta}$ . This set is given by the union of the following points

$$T_{c, \theta}^P := c + A_\theta V^P, \quad P \in \{NW, SW, NE, SE, N, S, E, W, O\}.$$

The next theorem states that the non-overlapping condition between two squares reduces to the containment verification of their sets of sentinels. It is a particular case of the sentinels theorem proved in [18].

**Theorem 1** *Let  $S_{c_i, \theta_i}$  and  $S_{c_j, \theta_j}$  be two squares defined by (3) and let  $T_{c_i, \theta_i}$  and  $T_{c_j, \theta_j}$  be their corresponding sets of sentinels. Then*

$$S_{c_i, \theta_i} \cap S_{c_j, \theta_j} = \emptyset \quad \Leftrightarrow \quad S_{c_i, \theta_i} \cap T_{c_j, \theta_j} = \emptyset \text{ and } S_{c_j, \theta_j} \cap T_{c_i, \theta_i} = \emptyset.$$

In order to check conditions of form  $S_{c_i, \theta_i} \cap T_{c_j, \theta_j} = \emptyset$  numerically, we need the definition of the inverse of the displacement operator (2)

$$h^{-1}(c, \theta, z) := A_\theta^T(z - c).$$



**Lemma 1** *Let  $z \in \mathbb{R}^2$  and  $S_{c,\theta}$  be a square defined by (3). Then*

$$z \in S_{c,\theta} \iff \max(|h_1^{-1}(c,\theta,z)|, |h_2^{-1}(c,\theta,z)|) - \frac{s}{2} < 0.$$

where  $h_1^{-1}$  and  $h_2^{-1}$  are the coordinates of the inverse operator.

*Proof* If  $z \in S_{c,\theta}$  then there exists  $x \in S_{0,0}$  such that  $x = h^{-1}(c,\theta,z)$  and the implication follows immediately. Conversely, let  $x := h^{-1}(c,\theta,z)$ . The left hand side of the equivalence implies that  $x \in S_{0,0}$ . If we let  $z' := c + A_\theta x$  then  $z' = c + A_\theta A_\theta^T(z - c) = z$ . Therefore  $z \in S_{c,\theta}$  and the result follows.  $\square$

Applying the inverse of the displacement operator of the square  $S_{c_i,\theta_i}$  to the point  $T_{c_j,\theta_j}^P \in T_{c_j,\theta_j}$  gives

$$h^{-1}(c_i,\theta_i,T_{c_j,\theta_j}^P) = A_{\theta_i}^T(c_j + A_{\theta_j}V^P - c_i), \quad V^P \in T_{0,0}. \quad (5)$$

Let  $c_{j,1}$  and  $c_{j,2}$  be the coordinates of the vector  $c_j$ . Then the coordinates of (5) are given by

$$\begin{aligned} u_1(c_i,c_j,\theta_i,\theta_j,V) &:= \cos(\theta_i)(c_{j,1} - c_{i,1}) - \sin(\theta_i)(c_{j,2} - c_{i,2}) + \\ &\quad + \cos(\theta_i - \theta_j)V_1 + \sin(\theta_i - \theta_j)V_2 \end{aligned}$$

and

$$\begin{aligned} u_2(c_i,c_j,\theta_i,\theta_j,V) &:= \sin(\theta_i)(c_{j,1} - c_{i,1}) + \cos(\theta_i)(c_{j,2} - c_{i,2}) + \\ &\quad - \cos(\theta_i - \theta_j)V_1 + \sin(\theta_i - \theta_j)V_2. \end{aligned}$$

The following proposition shows that the verification of  $S_{c_i,\theta_i} \cap T_{c_j,\theta_j} = \emptyset$  reduces to the evaluation of nine non-smooth functions.

**Proposition 2** *Let  $S_{c_i,\theta_i}$  and  $T_{c_j,\theta_j}$  be as in Theorem 1 and define the function*

$$u(c_i,c_j,\theta_i,\theta_j,V^P) := \max(|u_1(c_i,c_j,\theta_i,\theta_j,V^P)|, |u_2(c_i,c_j,\theta_i,\theta_j,V^P)|).$$

Then

$$S_{c_i,\theta_i} \cap T_{c_j,\theta_j} = \emptyset \iff u(c_i,c_j,\theta_i,\theta_j,V^P) - \frac{s}{2} \geq 0 \text{ for all } V^P \in T_{0,0}.$$

*Proof* Follows from the application of the Lemma 1 to the elements of  $T_{c_j,\theta_j}$ .  $\square$

### 3.3 The standard model

We conclude this section with the formal statement of the standard constraint satisfaction problem. Here and throughout we assume, without loss of generality, that the angle of the first square is always 0. This condition follows from the proper rotation of the remaining squares into the circle.

**Definition 1** [SCSP] Let  $\bar{r} > 0$  be an upper bound for the radius of the smallest circle into which one can pack  $n$  non-overlapping unit squares and  $s$  be a scaling factor. We denote the following problem by standard constraint satisfaction problem (SCSP)

$$\begin{aligned}
& \text{find} && (r, c_1, \theta_1, \dots, c_n, \theta_n) && (6) \\
& \text{s.t.} && u(c_i, c_j, \theta_i, \theta_j, V^P) - \frac{s}{2} \geq 0 \\
& && g_r(V_{c_i, \theta_i}^P) \leq 0 \\
& && c_{i,1}, c_{i,2} \in [-r, r] \\
& && \theta_i \in [0, \frac{\pi}{2}] \\
& && \theta_1 = 0 \\
& && r \leq \bar{r}
\end{aligned}$$

where  $i, j = 1, \dots, n$  with  $i \neq j$ ,  $V^P \in T_{0,0}$  and  $V_{c_i, \theta_i}^P \in V_{c_i, \theta_i}$ . Functions  $g_r$  and  $u$  are given by Propositions 1 and 2 respectively.

#### 4 Tiling

General purpose interval branch-and-bound procedures cannot solve the SCSP in a reasonable amount of time even for small values of  $n$  due to symmetries in the search space. This section introduces a tiling method to split (6) into a set of subproblems suitable for the Algorithm 1.

We employ the *Matlab*-like notation  $g := a : s : b$  to denote the array with  $k := \lfloor \frac{b-a}{s} \rfloor + 1$  elements where  $g_i := a + is$  for  $i = 0, \dots, k-1$ . In addition, we denote the array with the midpoints of  $g$  by  $g_c$ . Then,

$$g_{c,i} := \frac{g_i + g_{i+1}}{2}, \quad i = 0, \dots, k-2.$$

Let  $\bar{r} > 0$  be an upper bound for the SCSP. Then, the step length

$$l := \frac{2\bar{r}}{\lfloor 2\bar{r} + 1 \rfloor} \quad (7)$$

splits  $[-\bar{r}, \bar{r}]$  into  $\lfloor 2\bar{r} + 1 \rfloor$  equally divided intervals. Let  $V := \{v \in \mathbb{R} \mid v = -\bar{r} + il, i \in \mathbb{Z}\} \cap [-\bar{r}, \bar{r}]$  be the end points of each interval, satisfying  $v_i := -\bar{r} + il$  for  $i = 0, \dots, p := \lfloor 2\bar{r} + 1 \rfloor$ . Moreover, we write the midpoints of  $V$  as  $V_c$  where  $v_{c,i} := v_i + \frac{l}{2}$  for  $i = 0, \dots, p-1$ . Let  $\mathcal{V} := V \times V$  and  $\mathcal{C} := V_c \times V_c$ . We denote the elements of  $\mathcal{V}$  by  $v_{i,j} := \begin{pmatrix} v_i \\ v_j \end{pmatrix}$  for  $v_i, v_j \in v$  and  $0 \leq i, j \leq p$ . In the same way, we write the elements of  $\mathcal{C}$  as  $c_{i,j} := v_{i,j} + \begin{pmatrix} \frac{l}{2} \\ \frac{l}{2} \end{pmatrix}$  for  $v_{i,j} \in \mathcal{V}$  and  $0 \leq i, j \leq p-1$ . Algorithm 2 produces the sets  $\mathcal{V}$  and  $\mathcal{C}$ .

Let  $\triangle ABC$  be the triangle with vertices  $A, B, C \in \mathbb{R}^2$ . Then, we define the following triangles for  $0 \leq i, j \leq p-1$

$$\begin{aligned}
\triangle_{i,j}^T &:= \triangle v_{i,j+1} v_{i+1,j+1} c_{i,j}, \\
\triangle_{i,j}^L &:= \triangle v_{i,j+1} v_{i,j} c_{i,j}, \\
\triangle_{i,j}^D &:= \triangle v_{i,j} v_{i+1,j} c_{i,j}, \\
\triangle_{i,j}^R &:= \triangle v_{i+1,j} v_{i+1,j+1} c_{i,j}.
\end{aligned}$$

**Algorithm 2** Unscaled tiling**Input:** The upper bound  $\bar{r}$  for the radius in the SCSP.**Output:** The sets  $\mathcal{V}$  and  $\mathcal{C}$  containing the vertices of  $\Delta_{i,j}^o$  for  $0 \leq i, j \leq p-1$  and  $o \in \{T, L, D, R\}$ .

- 1:  $l \leftarrow \frac{2\bar{r}}{\lfloor 2\bar{r}+1 \rfloor}$ ;
- 2:  $v \leftarrow -\bar{r} : l : \bar{r}$ ;
- 3:  $\mathcal{V} \leftarrow v \times v$ ;
- 4:  $\mathcal{C} \leftarrow v_c \times v_c$ ;
- 5: return  $(\mathcal{V}, \mathcal{C})$ ;

Here,  $T, L, D$  and  $R$  stand for top, left, down and right respectively. Figure 2 shows that the definition aims to split the square with vertices  $v_{i,j}, v_{i+1,j}, v_{i+1,j+1}$  and  $v_{i,j+1}$  into four triangles. One can easily verify that the triangles can be written as

$$\begin{aligned} \Delta_{i,j}^T := \{x \in \mathbb{R}^2 \mid x_2 - x_1 \geq g_j - g_i, x_2 + x_1 \geq g_i + g_{j+1}, \\ x_1 \in [g_i, g_{i+1}], x_2 \in [g_j + \frac{l}{2}, g_{j+1}]\}, \end{aligned} \quad (8)$$

$$\begin{aligned} \Delta_{i,j}^L := \{x \in \mathbb{R}^2 \mid x_2 - x_1 \geq g_j - g_i, x_2 + x_1 \leq g_i + g_{j+1}, \\ x_1 \in [g_i, g_i + \frac{l}{2}], x_2 \in [g_j, g_{j+1}]\}, \end{aligned} \quad (9)$$

$$\begin{aligned} \Delta_{i,j}^D := \{x \in \mathbb{R}^2 \mid x_2 - x_1 \leq g_j - g_i, x_2 + x_1 \leq g_i + g_{j+1}, \\ x_1 \in [g_i, g_{i+1}], x_2 \in [g_j, g_j + \frac{l}{2}]\}, \end{aligned} \quad (10)$$

$$\begin{aligned} \Delta_{i,j}^R := \{x \in \mathbb{R}^2 \mid x_2 - x_1 \leq g_j - g_i, x_2 + x_1 \geq g_i + g_{j+1}, \\ x_1 \in [g_i + \frac{l}{2}, g_{i+1}], x_2 \in [g_j, g_{j+1}]\}. \end{aligned} \quad (11)$$

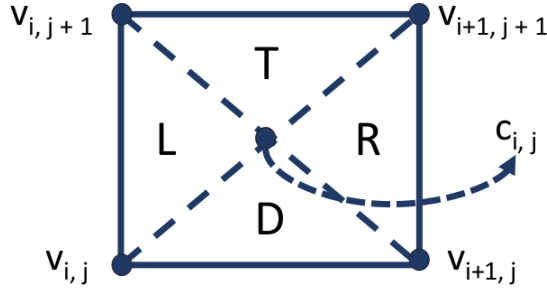


Fig. 2: The geometrical meaning of  $\Delta_{i,j}^o$  for  $0 \leq i, j \leq p-1$  and  $o \in \{T, L, D, R\}$ .

Lemma 2 is a collection of results needed in this section. In particular, Lemma 2-6 shows that the union of triangles  $\Delta_{i,j}^o$  for  $0 \leq i, j \leq p-1$  and  $o \in \{T, L, D, R\}$  tiles the search domain associated to the center variables in the SCSP.

**Lemma 2** Let  $l, p, v, v_c, \mathcal{V}, \mathcal{C}$  and  $\Delta_{i,j}^o$  be defined as above. Then,

1.  $l < 1$ .
2.  $x \in v \Rightarrow -x \in v$ .
3. If  $v_{i,j} \in \mathcal{V}$  then  $v_{i,j}^{90}, v_{i,j}^{180}, v_{i,j}^{270}, v_{i,j}^x, v_{i,j}^y, v_{i,j}^{Id}, v_{i,j}^{-Id} \in \mathcal{V}$  where

$$v_{i,j}^{90} := \begin{pmatrix} -v_j \\ v_i \end{pmatrix}, v_{i,j}^{180} := \begin{pmatrix} -v_i \\ -v_j \end{pmatrix}, v_{i,j}^{270} := \begin{pmatrix} v_j \\ -v_i \end{pmatrix},$$

$$v_{i,j}^x := \begin{pmatrix} v_i \\ -v_j \end{pmatrix}, v_{i,j}^y := \begin{pmatrix} -v_i \\ v_j \end{pmatrix}, v_{i,j}^{Id} := \begin{pmatrix} v_j \\ v_i \end{pmatrix}, v_{i,j}^{-Id} := \begin{pmatrix} -v_j \\ -v_i \end{pmatrix}.$$

4. If  $c_{i,j} \in \mathcal{C}$  then  $c_{i,j}^{90}, c_{i,j}^{180}, c_{i,j}^{270}, c_{i,j}^x, c_{i,j}^y, c_{i,j}^{Id}, c_{i,j}^{-Id} \in \mathcal{C}$  where the vectors are defined analogously as above.
5.  $\Delta_{i,j}^o$  is an isosceles triangle with base length  $l$  and legs with length  $\frac{l\sqrt{2}}{2}$  for  $0 \leq i, j \leq p-1$  and  $o \in \{T, L, D, R\}$ .
- 6.

$$[-\bar{r}, \bar{r}]^2 \equiv \bigcup_{\substack{0 \leq i, j \leq p \\ o \in \{T, L, D, R\}}} \Delta_{i,j}^o.$$

*Proof* 1. For  $a > 0$ , we have  $\lfloor a+1 \rfloor = a+1-\delta$  where  $\delta \in [0, 1)$  is the fractional part of  $a+1$ . Then  $1-\delta > 0$  and  $\lfloor a+1 \rfloor > a$ . The result follows by taking  $a = 2\bar{r}$ .

2. If  $x \in v$  then  $-x = \bar{r} - il$  for some  $i \in 0, \dots, p$ . Let  $y = -\bar{r} + jl$  and we need to verify if there exists some  $j \in 0, \dots, p$  such that  $y = -x$ . The equality holds by taking  $j = p - i$ .
3. If  $v_{i,j} \in \mathcal{V}$  then  $v_{j,i} \in \mathcal{V}$  and the result follows from the application of Lemma 2-2 of this proposition to each case.
4. The proof is similar to the case above.
5. For  $\Delta_{i,j}^T$ , we have  $\|v_{i,j+1} - v_{i+1,j+1}\| = l$  and

$$\|v_{i,j+1} - c_{i,j}\| = \|v_{i+1,j+1} - c_{i,j}\| = \frac{l\sqrt{2}}{2}.$$

The proof is similar for  $o \in \{L, D, R\}$ .

6. Let  $S_{i,j}$  be the closed square with vertices  $v_{i,j}, v_{i+1,j}, v_{i+1,j+1}, v_{i,j+1}$ . Since  $v_0 = -\bar{r}$  and  $v_p = \bar{r}$  it is clear that

$$[-\bar{r}, \bar{r}]^2 \equiv \bigcup_{0 \leq i, j \leq p-1} S_{i,j}.$$

The result follows by noting that

$$S_{i,j} \equiv \bigcup_{o \in \{T, L, D, R\}} \Delta_{i,j}^o.$$

□

We also assign a label to each triangle in the tiling. It helps us to easily identify a specific triangle during the proof of the case  $n = 3$  in Section 5. Triangles of form  $\Delta_{i,j}^T$  receive an index that is divisible by 4. In the same way, we assign labels to the left, down and right triangles with the congruence classes 1, 2 and 3 modulo 4, respectively. We denote the triangle with label  $i$  by  $T_i$ . Figure 3-Left shows the tiling for the best known upper bound of  $r_3$ .

We show now that each triangle of form  $\Delta_{i,j}^o$  contains the center of at most one unit square.

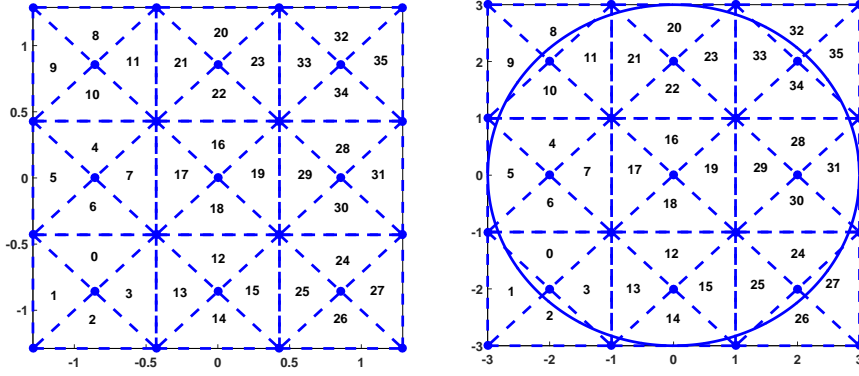


Fig. 3: Left: Tiling for the square  $[-\bar{r}_3, \bar{r}_3]^2$  where  $\bar{r}_3 = \frac{5\sqrt{17}}{16}$ . Right: Tiling for the square  $[-3, 3]^2$ .

**Lemma 3** *The minimal distance between the centers of two non-overlapping unit squares is 1.*

*Proof* Assume the contrary, let  $pq$  be a line segment of the centers with lower than 1. Let  $C_p$  and  $C_q$  the circles of radius  $\frac{1}{2}$  drawn into the squares. Then  $C_p$  and  $C_q$  intersect. But then since the squares are supersets of  $C_p$  and  $C_q$ , respectively, they also intersect. A contradiction.  $\square$

**Proposition 3** *Let  $\Delta_{i,j}^o$  for  $0 \leq i, j \leq p-1$  and  $o \in \{T, L, D, R\}$  be as defined above. If  $S_{c_1, \theta_1}$  and  $S_{c_2, \theta_2}$  are two unit squares such that  $c_1, c_2 \in \Delta_{i,j}^o$  then  $S_{c_1, \theta_1} \cap S_{c_2, \theta_2} \neq \emptyset$ .*

*Proof* Lemma 2-1 shows that  $l < 1$  and Lemma 2-5 gives that the base length of  $\Delta_{i,j}^o$  is  $l$  while its legs have length  $\frac{l\sqrt{2}}{2}$ . The result follows from Lemma 3.  $\square$

Let  $K := 4p^2$  be the number of triangles in the tiling. Proposition 3 states that we can split the SCSP into a set of  $\binom{K}{n}$  subproblems. In each subproblem, we enforce that the center of each square belongs to a given triangle. For example, one can define the subproblem  $T_0T_{19}T_{33}$  in the same tiling displayed in Figure 3-Left. In this case, we set the standard constraint satisfaction problem defined in (1) and add to the model the linear inequalities given by Equations (8)-(11) for  $\Delta_{0,0}^T$ ,  $\Delta_{1,1}^R$  and  $\Delta_{2,2}^L$  respectively.

We conclude this subsection by proving that several subproblems can be discarded without any processing due to symmetries in  $\mathcal{V}$  and  $\mathcal{C}$ . Let  $f^{90}, f^{180}, f^{270} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  be the linear mappings that rotate the vector  $x \in \mathbb{R}^2$  by an angle of 90, 180 and 270 degrees respectively. In the same way, define the linear mappings  $f^x, f^y, f^{Id}, f^{-Id} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  as the reflections around the lines  $x = 0$ ,  $y = 0$ ,  $y = x$  and  $y = -x$  respectively.

**Proposition 4** *Let  $\Delta_{i,j}^o$  for  $0 \leq i, j \leq p-1$  and  $o \in \{T, L, D, R\}$  be a triangle of form (8)-(11). Then,  $f^{op}(\Delta_{i,j}^o)$  for  $op \in \{90, 180, 270, r, x, Id, -Id\}$  is a triangle of form  $\Delta_{i',j'}^{o'}$  with  $0 \leq i', j' \leq p-1$  and  $o' \in \{T, L, D, R\}$ .*

*Proof* The triangle  $\Delta_{i,j}^o$  has two vertices in  $\mathcal{V}$  and one vertex in  $\mathcal{C}$ . Let  $A$  and  $B$  be the vertices in  $\mathcal{V}$  and  $C$  be the vertex in  $\mathcal{C}$ . Lemma 2-3 ensures that  $f^{op}(A), f^{op}(B) \in \mathcal{V}$  while Lemma 2-4 gives that  $f^{op}(C) \in \mathcal{C}$ . Since rotations and reflections are rigid transformations, the result holds.  $\square$

Proposition 4 allows us to discard subproblems that are symmetric by rotations or reflections. For example, let  $\bar{r}_3 = \frac{5\sqrt{17}}{16}$  and  $r_3, S_{c_1, \theta_1}, S_{c_2, \theta_2}, S_{c_3, \theta_3}$  be a feasible arrangement for (6) with  $c_1 \in T_7, c_2 \in T_{12}$  and  $c_3 \in T_{22}$ . Then, Proposition 4 ensures that there exists a feasible arrangement  $r_3, S_{c'_1, \theta'_1}, S_{c'_2, \theta'_2}, S_{c'_3, \theta'_3}$  satisfying  $c'_1 \in T_{19}, c'_2 \in T_{12}$  and  $c'_3 \in T_{22}$ . Moreover, since  $T_{19}T_{12}T_{22}$  is obtained by a reflection around the  $y$  axis of  $T_7T_{12}T_{22}$ , we know that  $c'_i = f^y(c_i)$  for  $i = 1, 2, 3$ .

The tiling produced by Algorithm 2 suffices if one wants to use a complete global optimization approach for the packing problem. On the other hand, it is not suitable for a rigorous approach since the elements in  $\mathcal{V}$  and  $\mathcal{C}$  are floating point vectors subject to rounding errors. To overcome this problem, we introduce a scaled tiling. In this case, we ensure that the points at  $\mathcal{V}$  and  $\mathcal{C}$  are integer vectors to the cost of working with squares that are not unit but have the side length contained in a small interval  $\mathbf{s}$ . Algorithm 3 produces the scaled vertices for the tiling as well as the interval  $\mathbf{s}$ .

---

**Algorithm 3** Scaled tiling method.

---

**Input:** The upper bound  $\bar{r}$  for the radius in the SCSP.

**Output:** The sets of integer vectors  $\mathcal{V}$  and  $\mathcal{C}$  containing the vertices of  $\Delta_{i,j}^o$  for  $0 \leq i, j \leq p-1$  and  $o \in \{T, L, D, R\}$  and the scaling interval  $\mathbf{s}$ .

- 1:  $l \leftarrow \frac{2\bar{r}}{\lfloor 2\bar{r}+1 \rfloor}$ ;
  - 2:  $s \leftarrow \frac{2}{l}$ ;
  - 3:  $m \leftarrow \lfloor 2\bar{r} + 1 \rfloor$ ;
  - 4:  $v \leftarrow -m : 2 : m$ ;
  - 5:  $\mathcal{V} \leftarrow v \times v$ ;
  - 6:  $\mathcal{C} \leftarrow v_c \times v_c$ ;
  - 7: return  $(\mathcal{V}, \mathcal{C})$  and  $\mathbf{s}$ ;
- 

The elements in  $\mathcal{V}$  and  $\mathcal{C}$  are integer vectors by construction. Then, the Equations (8)-(11) are exactly representable. On the other hand, we replace the constant  $s$  in the Problem (6) by the interval  $\mathbf{s}$  to keep the mathematical certainty of our statements. The lemmas and propositions in the last section remain valid after the proper scaling. Figure 3-Right illustrates the scaled tiling for  $\bar{r}_3 = \frac{5\sqrt{(17)}}{16}$ . Note that the tiling would be the same for  $\bar{r}_4 = \sqrt{2}$  and the only difference between both cases would be the scaling interval  $\mathbf{s}$ .

Markót and Csentes [24] propose tiling constraints for the circle packing problem based on rectangles. The same idea could be used for the packing of squares into a circle. On the other hand for the case  $n = 3$ , one would need to split the search domain in 144 squares instead of 36 as proposed in this paper.

### 5 Packing 3 unit squares

Erich Friedman [9] gives an upper bound for the case  $n = 3$ ,  $\bar{r}_3 = \frac{5\sqrt{17}}{16}$ . Algorithm 3 gives the tiling displayed in Figure 3-Right and the interval scaling factor

$$s := [2.32834200034879, 2.32834200034880]. \quad (12)$$

Figure 4-Left displays an optimal configuration associated to the scaled version of the problem. This section proves the theorem below.

**Theorem 2** *Let  $r_3$  be the solution of (1) for  $n = 3$ . Then,*

$$r_3 \in [1.28847050800547, 1.28847050800553].$$

Moreover, the parameters of  $S_{c_1, \theta_1}$ ,  $S_{c_2, \theta_2}$  and  $S_{c_3, \theta_3}$  belong to the boxes in Table 6.

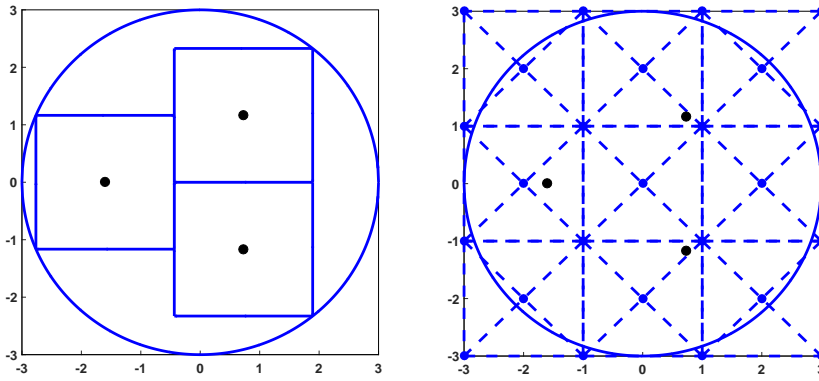


Fig. 4: Left: An optimal configuration for  $n = 3$ . Right: Triangles 7, 12 and 22 contain an optimal arrangement.

*Proof* We perform the computational part of the proof in a *core i7* processor with a frequency of 2.6 GHz, 6Gb of RAM and *Windows 10*. We compiled the code using the *g++ 7.3* compiler with the option *-O3*. A supplementary material for the proof, containing the statistics and the log files for each subproblem is available in <http://www.mat.univie.ac.at/~montanhe/publications/n3.zip>

We prove the theorem in three phases. At the  $i$ -th iteration, we consider instances of form (6) and define subproblems by adding tiling constraints of form (8)-(11) accordingly.

The proof considers the scaled version of the problem to ensure the mathematical certainty of our statements. Therefore, the CSPs in this section are of form (6) with the constant  $s$  replaced by the interval  $\mathbf{s}$  in Equation (12). We obtain the unscaled interval for  $r_3$  and Table 6 by dividing every box found in the last iteration by  $\mathbf{s}$ .

We also assume the labeling scheme for the triangles introduced in Section 4 and displayed on Figure 3-Right. Therefore, the subproblem  $T_7T_{12}T_{22}$  refers to the SCSP

with the interval scaling factor  $\mathbf{s}$  and such that  $c_1 \in T_7 := \Delta_{0,1}^R$ ,  $c_2 \in T_{12} := \Delta_{1,0}^T$  and  $c_3 \in T_{22} := \Delta_{1,2}^D$ .

**Phase 1.** In this iteration, we are interested in reducing the search domain of each subproblem and finding triangles which can contain the center of squares with no rotation. The tiling has 36 triangles, but the symmetries in  $\mathcal{V}$  and  $\mathcal{C}$  reduce the number of subproblems to 6. Table 2 shows the instances discarded without any processing in the first phase.

Table 2: Instances discarded in the first phase without processing. Symm. to stands for symmetric to. Symm. type shows the operation needed to obtain Instance from the element in the second column.

Instance	Symm. to	Symm. type	Instance	Symm. to	Symm. type
$T_2$	$T_1$	Ref. $y = x$	$T_{21}$	$T_6$	Rot. $90^\circ$
$T_3$	$T_0$	Ref. $y = x$	$T_{22}$	$T_{12}$	Rot. $180^\circ$
$T_6$	$T_4$	Ref. $x = 0$	$T_{23}$	$T_6$	Ref. $y = -x$
$T_8$	$T_2$	Ref. $x = 0$	$T_{24}$	$T_3$	Rot. $270^\circ$
$T_9$	$T_2$	Rot. $90^\circ$	$T_{25}$	$T_3$	Ref. $y = 0$
$T_{10}$	$T_3$	Rot. $90^\circ$	$T_{26}$	$T_2$	Ref. $y = 0$
$T_{11}$	$T_3$	Ref. $x = 0$	$T_{27}$	$T_2$	Rot. $270^\circ$
$T_{12}$	$T_7$	Rot. $270^\circ$	$T_{28}$	$T_6$	Rot. $180^\circ$
$T_{13}$	$T_6$	Ref. $y = x$	$T_{29}$	$T_{12}$	Rot. $270^\circ$
$T_{14}$	$T_5$	Rot. $270^\circ$	$T_{30}$	$T_6$	Ref. $y = 0$
$T_{15}$	$T_6$	Rot. $270^\circ$	$T_{31}$	$T_{14}$	Rot. $270^\circ$
$T_{17}$	$T_{16}$	Rot. $270^\circ$	$T_{32}$	$T_2$	Rot. $180^\circ$
$T_{18}$	$T_{17}$	Rot. $270^\circ$	$T_{33}$	$T_3$	Rot. $180^\circ$
$T_{19}$	$T_{17}$	Rot. $180^\circ$	$T_{34}$	$T_3$	Ref. $y = -x$
$T_{20}$	$T_{14}$	Rot. $180^\circ$	$T_{35}$	$T_2$	Ref. $y = -x$

Instances  $T_0, T_1, T_4, T_5, T_7$  and  $T_{16}$  require processing. We run the Algorithm 1 with  $\epsilon_T = 10^{-1}$ ,  $\epsilon_C = 10^{-11}$ ,  $\epsilon_x = 10^{-13}$  and time limit of 300 seconds. In this phase, we remove the condition  $\theta_1 = 0$  in Problem (6). Table 3 summarizes the results of the processed instances on phase 1. It shows that  $T_1$  and  $T_5$  are infeasible and any combination containing one of these triangles or their symmetric counterparts could be removed in the next phases. Moreover, it shows that only triangles  $T_7$  and  $T_{16}$  can contain the center of a square with rotation angle 0. Since we are assuming that  $\theta_1 = 0$  in the optimal configuration for  $n = 3$ , we only have to check the combinations containing at least one of these triangles.

**Phase 2.** This phase aims to discard as many instances as possible to reduce the number of hard subproblems in the last iteration. There are 630 possible combinations of 36 triangles taken 2 by 2. After eliminating symmetric and previously discarded cases, we obtain 43 instances. We also propagate any reduction in the search domain in the first phase to the subproblems in the second phase. Again, we remove the condition  $\theta_1 = 0$  from Problem (6).

We run the Algorithm 1 with  $\epsilon_T = 10^{-1}$ ,  $\epsilon_C = 10^{-11}$ ,  $\epsilon_x = 10^{-13}$  and time limit of 3600 seconds. We stop the algorithm as soon as the feasibility verification method described in Section 2 succeeds in finding a feasible point. The supplementary material contains the list of all instances discarded without processing. Table 4 gives the statistics for the 43 processed instances.



Table 3: Statistics for the processed instances on phase 1. Status gives the termination status of the instance. Time(s) gives the processing time in seconds. Column steps displays the number of calls of the state machine described in Table 1. Column  $\theta$  is a rigorous enclosure for the rotation angle.

Instance	Status	Time(s)	steps	$\theta$
$T_0$	Timeout	300	428253	[0.38528, 1.21015]
$T_1$	Infeasible	1	1	-
$T_4$	Timeout	300	421457	[0.33751, 1.51529]
$T_5$	Infeasible	1	1	-
$T_7$	Timeout	300	398689	[0, 1.5708]
$T_{16}$	Timeout	300	336345	[0, 1.5708]

Table 4: Statistics for the processed instances on phase 2. Status gives the termination status of the instance. Time(s) gives the processing time in seconds. Column steps displays the number of calls of the state machine described in Table 1.

Instance	Status	Time(s)	steps	Instance	Status	Time(s)	steps
$T_7T_{17}$	Infeasible	1	17697	$T_0T_{28}$	Feasible	1	1
$T_7T_{19}$	Feasible	1	1	$T_0T_{29}$	Feasible	1	1
$T_7T_{29}$	Feasible	1	1	$T_0T_{30}$	Infeasible	1	11395
$T_{16}T_{17}$	Feasible	1	3	$T_0T_{33}$	Feasible	1	1
$T_{16}T_{18}$	Feasible	1	2	$T_0T_{34}$	Feasible	1	1
$T_0T_3$	Infeasible	1	81	$T_4T_6$	Infeasible	2	1603
$T_0T_4$	Infeasible	2	1211	$T_4T_7$	Infeasible	1	2657
$T_0T_6$	Infeasible	2	233	$T_4T_{12}$	Feasible	1	1
$T_0T_7$	Infeasible	2	2219	$T_4T_{13}$	Infeasible	1	13771
$T_0T_{10}$	Infeasible	1	861	$T_4T_{15}$	Feasible	1	1
$T_0T_{11}$	Infeasible	2	1059	$T_4T_{16}$	Infeasible	1	24983
$T_0T_{12}$	Infeasible	1	2527	$T_4T_{17}$	Infeasible	1	4923
$T_0T_{13}$	Infeasible	1	243	$T_4T_{18}$	Feasible	1	1
$T_0T_{15}$	Infeasible	2	1535	$T_4T_{19}$	Feasible	1	1
$T_0T_{16}$	Feasible	1	1	$T_4T_{21}$	Infeasible	1	705
$T_0T_{17}$	Infeasible	1	3397	$T_4T_{22}$	Infeasible	2	19323
$T_0T_{18}$	Infeasible	1	3889	$T_4T_{28}$	Feasible	1	1
$T_0T_{19}$	Feasible	1	1	$T_4T_{29}$	Feasible	1	1
$T_0T_{21}$	Infeasible	1	8021	$T_4T_{30}$	Feasible	1	1
$T_0T_{22}$	Feasible	1	1	$T_7T_{12}$	Feasible	1	1
$T_0T_{23}$	Feasible	1	1	$T_7T_{16}$	Feasible	1	1
$T_0T_{24}$	Infeasible	2	1227				

We conclude the second phase with 22 infeasible subproblems. Again, any case in the next phase containing a combination found infeasible in this step can be discarded without any processing.

**Phase 3.** In this phase we set the full model in Problem (6), including the constraint  $\theta_1 = 0$ . Table 3 shows that  $c_1 \in T_7$  or  $c_1 \in T_{16}$ . Therefore, after removing the cases where one of these conditions do not hold and eliminating symmetric and already proved infeasible subproblems, we obtain 12 instances of the 7140 possible ones.

If an instance contains both triangles  $T_7$  and  $T_{16}$ , we denote by  $T_{7*}T_{16}T_x$  the case where we enforce the angle of the square centered in  $T_7$  to be zero. In the same

way, we write  $T_7T_{16*}T_x$  for the instances where the square centered in  $T_{16}$  has no rotation angle.

For the last phase, we run Algorithm 1 with  $\epsilon_T = 10^{-1}$ ,  $\epsilon_C = 10^{-11}$ ,  $\epsilon_x = 10^{-13}$  and no time limit. Table 5 provides the statistics of the processed instances. Moreover, Table 5 shows that it is the only instance containing the optimal configurations for  $n = 3$ . Figure 4-Right shows an approximation of the center of each square in the optimal case.

Algorithm 1 produces 4 clusters for the instance  $T_7T_{12}T_{22}$ . The maximum width of a cluster is  $6.23 * 10^{-13}$ . The precision is smaller than  $\epsilon_x$  due to the cluster builder procedure described in Section 2. Table 6 gives the unscaled clusters.

Table 5: Statistics for the processed instances on phase 3. Status gives the termination status of the instance. Time(s) gives the processing time in seconds. Column steps displays the number of calls of the state machine described in Table 1.

Instance	Status	Time(s)	steps
$T_{7*}T_{12}T_{16}$	Infeasible	33	146629
$T_7T_{12}T_{16*}$	Infeasible	134	440307
$T_7T_{12}T_{22}$	Clusters found	628	2183739
$T_{7*}T_{16}T_{18}$	Infeasible	3	16901
$T_7T_{16*}T_{18}$	Infeasible	3	18273
$T_{7*}T_{16}T_{19}$	Infeasible	4	19729
$T_7T_{16*}T_{19}$	Infeasible	1	5491
$T_{7*}T_{16}T_{29}$	Infeasible	8	40345
$T_7T_{16*}T_{29}$	Infeasible	2	11071
$T_{16}T_{17}T_{18}$	Infeasible	2	7319
$T_0T_{16}T_{19}$	Infeasible	0	2833
$T_0T_{16}T_{29}$	Infeasible	2	9317

□

## 6 Conclusion

This paper presents a framework for the rigorous optimization of the packing of unit squares into a circle. We express the question as the standard constraint satisfaction problem stated by Definition 1. The model considers the concept of sentinels to formulate non-overlapping constraints and the convexity of the squares and the circle to describe containment conditions.

General purpose rigorous optimization solvers cannot achieve the solution of the standard constraint satisfaction problem due to symmetries in the search domain. To overcome this difficulty, we propose a tiling method that splits the search space related to the center of each unit square into isosceles triangles. Our tiling divides the original problem into a set of subproblems that are suitable for the interval branch-and-bound approach. We also ensure that the parameters in each subproblem are free of rounding errors by introducing a proper scaling of the search domain.

To show the capabilities of our approach, we solve the first open case reported in the literature,  $n = 3$ . We implement the interval branch-and-bound in the *C++*

and the code is publicly available. We perform the proof on an ordinary laptop with 6 Gb of RAM and a core *i7* processor.

The proof of the case  $n = 3$  requires the solution of 6 subproblems with one square, 43 with two and only 12 with three squares. We discard most subproblems without processing due to symmetries in the tiling. Among the 61 subproblems, just 6 require more than 100 seconds to conclude the search. At the end of the process, we obtained 4 boxes with the following properties

1. The maximum width of any coordinate of the resulting boxes is  $6.23 * 10^{-13}$ .
2. If one disregard symmetries, every solution of (1) is contained in at least one of the 4 boxes

The method proposed in this paper could, in principle, be used to find the optimal arrangement for higher values of  $n$  (e.g.,  $n = 4, 5, 6$ ).

Table 6: Enclosures of the optimal arrangement for  $n = 3$ . There are 4 clusters, each of them separated by a blank line. The first coordinate of the center of the  $i$ -th square is given by  $\mathbf{c}_1$  and the second coordinate by  $\mathbf{c}_2$ . The rotation angle is given by  $\theta$ .

Square	$\mathbf{c}_1$	$\mathbf{c}_2$	$\theta$
1	[-0.687500000000001, -0.687499999999988]	[-0.000000000000018, 0.000000000000023]	[0.0, 0.0]
2	[0.312499999999993, 0.312500000000014]	[-0.500000000000007, -0.499999999999989]	[1.57079632679426, 1.57079632679491]
3	[0.312499999999993, 0.312500000000011]	[0.499999999999992, 0.500000000000007]	[-0.000000000000001, 0.0000000000000050]
1	[-0.687500000000001, -0.687499999999988]	[-0.000000000000021, 0.000000000000012]	[0.0, 0.0]
2	[0.312499999999998, 0.312500000000019]	[-0.500000000000003, -0.499999999999990]	[-0.000000000000001, 0.0000000000000036]
3	[0.312499999999993, 0.312500000000009]	[0.499999999999995, 0.500000000000007]	[-0.000000000000001, 0.0000000000000029]
1	[-0.687500000000001, -0.687499999999991]	[-0.000000000000012, 0.000000000000016]	[0.0, 0.0]
2	[0.312499999999993, 0.312500000000011]	[-0.500000000000007, -0.499999999999993]	[1.57079632679463, 1.57079632679491]
3	[0.312499999999998, 0.312500000000014]	[0.499999999999993, 0.500000000000004]	[1.57079632679453, 1.57079632679491]
1	[-0.687500000000001, -0.687499999999992]	[-0.000000000000014, 0.000000000000014]	[0.0, 0.0]
2	[0.312499999999998, 0.312500000000011]	[-0.500000000000003, -0.499999999999995]	[-0.000000000000001, 0.0000000000000027]
3	[0.312499999999998, 0.312500000000011]	[0.499999999999995, 0.500000000000004]	[1.57079632679463, 1.57079632679491]

## References

1. W. Bentz. Optimal packings of 13 and 46 unit squares in a square. *the electronic journal of combinatorics*, 17(1):R126, 2010.
2. E. G. Birgin, R. D. Lobato, and J. M. Martínez. Packing ellipsoids by nonlinear optimization. *Journal of Global Optimization*, 65(4):709–743, Aug 2016.
3. E. G. Birgin, R. D. Lobato, and J. M. Martínez. A nonlinear programming model with implicit variables for packing ellipsoids. *Journal of Global Optimization*, 68(3):467–499, Jul 2017.
4. E. G. Birgin, J. M. Martínez, W. F. Mascarenhas, and D. P. Ronconi. Method of sentinels for packing items within arbitrary convex regions. *Journal of the Operational Research Society*, 57(6):735–746, 2006.
5. I. Castillo, F. J. Kampas, and J. D. Pintér. Solving circle packing problems by global optimization: Numerical results and industrial applications. *European Journal of Operational Research*, 191(3):786–802, 2008.
6. F. Chung and R. L. Graham. Packing equal squares into a large square. *Journal of Combinatorial Theory, Series A*, 116:1167–1175, 2009.
7. F. Domes and A. Neumaier. Rigorous verification of feasibility. *Journal of Global Optimization*, 61:255–278, 2015.
8. P. Erdős and R. L. Graham. On packing squares with equal squares. *Journal of Combinatorial Theory(A)*, 19(1):9–123, 1975.
9. E. Friedman. Erich packing center. <http://www2.stetson.edu/~efriedma/packing.html>. Accessed: 2017-12-04.
10. E. Friedman. Packing unit squares in squares: A survey and new results. *The Electronic Journal of Combinatorics*, 2009.
11. E. R. Hansen. *Global optimization using interval analysis*. Marcel Dekker Inc., New York, 1992.
12. J. Kallrath and S. Rebennack. Cutting ellipses from area-minimizing rectangles. *Journal of Global Optimization*, 59(2):405–437, Jul 2014.
13. R. B. Kearfott. *Rigorous global search: Continuous problems*. Kluwer Academic Publishers, 1996.
14. Michael Lerch, German Tischler, Jürgen Wolff Von Gudenberg, Werner Hofschuster, and Walter Krämer. Filib++, a fast interval library supporting containment computations. *ACM Trans. Math. Softw.*, 32(2):299–324, June 2006.
15. M. C. Markót. Interval methods for verifying structural optimality of circle packing configurations in the unit square. *Journal of Computational and Applied Mathematics*, 199(2):353–357, 2007.
16. M. C. Markót and T. Csendes. A new verified optimization technique for the "packing circles in a unit square" problems. *SIAM Journal on Optimization*, 16:193–219, 2005.
17. W. F. Mascarenhas. Moore: Interval arithmetic in c++20. In Guilherme A. Barreto and Ricardo Coelho, editors, *Fuzzy Information Processing*, pages 519–529, Cham, 2018. Springer International Publishing.
18. W. F. Mascarenhas and E. G. Birgin. Using sentinels to detect intersections of convex and nonconvex polygons. *Computational & Applied Mathematics*, 29(2):247–267, 2010.
19. H. Nagamochi. Packing unit squares in a rectangle. *the electronic journal of combinatorics*, 12(1):R37, 2005.

20. A. Neumaier. *Interval methods for systems of equations*, volume 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge Univ. Press, Cambridge, 1990.
21. H. Schichl and A. Neumaier. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, 33(4):541–562, 2005.
22. E. Specht. Packomania. <http://www.packomania.com/>. Accessed: 2017-12-04.
23. W. Stromquist. Packing 10 or 11 unit squares in a square. *The Electronic Journal of Combinatorics*, 10(8):1–11, 2003.
24. P. G. Szabó, M. C. Markót, T. Csendes, E. Specht, L. G. Casado, and I. García. *New Approaches to Circle Packing in a Square - With Program Codes*. Springer, Berlin, 2008.