Packing Circles within Ellipses[*]

E. G. Birgin[†] L. H. Bustamante[†] H. F. Callisaya[‡] J. M. Martínez[‡]

March 14, 2012[§]

Abstract

The problem of packing circles within ellipses is considered in the present paper. A new ellipse-based system of coordinates is introduced by means of which a closed formula to compute the distance of an arbitrary point to the boundary of an ellipse exists. Nonlinear programming models for some variants of 2D and 3D packing problems involving circular items and elliptical objects are given. The resulting models are medium-sized highly nonlinear challenging nonlinear programming problems for which a global solution is sought. For this purpose, multistart strategies are carefully and thoroughly explored. Numerical experiments are exhibited.

Key words: Packing, ellipses, circles, global optimization.

1 Introduction

The problem of packing items within bounded regions in the Euclidean space has multiple applications in Physics, Chemistry, Engineering, Geophysics, and Arts. Problems of this type exhibit various levels of complexity. The easiest problems involve packing simple items within simple domains and, sometimes, analytical solutions for these problems exist [14]. Frequently, simple problems motivate interesting recreational applications [16]. On the other extreme of the simplicity range, we encounter the problem of packing irregular items within irregular domains. In this case, close solutions do not exist, but there are important applications to environmental sciences. The irregular packing of soil particles, for example, largely determines the way in which root formation of plants occurs, as well as water movements in the soil [18].

Molecular Dynamics (MD) motivated the development of a very successful approach for packing molecules on different domains of the Euclidean 3D-space. The package Packmol [21, 22] employs nonlinear optimization procedures [1, 7] and heuristics for distributing molecules on given domains with the purpose of building suitable low-energy initial configurations for MD

[*]This work was supported by PRONEX-CNPq/FAPERJ (E-26/111.449/2010-APQ1), FAPESP (2006/53768-0, 2009/10241-0, and 2010/10133-0) and CNPq (Grant 306802/2010-4).

[†]Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, São Paulo SP, Brazil. e-mail: {egbirgin | lhbusta}@ime.usp.br

[‡]Department of Applied Mathematics, Institute of Mathematics, Statistics and Scientific Computing, University of Campinas, Campinas, SP, Brazil. e-mail: {fchect | martinez}@ime.unicamp.br

[§]Revised on September 3, 2012.

simulations. The enormous number of applications motivates detailed analysis and experimentation with geometrical problems of increasing complexity. Completely arbitrary packing problems can be addressed by techniques that, in general, are computationally expensive. Sometimes, at each step of a packing problem an expensive auxiliary problem needs to be solved. For example, for packing balls within an arbitrary domain the balls' centers must be kept within the domain and a procedure to compute the distance to the boundary must be available. This involves to solve a potentially expensive optimization problem for each trial configuration of the packing problem. On the other hand, there exist extremely simple but useful situations, like packing balls within a square or a convex polygon. We are convinced that solving with affordable methods increasingly complex situations is useful in the process of finding practical procedures for the arbitrarily complex cases. For this purpose, in the present paper we tackle the problem of packing circles within ellipses. The technology developed in connection to this geometrical problem opens the possibility of addressing more and more complex cases for relevant large-scale applications, as the ones considered in the MD field.

Nonlinear programming formulations and methods had been successfully applied to a wide range of circle packing problems. A short description of recent works follows. Based on the Φ-functions framework, a variety of packing problems had been solved formulating them as nonlinear programming models and combining clever initial guesses with local and global optimizations techniques for pursuing a model global solution. Circles and non-convex polygons with rotations packed within a multiple connected region were considered in [28], the maximum number of identical circles that can be packed within a circle with prohibited areas were considered in [26], and packing various-sized circles within a strip was considered in [25], among others. MINLP models for cutting circles and convex polygons from rectangles with minimum area were introduced in [19]. A review of NLP models for solving several classes of circle packing problems was presented in [13], where several applications are surveyed and the relevance of nonlinear global optimization techniques for solving circle packing problems is highlighted.

Nonlinear systems of equations are solved with Newtonian methods in [5] to find high accuracy solutions to the problem of packing identical circles within a variety of containers. As initial guesses, solutions found in [10] with classical multistart strategies and local augmented Lagrangian methods were considered. Local augmented Lagrangian techniques and multistart strategies were also considered in [9] for tackling the problem of packing the maximum number of identical-sized circles within circular and rectangular containers. Other former works that considered nonlinear programming models for circle packing problems can be found in [10] and the references therein. In all cases, circles are packed within circular or polygonal (mainly rectangular) containers. In these cases, the set of constraints that say the circular items must be placed within the container (i.e. with their centers within the container and a radius far from the border) can be easily written as a set of constraints applied to the circles' centers and involving a resized container. This is not the case for elliptical containers.

The problem of packing circles within ellipses, as well as extensions to the three-dimensional case in which the container definition involves ellipses, are considered in the present work. The novelty of the introduced approach is based on a new ellipse-based system of coordinates. The new system of coordinates is based on a reference ellipse and provides a closed formula to compute the distance of an arbitrary point to the boundary of the reference ellipse. This is the key ingredient for the formulation of the tackled packing problems as a sequence of increasinglydimension continuous and differentiable nonlinear programming problems. The resulting models are medium-sized highly nonlinear challenging nonlinear programming problems for which a global solution is sought. For this purpose, multistart strategies are carefully and thoroughly explored.

The rest of this paper is organized as follows. The new ellipse-based system of coordinates is introduced in Section 2. Nonlinear programming models for several packing problems involving ellipses are presented in Section 3. The basic nonlinear optimization procedures are described on Section 4. The multistart strategies are presented in connection to the numerical experiments in Section 5. Conclusions are given in Section 6.

2 Ellipse-based system of coordinates

We consider the ellipse defined by $u^2/a^2 + v^2/b^2 = 1$, where $a \geq b$. Points (x, y) in the solid region delimited by this ellipse may be parametrized in the following way:

$$\begin{gathered} x = [1 + (s-1)(b^2/a^2)]u, \quad y = sv, \\ u^2/a^2 + v^2/b^2 = 1, \\ 0 \leq s \leq 1, \end{gathered} \tag{1}$$

where u, v, s are continuous variables. If (x, y) is given by (1), the point on the ellipse's boundary which is closest to (x, y) is (u, v). Consequently, the distance between (x, y) and the boundary of the ellipse is

$$(1-s)\sqrt{(b^2/a^2)^2u^2 + v^2}. \tag{2}$$

An equivalent formulation with only two variables (instead of three) may be obtained substituting u by $a\cos(t)$, v by $b\sin(t)$, adding the constraint $0 \leq t \leq 2\pi$, and removing the constraint $u^2/a^2 + v^2/b^2 = 1$ that is automatically satisfied. In this way, we arrive to the equivalent parametrization:

$$\begin{gathered} x = [1 + (s-1)(b^2/a^2)]\, a\cos(t), \quad y = s\, b\sin(t), \\ 0 \leq t \leq 2\pi, \quad 0 \leq s \leq 1. \end{gathered} \tag{3}$$

In this case, the point on the boundary which is closest to (x, y) is $(a\cos(t), b\sin(t))$. Consequently, the distance between (x, y) and the boundary of the ellipse is given by

$$(1-s)\frac{b}{a}\sqrt{b^2\cos(t)^2 + a^2\sin(t)^2}. \tag{4}$$

In (3), each point (x, y) is represented by the pair of continuous variables (s, t). Constraint $s \in [0, 1]$ implies that (x, y) belongs to the solid region delimited by the ellipse. In fact, if $s = 1$, (x, y) is on the ellipse (boundary) and, if $s \in [0, 1)$, (x, y) is an interior point of the ellipse. If $s > 1$ then (x, y) does not belong to the solid ellipse, being able to represent any other point in the plane. Therefore, replacing $s \in [0, 1]$ by $s \in [0, +\infty)$, (3) represents a parametrization of the plane. This parametrization is said to be ellipse-based because it depends on a given ellipse with semi-axes $a \geq b$ and arbitrarily centered at the origin of the Cartesian axes. The analogy with polar coordinates is clear. This new coordinate system for the plane preserves an important

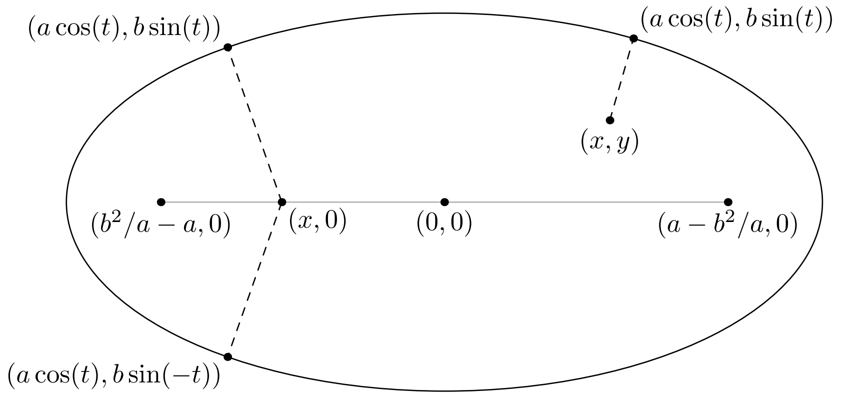


Figure 1: Parametrization defined in (3) and, for a given (x, y), the point on the boundary that is closest to (x, y).

property of polar coordinates: it is "distance-revealing" in the sense that, given (s, t), one can compute the distance between the point (x, y) determined by (3) and the ellipse boundary using the simple formula (4). Moreover the point on the boundary that realises this distance is also revealed by the (s, t) coordinates and is given by $(a\cos(t), b\sin(t))$. In Figure 1 we show graphically how this parametrization works. Observe that, for all the points in the line segment between $(b^2/a - a, 0)$ and $(a - b^2/a, 0)$, there are two points that realise the distance to the boundary. More theoretical properties and extensions of this parametrization can be found in [15]. In the present work, we opted by dealing with the three-variables parametrization given by (1) in the forthcoming sections since, in this case, constraints of the constructed optimization models are formulated as polynomials.

A different parametrization based on "parallel curves" has been used in [16] to solve different instances of the problem of finding the ellipse with smallest area that encloses a number of fixed unitary circles. The solutions exhibited in [16] have been obtained using a commercial global optimization package [12]. Roughly speaking, the concept of parallel curves is as follows. Given $r \geq 0$, we say that the curve $\gamma(r)$ is parallel to the curve γ if for all $P \in \gamma(r)$ the distance between P and γ is r. Clearly, the circle with radius $1 - r$ is parallel to the unitary circle, but the analogous property is not true for ellipses, i.e. curves parallel to ellipses are not ellipses. The problem of packing balls of radius r within an ellipse is equivalent to enclosing the balls' centers (with pairwise distances at least $2r$) by the parallel curve at distance r to the ellipse boundary. Unfortunately, when r grows, the region enclosed by the parallel curve is tricky not at all similar to an ellipse. There is a lot of difficulty in addressing the cases in which parallel curves cease to enclose convex domains, as happens to occur when we want to consider balls with not small radius or ellipses with high eccentricity. In these cases the parallel-curve approach cannot be used. On the other hand, our approach is general enough to encompass all the cases, independently of the dimensions of the balls and the ellipse's eccentricity.

3 Nonlinear programming models

In this section, we introduce continuous and differentiable nonlinear programming models for four packing problems. These four arbitrarily chosen problems are examples of the many variations of 2D and 3D packing problems involving circular or spherical items and elliptical containers that can be formulated as smooth and continuous nonlinear programming problems with the help of the parametrization given by (1). Other examples including conic sections and intersections of ellipses can be found in [15].

3.1 Given circles within a given ellipse

Consider the problem of finding m non-overlapping circles with radii r_i, $i = 1, \dots, m$, which are contained in a solid ellipse defined by $a \geq b$. Let us denote by (x_i, y_i) the center of each circle. Then, in order to find the solution of the problem, we must solve the feasibility problem given by:

$$(u_i/a)^2 + (v_i/b)^2 = 1, \; i = 1, \dots, m, \tag{5}$$

$$\left(\frac{s_i - 1}{r_i}\right)^2 \left[(b^2/a^2)^2 u_i^2 + v_i^2\right] \geq 1, \; i = 1, \dots, m, \tag{6}$$

$$\frac{1}{(r_i + r_j)^2} \left\{ \left[(1 + (s_i - 1)(b^2/a^2))u_i - (1 + (s_j - 1)(b^2/a^2))u_j\right]^2 + [s_i v_i - s_j v_j]^2 \right\} \geq 1, \; \forall\, j > i, \tag{7}$$

$$0 \leq s_i \leq 1, \; i = 1, \dots, m. \tag{8}$$

In (5–8), variables $u_i, v_i, s_i \in \mathbb{R}$ represent the i-th circle, whose center's coordinates (x_i, y_i) can be recovered, by (1), using

$$x_i = \left[1 + (s_i - 1)(b^2/a^2)\right] u_i, \quad y_i = s_i v_i, \quad i = 1, \dots, m. \tag{9}$$

Similarly, employing (3), a feasibility problem by means of which we may compute the desired circles can be formulated with unknowns t_i and s_i, $i = 1, \dots, m$. We prefer (5–8) in our experiments because all the constraints are formulated as polynomials. The unknowns of the nonlinear feasibility problem (5–8) are (u_i, v_i), $i = 1, \dots, m$ and s_i, $i = 1, \dots, m$. Equalities and inequalities in (5), (6), and (7) define $2m + m(m-1)/2$ nonlinear constraints and (8) defines m bound constraints on the variables s_i.

3.2 Maximizing the number of identical circles within a given ellipse

We wish to maximize the number of circles with radii r that can be packed in a given ellipse. Clearly, if, for a fixed m, the feasibility problem (5–8) has a solution, then the maximal number of enclosed balls is at least m. The solution of this problem involves solving a sequence of feasibility problems with variable m. Assuming that a lower bound m_{lb} on the number of identical balls with radii r that can be packed within the given ellipse is known (m_{lb} equal to zero might be a feasible choice), the procedure starts by trying to solve feasibility problem (5–8) with $m = m_{lb} + 1$. If a feasible solution is found, we set $m \leftarrow m + 1$ and we try again. The process stops when m is such that a feasible solution of (5–8) cannot be found, and the solution found for (5–8) with $m^* = m - 1$ is considered as the solution of packing as many identical balls as possible. Similar strategies have been employed in [5, 6, 8, 9] where empirical justifications for the use of this kind of sequential process of increasing m one by one, instead of other strategies such as bisection, are given.

3.3 Minimizing the area of the ellipse that encloses a set of circles

We wish to find the ellipse of smallest area that encloses m circles with radii $r_1, \dots, r_m$. In principle, the problem may be formulated as the nonlinear programming problem

$$\text{Minimize } ab \tag{10}$$

subject to (5–8), $a \geq 0$, and $b \geq 0$. Note that a and b are variables of this problem, unlike the previous ones, in which this quantities were given data. Moreover, since the validity of the parametrization requires that $a \geq b$, it is convenient to define a new slack variable $w \geq 0$ such that $a = b + w$. Therefore, the non-negativity constraints $a \geq 0$, and $b \geq 0$ must be replaced by

$$b \geq 0 \ \text{ and } w \geq 0, \tag{11}$$

and all the occurrences of a in the objective function (10) and the constraints (5–8) must be replaced by $b + w$.

3.4 Balls in a cylindrical container

The three 2D problems described in the previous subsections can be extended to consider the 3D situation. As an example, we describe the problem in which we have m 3D-balls with radii $r_1, \dots, r_m$ and we wish to pack these m balls within a cylinder whose basis is an ellipse with half-axes $a \geq b$. We consider that the objective is to minimize the container's height.

Let us denote by (x_i, y_i, z_i), $i = 1, \dots, m$, the centers of the required balls. As in (9) x_i and y_i are written as functions of variables u_i, v_i, s_i, $i = 1, \dots, m$, such that $(u_i, v_i, 0)$ is the point on the boundary of the elliptical cylinder's basis that is closest to $(x_i, y_i, 0)$ – the projection of (x_i, y_i, z_i) onto the x-y plane. Therefore, constraints (5) and (8) are part of this new problem. Constraint (6) imposes the projection of each ball's center onto the x-y plane to be contained in the ellipse. Therefore, constraint (6) is also present in this new problem. Finally, the 3D non-overlapping constraints are given by

$$\frac{1}{(r_i + r_j)^2} \left\{ \left[(1 + (s_i - 1)(b^2/a^2))u_i - (1 + (s_j - 1)(b^2/a^2))u_j\right]^2 + [s_i v_i - s_j v_j]^2 + [z_i - z_j]^2 \right\} \geq 1, \ \forall\, j > i, \tag{12}$$

The height variables z_i are subject to the constraint

$$0 \leq z_i \leq z_{\max} \tag{13}$$

and the objective of the problem is to

$$\text{Minimize } z_{\max}$$

subject to (5,6,8,12,13).

4 Optimization procedures

All the problems presented in Section 3 have been formulated under the Nonlinear Programming framework. They have the mathematical form

$$\text{Minimize } f(x) \text{ subject to } h(x) = 0,\ g(x) \le 0,\ \ell \le x \le u, \tag{14}$$

where f is a scalar smooth function, h and g are vectorial smooth functions and the prescribed bounds on x can be minus or plus infinite.

Our main tool for solving (14) will be Algencan [1, 2]. Algencan is a code for solving large-scale nonlinear programming problems which is periodically updated and publicly available at the TANGO Project web site *http://www.ime.usp.br/∼egbirgin/tango/*. Algencan converges, under reasonable assumptions, to local minimizers of the original problem. Convergence to global minimizers has been enhanced in the present research by means of procedures that evoke well-known multistart approaches for stochastic global optimization. A different globalization procedure for Algencan was introduced in [4] with guaranteed convergence to global minimizers, but an implementation for large-scale problems is not available yet.

The feasibility problem (5–8), related to the problem of maximizing the number of identical circles within a given ellipse, is a particular case of (14) in which the objective function is null, i.e. it can be written in the form

$$\text{Find } x \text{ such that } h(x) = 0,\ g(x) \le 0,\ \ell \le x \le u, \tag{15}$$

where $x = (u, v, s) \in \mathbb{R}^{3m}$ from now on. A feasibility problem of the form (15) is equivalent to the bound-constrained minimization problem

$$\text{Minimize } f(x) \text{ subject to } \ell \le x \le u, \tag{16}$$

where

$$f(x) = \|h(x)\|_2^2 + \|g(x)_+\|_2^2 \tag{17}$$

and $[g(x)_+]_i = \max\{0, g_i(x)\}$. If $f(\cdot)$ vanishes at a *global* minimizer x^* of (16) then x^* is a feasible solution to (15). If, on the other hand, $f(x^*) > 0$ then (15) is infeasible. For the particular case of the feasibility problem (5–8), there are practical reasons for considering the bound-constrained minimization formulation (16) instead of the feasibility problem formulation (15) that we now describe. Computing $f(\cdot)$ involves $O(m^2)$ constraints. However, since a sum involving the values of those constraints is needed, instead of their individual values as it would be needed for solving (15) by a traditional approach, a strategy developed for the N-body problem [17] can be applied to reduce the complexity of evaluating $f(\cdot)$. This strategy, successfully being used by the MD packing software Packmol [21, 22], is fully described in [10], where extensive numerical experiments show the reductions obtained in the cost of evaluating $f(\cdot)$.

5 Experiments

In this section we report numerical experiments for the problem of packing as many identical unitary-radius circles as possible within a given ellipse. All tests were conducted on a 2.4GHz

INTEL CORE 2 QUAD with 8GB of RAM memory and running GNU/LINUX operating system (Ubuntu 11.10, kernel 3.0.0-14). The code, fully written in FORTRAN 77, was compiled by the GFortran FORTRAN compiler of GCC (version 4.6.1) with the `-O3` optimization directive enabled.

As it was mentioned in the previous section, the solution method consists in solving a sequence of the bound-constrained minimization problem (16) for increasing values of $m = m_{lb}+1, m_{lb}+2, \dots$. In this case, feasibility problem (15) takes the form

$$\begin{array}{rcll} h_i^{\xi}(u,v,s) & = & 0, & i=1,\dots,m,\\ g_i^{\xi}(u,v,s) & \le & 0, & i=1,\dots,m,\\ \kappa_{ij}^{\xi}(u,v,s) & \le & 0, & i=1,\dots,m,\ j=i+1,\dots,m,\\ 0 \ \le \ s_i & \le & 1, & i=1,\dots,m, \end{array} \tag{18}$$

where

$$\begin{array}{rcl} h_i^{\xi}(u,v,s) & = & 1-\left[(u_i/a)^2+(v_i/b)^2\right],\\ g_i^{\xi}(u,v,s) & = & 1-(\frac{s_i-1}{r})^2\left[(b^2/a^2)^2u_i^2+v_i^2\right],\\ \kappa_{ij}^{\xi}(u,v,s) & = & 1-\frac{1}{4r^2}\left[\left\{(1+(s_i-1)(b^2/a^2))u_i-(1+(s_j-1)(b^2/a^2))u_j\right\}^2+\left\{s_iv_i-s_jv_j\right\}^2\right], \end{array} \tag{19}$$

and $\xi=(a,b,r,m)$ is used to stress the dependency on the problem's parameters. Therefore, problem (16) takes the form

$$\text{Minimize } f^{\xi}(u,v,s) \text{ subject to } 0 \le s \le 1, \tag{20}$$

where

$$f^{\xi}(u,v,s)=\frac{1}{2}\left\{\sum_{i=1}^{m}h_i^{\xi}(u,v,s)^2+\sum_{i=1}^{m}g_i^{\xi}(u,v,s)_+^2+\sum_{i=1}^{m}\sum_{j=i+1}^{m}\kappa_{ij}^{\xi}(u,v,s)_+^2\right\}.$$

If, for a given m, a minimizer of (20) with null objective function is found, the value of m is increased by one and the process continues. Otherwise, we keep trying for this value of m until an imposed CPU time limit is reached. This strategy lies in two facts: (a) a stochastic global optimization method based on multistart strategies is used for the process of looking for a global minimizer of (20), and (b) we have the ability of recognizing that a global minimizer was found only if the global minimum of the instance being solved is zero. The whole procedure is described by Algorithm 4.1.

Algorithm 4.1: Let $a \ge b > 0$ and $r > 0$ be given constants.

Step 1. Compute a lower bound m_{lb}, i.e. a value $m_{lb} \ge 0$ such that $(u^*,v^*,s^*) \in \mathbb{R}^{3m_{lb}}$ with $f^{(a,b,r,m_{lb})}(u^*,v^*,s^*)=0$ is known (note that $m_{lb}=0$ is a valid choice). Set $m \leftarrow m_{lb}+1$.

Step 2. If the time limit is achieved, stop declaring that $m^*=m-1$ items were packed.

Step 3. Compute a random initial guess $(u_{\text{ini}}, v_{\text{ini}}, s_{\text{ini}})$ for problem (20).

Step 4. Find a stationary point (u^*, v^*, s^*) of (20) starting from $(u_{\rm ini}, v_{\rm ini}, s_{\rm ini})$.

Step 5. If $f^{(a,b,r,m)}(u^*, v^*, s^*)$ vanishes, set $m = m + 1$.

Step 6. Go to Step 2.

In the following subsections we describe each step of Algorithm 4.1 in detail.

5.1 Computation of the lower bound m_{lb}

To be used in connection with the optimization strategy for packing as many identical circles as possible within an ellipse, we developed a lower bound based on the 2D bee's honeycomb or hexagonal lattice (see Figure 2). This lattice is the densest 2D lattice and non-lattice packing in the (infinite) plane with density $\frac{1}{6}\pi\sqrt{3} \approx 0.9068996821$ [30]. In a similar context, lattices were also considered in [27] to generate initial guesses for an optimization procedure.

Given a point $(\bar{x}, \bar{y})^T \in \mathbb{R}^2$, we name by $\mathcal{L}(\bar{x}, \bar{y})$ the infinite set of points of the hexagonal lattice whose "rows" are parallel to the horizontal Cartesian axis and such that $(\bar{x}, \bar{y})^T \in \mathcal{L}(\bar{x}, \bar{y})$. Figure 2(a–d) illustrates lattices $\mathcal{L}(r, 0)$, $\mathcal{L}(0, 0)$, $\mathcal{L}(r, r)$, $\mathcal{L}(0, r)$, with $r = 1$, respectively. The computation of the lower bound m_{lb} on the number of identical circles with radii r that can be packed within an ellipse given by $a \geq b$ starts by considering a lattice $\mathcal{L}(\bar{x}, \bar{y})$ and computing the finite set of points $P(\bar{x}, \bar{y}) = \mathcal{L}(\bar{x}, \bar{y}) \cap \{(x, y)^T \in \mathbb{R}^2 \mid (x/a)^2 + (y/b)^2 \leq 1\}$. Those are the centers of dark and light-grey circles in Figure 2, that are "candidates" to be inside the ellipse. Then, for each $(x_i, y_i)^T \in P(\bar{x}, \bar{y})$, we look for u_i, v_i, and $0 \leq s_i \leq 1$, such that

$$\begin{array}{rcl} \left[1 + (s_i - 1)(b^2/a^2)\right] u_i & = & x_i, \\ s_i v_i & = & y_i, \\ (u_i/a)^2 + (v_i/b)^2 & = & 1, \\ ((s_i - 1)/r)^2 \left[(b^2/a^2)^2 u_i^2 + v_i^2\right] & \geq & 1, \end{array} \tag{21}$$

to check whether the circle is inside the ellipse or not. This (three-unknowns) feasibility problem is solved with Algencan. In Figure 2, light-grey circles are the ones inside the ellipse, while the dark-grey circles are the ones with their center inside the ellipse but not totally contained inside the ellipse. The number of light-grey circles is the lower bound given by the lattice $\mathcal{L}(\bar{x}, \bar{y})$.

Different lattices may provide different lower bounds. For example, lattices in Figure 2(a–d) provide lower bounds equal to 374, 365, 368, and 366, respectively, for the instance with $a = 2b = 30$ and $r = 1$. In this work, for each ellipse given by $a \geq b$ and circles of radii r, the lower bound m_{lb} is computed as the biggest lower bound obtained by any of the four lattices depicted on Figure 2. Table 1 shows some numerical results. It can be observed that: (a) none of the four considered lattices is better than the others, in the sense of providing a better lower bound for all the considered cases; (b) the larger the ellipse, the better the lower bound (if we consider that a larger density is an indication that the lower bound is nearer to the optimal solution); and (c) the larger the ratio a/b, the poorer the lower bound is. From (b) and (c), we see that the described strategy provides "reasonable" lower bounds for large and "fat" ellipses, leaving more space for cheaply finding improved solutions to the instances with small or narrow ellipses. The whole process of computing a lower bound as described takes less than 0.1 second of CPU time in the computational environment described at the beginning of the section.

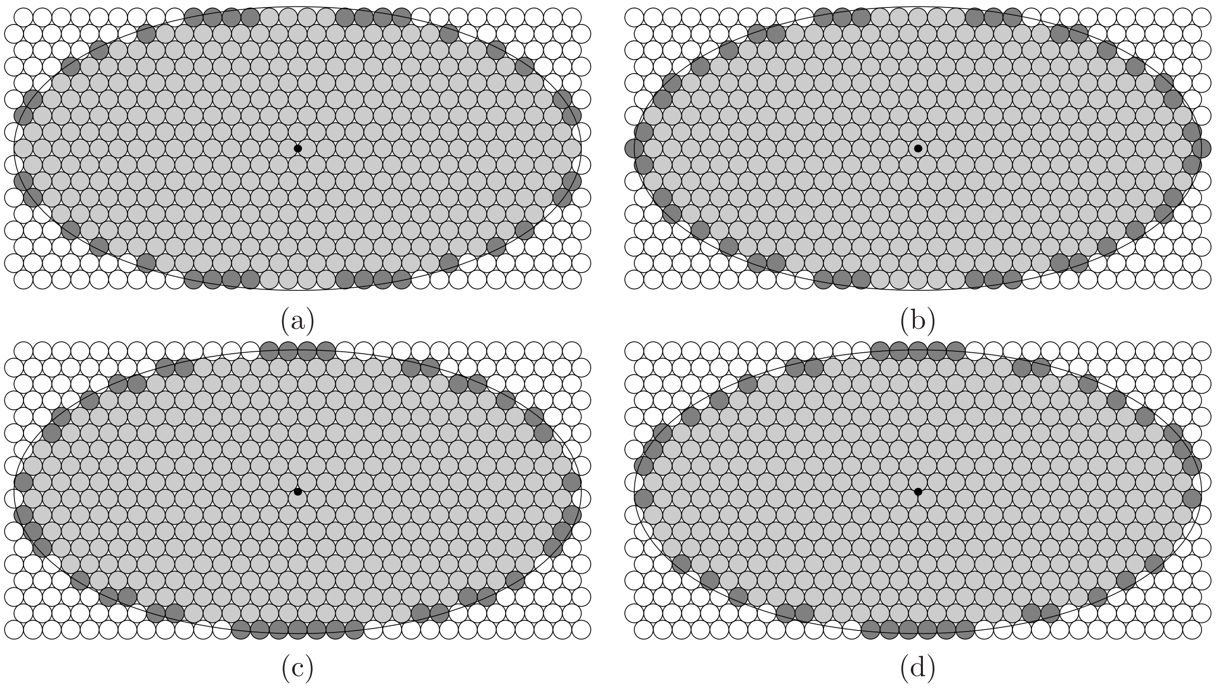


Figure 2: (a–d) Lattices $\mathcal{L}(r,0)$, $\mathcal{L}(0,0)$, $\mathcal{L}(r,r)$, $\mathcal{L}(0,r)$, respectively, illustrating how they are used to build lower bounds for the instance with $a = 30$, $b = 15$, and $r = 1$. White circles are the circles in the lattices whose centers are outside the ellipse. Dark-grey circles are the ones whose centers are inside the ellipse but that are not completely contained inside the ellipse. Light-grey circles are the ones contained inside the ellipse and that contribute to the computation of the lower bound. The four lattices provide the lower bounds 374, 365, 368, and 366, respectively, i.e. $\mathcal{L}(r,0)$ provides the best lower bound for this instance.

Instance					Lattices' lower bounds				Best lower bound		CPU Time
	Name	a	b	a/b	$\mathcal{L}(r,0)$	$\mathcal{L}(0,0)$	$\mathcal{L}(r,r)$	$\mathcal{L}(0,r)$	m_{lb}	Density	(in secs.)
Set 1	s1i01	4	2	2	**4**	3	1	3	4	0.5000	0.02
	s1i02	6	3	2	**12**	9	9	9	12	0.6667	0.03
	s1i03	8	4	2	18	19	**20**	18	20	0.6250	0.03
	s1i04	10	5	2	**36**	35	30	30	36	0.7200	0.05
	s1i05	12	6	2	50	45	49	**51**	51	0.7083	0.06
	s1i06	14	7	2	**74**	71	67	68	74	0.7551	0.10
	s1i07	16	8	2	92	**95**	94	92	95	0.7422	0.12
	s1i08	18	9	2	**126**	121	123	121	126	0.7778	0.17
	s1i09	20	10	2	158	**159**	151	155	159	0.7950	0.21
	s1i10	22	11	2	**192**	185	190	**192**	192	0.7934	0.24
	s1i11	24	12	2	**238**	225	226	222	238	0.8264	0.26
	s1i12	26	13	2	268	267	**273**	**273**	273	0.8077	0.32
	s1i13	28	14	2	**320**	317	311	314	320	0.8163	0.43
	s1i14	30	15	2	**374**	365	368	366	374	0.8311	0.47
Set 2	s2i01	32.1429	14	2.2959	368	**369**	363	364	369	0.8200	0.45
	s2i02	34.6154	13	2.6627	360	359	365	**367**	367	0.8156	0.45
	s2i03	37.5000	12	3.1250	362	**363**	354	360	363	0.8067	0.45
	s2i04	40.9091	11	3.7190	354	359	**364**	360	364	0.8089	0.48
	s2i05	45.0000	10	4.5000	358	**361**	353	353	361	0.8022	0.48
	s2i06	50.0000	9	5.5556	**354**	349	349	349	354	0.7867	0.47
	s2i07	56.2500	8	7.0313	340	339	**346**	346	346	0.7689	0.53
	s2i08	64.2857	7	9.1837	340	**341**	329	326	341	0.7578	0.50
	s2i09	75.0000	6	12.5000	308	307	329	**331**	331	0.7356	0.46
	s2i10	90.0000	5	18.0000	318	**323**	290	292	323	0.7178	0.51
	s2i11	112.5000	4	28.1250	274	273	**296**	**296**	296	0.6578	0.53
	s2i12	150.0000	3	50.0000	264	**265**	235	233	265	0.5889	0.56
Set 3	s3i01	100.0000	2.0000	50.0000	86	**87**	49	51	87	0.4350	0.28
	s3i02	75.0000	1.5000	50.0000	**56**	55	0	0	56	0.4978	0.15
	s3i03	75.0000	2.6667	28.1250	70	69	**107**	**107**	107	0.5350	0.24
	s3i04	56.2500	2.0000	28.1250	48	**49**	29	29	49	0.4356	0.15
	s3i05	60.0000	3.3333	18.0000	**128**	125	99	99	128	0.6400	0.25
	s3i06	45.0000	2.5000	18.0000	42	41	**59**	**59**	59	0.5244	0.14

Table 1: Description of the test sets. Set 1 corresponds to ellipses with increasing area and constant ratio a/b, Set 2 corresponds to ellipses with constant area (equal to the area of the largest instance in Set 1) and increasing ratio a/b, and Set 3 corresponds to "narrow" ellipses with relatively large area and large ratio a/b.

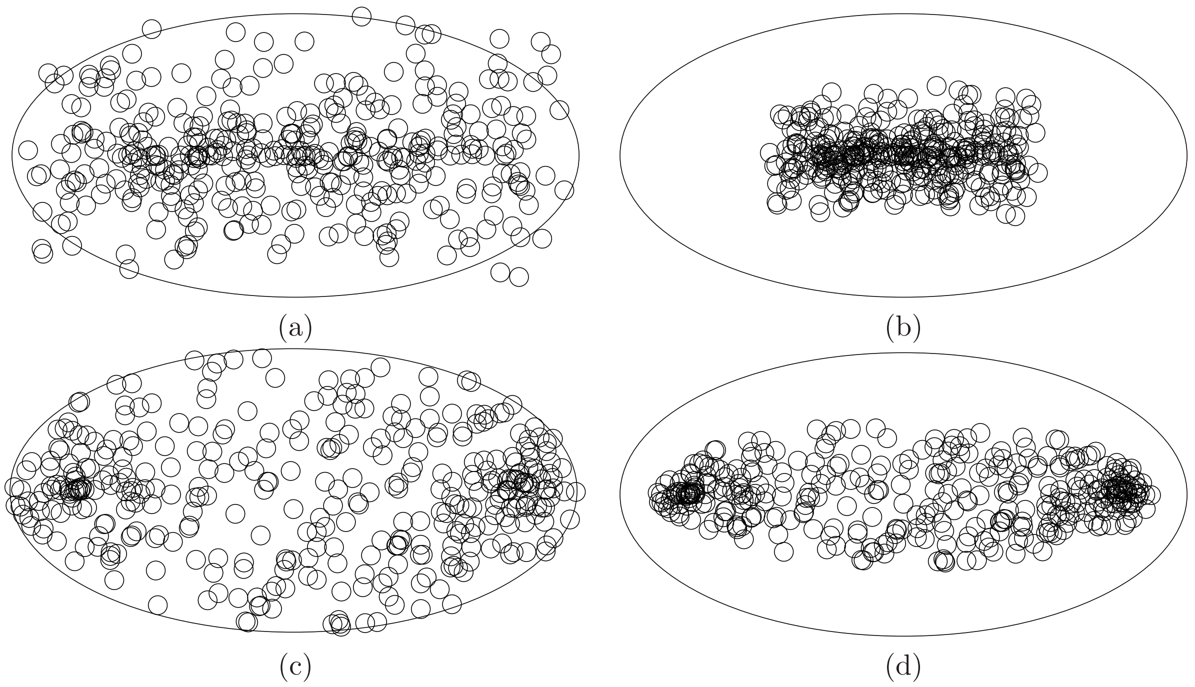


Figure 3: (a–d) Random initial guesses R1, R2, R3, and R4, respectively, for trying to pack $m = 375$ circles (one more than the number of circles given by the lattice-based lower bound) for the instance with $a = 30$, $b = 15$, and $r = 1$.

5.2 Plain alternatives for multistart initial guesses

In this subsection, we describe different simple alternatives for generating initial guesses to find a stationary point of (20) at Step 4 of Algorithm 4.1. We considered trivial initial random guesses that do not make use of any previously obtained solution. The four considered types of random initial guesses differ on the domain within which the circles' variables are generated. Type R1 corresponds to random $(u_i, v_i, s_i)^T \in [-a, -b, 0] \times [a, b, 1]$, type R2 corresponds to random $(u_i, v_i, s_i)^T \in [-\frac{1}{2}a, -\frac{1}{2}b, 0] \times [\frac{1}{2}a, \frac{1}{2}b, 1]$, type R3 corresponds to random (u_i, v_i) satisfying $(u_i/a)^2 + (v_i/b)^2 = 1$ and $s_i \in [0, 1]$, and type R4 corresponds to random $(u_i, v_i, s_i)^T$ satisfying $(u_i/a)^2 + (v_i/b)^2 = 1$ and $s_i \in [0, \frac{1}{2}]$. These four types of random initial guesses are depicted on Figure 3. As it is well known, none of the initial guesses' strategies provides uniformly distributed circles. If required, uniformly distributed circles could be obtained by generating uniformly distributed random $(x_i, y_i) \in [-a, -b] \times [a, b]$ and then paying the extra cost of solving the (three-unknowns) feasibility problem (21) to find (u_i, v_i, s_i).

The effect of using the four described types of initial guesses was evaluated by running Algorithm 4.1. At Step 1, the lower bound m_{lb} was computed as described in Subsection 4.1. At Step 2, we arbitrarily fixed a CPU time limit of six hours. At Step 3, the four different strategies R1, R2, R3, and R4 were considered for generating the random initial guess. At Step 4 of Algorithm 4.1, approximate stationary points, candidates to global minimizers of the bound-constrained problem (20), were computed using Algencan [1, 2]. Default parameters of Algencan were used, i.e. approximate stationary points (u^*, v^*, s^*) satisfy the condition

$$\|P_\Omega[(u^*, v^*, s^*) - \nabla f^\xi(u^*, v^*, s^*)] - (u^*, v^*, s^*)\|_\infty \leq 10^{-8}, \tag{22}$$

where $\Omega = \{(u, v, s) \in \mathbb{R}^{3m} \mid 0 \leq s \leq 1\}$ represents the feasible region of problem (20) and $P_\Omega(u, v, s)$ denotes the orthogonal Euclidean projection of (u, v, s) onto the convex set Ω. An approximate stationary point (u^*, v^*, s^*) is considered a global minimizer of (20) at which the objective function $f^\xi(u^*, v^*, s^*)$ vanishes (Step 5 of Algorithm 4.1) whenever

$$\max\{\|h^\xi(u^*, v^*, s^*)\|_\infty, \|g^\xi(u^*, v^*, s^*)_+\|_\infty, \|\kappa^\xi(u^*, v^*, s^*)_+\|_\infty, \} \leq 10^{-4}, \tag{23}$$

i.e. when the sup-norm of the infeasibility violation of feasibility problem (18) is not greater than 10^{-4}. On the other hand, since $f^\xi(u, v, s)$ is the squared Euclidean norm of the infeasibility violation, numerical experiments have shown that, in most of the cases, the stopping criterion (22) implies

$$f^\xi(u^*, v^*, s^*) \approx 10^{-16} \tag{24}$$

and

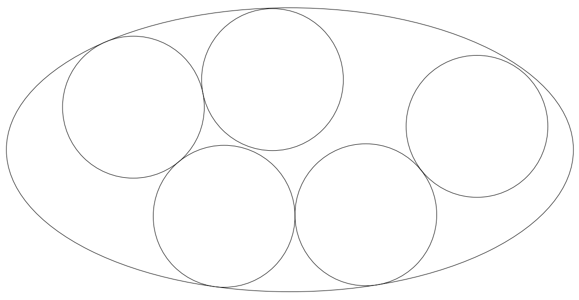
$$\max\{\|h^\xi(u^*, v^*, s^*)\|_\infty, \|g^\xi(u^*, v^*, s^*)_+\|_\infty, \|\kappa^\xi(u^*, v^*, s^*)_+\|_\infty\} \approx 10^{-8}. \tag{25}$$

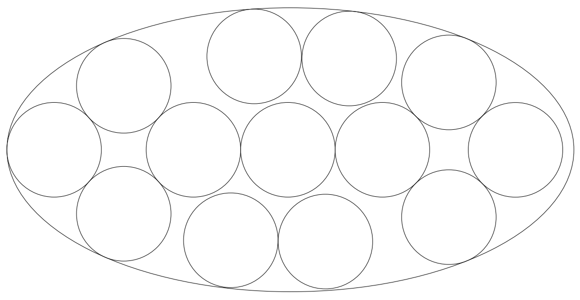
Values of (24) and (25) are reported in the numerical experiments. First derivatives and the sparse Hessian matrix of $f^\xi(u, v, s)$ were coded by hand. Moreover, the strategy described in [10] to reduce the complexity of evaluating the objective function and its derivatives was used. Codes are available for download at *http://www.ime.usp.br/∼egbirgin/packing/*.

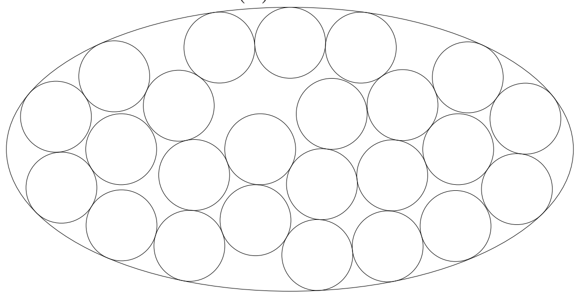
As expected, the plain multistart strategy delivered very poor results. When using initial guess types R1, R2, and R3, Algorithm 4.1 was capable of improving the number of packed circles given by the lattice-based lower bounds in a few number of instances, being in all the instances outperformed by Algorithm 4.1 with type R4 of random initial guesses. Moreover, as suggested in the last paragraph of Subsection 4.1, Algorithm 4.1 with type R4 of random initial guesses was only capable of improving solutions to instances with a small number of circles (smaller instances of Set 1) or instances for which the lattice-based lower bound is very weak (most instances with flat ellipses from Set 3). Table 2 shows the results of Algorithm 4.1 combined with random initial guesses of type R4. In the table, the first three columns identify the instances, including their computed lower bounds. Column m shows the number of circles packed. Columns f^ξ and $\|c^\xi\|_\infty$ correspond to the quantities in (24) and (25), respectively, and quantify the precision of the computed solutions. In column #IG (number of initial guesses), A(B) reports the number of random initial guesses, meaning that, to pack $m_{lb}+1, \ldots, m$ circles, Steps 3 and 4 of Algorithm 4.1 were executed A times and that from those A executions B were used to pack exactly m circles. Column named "CPU Time" shows the CPU time in seconds elapsed from the beginning until the solution with m circles was found. The last column, #AIG (additional initial guesses), shows the number of unsuccessful trials of packing $m + 1$ circles that were done in the remaining time until achieving the imposed CPU time limit of six hours. Figures 4 and 5 show the graphical representation of the obtained solutions that improved the solutions given by the lower bound strategy. The graphical representations suggest that some of those solutions that improved the lower bounds seem to be far from being optimal.

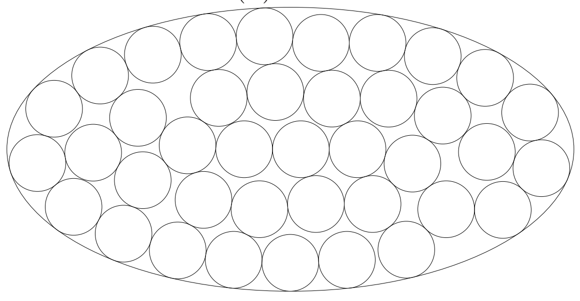
Instance data			Solution found and computational effort						Extra effort
	Name	m_{lb}	m	f^{ξ}	$\|c^{\xi}\|_{\infty}$	#IG		CPU Time	#AIG
Set 1	s1i01	4	5	6.2D−18	2.1D−09	5	(5)	0.00	18382854
	s1i02	12	13	1.7D−16	1.5D−08	33	(33)	0.14	3083185
	s1i03	20	24	7.7D−17	9.6D−09	202	(26)	4.13	652782
	s1i04	36	39	1.3D−16	1.0D−08	36082	(35485)	4935.44	100794
	s1i05	51	56	2.4D−17	5.2D−09	13280	(4796)	3248.33	55003
	s1i06	74	76	7.7D−16	2.0D−08	31888	(24132)	20415.82	1522
	s1i07	95	97	7.6D−17	7.1D−09	13407	(8410)	18193.36	2086
	s1i08	126	—	—	—	—	—	—	6364
	s1i09	159	—	—	—	—	—	—	3038
	s1i10	192	—	—	—	—	—	—	2074
	s1i11	238	—	—	—	—	—	—	573
	s1i12	273	—	—	—	—	—	—	669
	s1i13	320	—	—	—	—	—	—	364
	s1i14	374	—	—	—	—	—	—	198
Set 2	s2i01	369	—	—	—	—	—	—	178
	s2i02	367	—	—	—	—	—	—	313
	s2i03	363	—	—	—	—	—	—	269
	s2i04	364	—	—	—	—	—	—	269
	s2i05	361	—	—	—	—	—	—	224
	s2i06	354	—	—	—	—	—	—	229
	s2i07	346	—	—	—	—	—	—	387
	s2i08	341	—	—	—	—	—	—	605
	s2i09	331	—	—	—	—	—	—	802
	s2i10	323	—	—	—	—	—	—	1533
	s2i11	296	—	—	—	—	—	—	2117
	s2i12	265	—	—	—	—	—	—	1913
Set 3	s3i01	87	98	2.9D−11	4.1D−06	34507	(1455)	19705.80	260
	s3i02	56	60	5.2D−16	8.6D−09	1641	(1610)	245.50	12563
	s3i03	107	109	3.2D−16	8.4D−09	18872	(9436)	12838.60	1217
	s3i04	49	68	3.6D−16	1.8D−08	19419	(6254)	5888.90	4109
	s3i05	128	—	—	—	—	—	—	2374
	s3i06	59	71	1.3D−16	1.0D−08	59180	(33678)	14992.50	2295

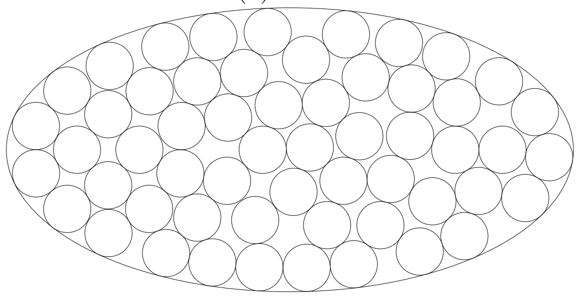
Table 2: Numerical results of applying Algorithm 4.1 combined with initial points of type R4.

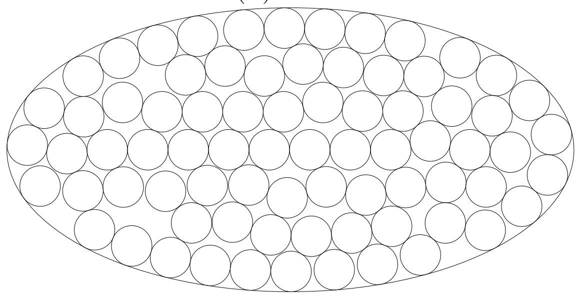

(a) s1i01


(b) s1i02


(c) s1i03


(d) s1i04


(e) s1i05


(f) s1i06

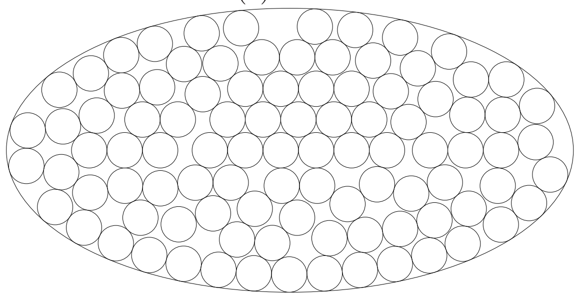
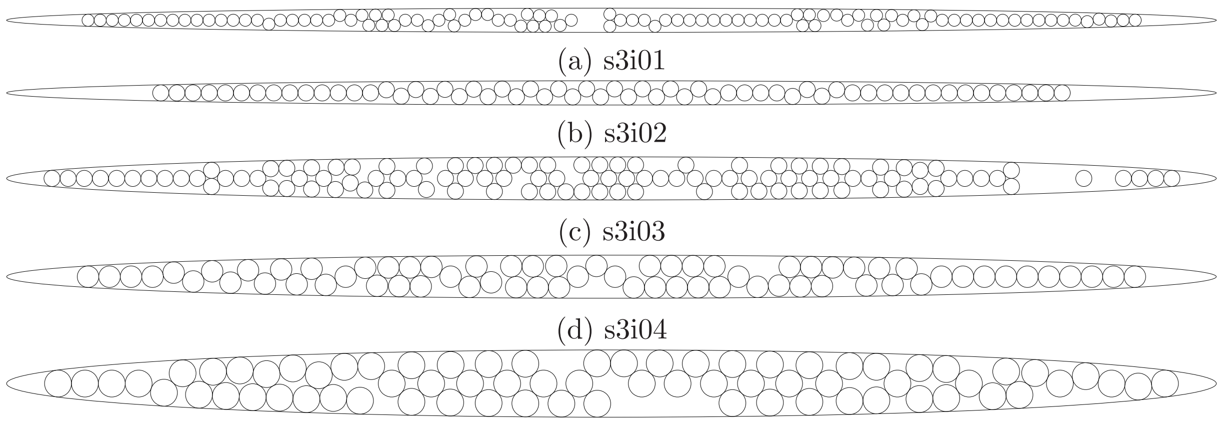

(g) s1i07

Figure 4: Graphical representation of the solutions obtained by the plain multistart version of Algorithm 4.1 to instances of Set 1 that improved the solution given by the lower bound.


(a) s3i01

(b) s3i02

(c) s3i03

(d) s3i04

(e) s3i06

Figure 5: Graphical representation of the solutions obtained by the plain multistart version of Algorithm 4.1 to instances of Set 3 that improved the solution given by the lower bound.

5.3 Additional alternatives for multistart initial guesses

The strategies described in the previous subsection to generate random initial guesses to pack, say, $\bar{m}$ circles do not make use of the known solution for packing $\bar{m} - 1$. That decision had a clear motivation: to be able to rapidly reproduce a given result (solution) without having to reproduce the whole experiment from the beginning. However, due to the poor performance of such strategy, we describe in the present subsection additional strategies to generate initial guesses that do make use of those previously computed solutions. The three considered strategies are:

M1: The initial guess to pack $\bar{m}$ circles always consists on the known solution for packing $\bar{m}-1$ circles plus random initial values generated as in strategy R4 for the three unknowns associated with the additional circle.

M2: The first trial of this strategy is as in strategy M1. However, when an undesired stationary point for packing $\bar{m}$ circles is found, new random values are given (as in strategy R4) for the "most infeasible circle", preserving the other values of the stationary point. This modified stationary point is used as initial guess for the next trial.

M3: This strategy is as in M2, but the variables of a random circle are modified instead of the ones associated with the most infeasible circle.

Note that strategy M2 tries to resemble a tunneling strategy, although an improvement in the objective function at the new random guess with respect to the current stationary point is not required at all. Tables 3–5 show the results of strategies M1–M3, respectively. In the tables, values of m in bold represent the best obtained results among strategies M1–M3. Figures 6–10 illustrate the best solutions found by any of the three strategies. Figures in the tables show that strategies M1, M2, and M3 found the best solutions (among the solutions found by the three strategies) in 17, 27, and 15 instances (out of 32), respectively. Figures also show that, if strategy M3 is ignored, all best solutions are still being found by strategies M1 or M2, and that, between M1 and M2, none of them is better than the other.

As strategy M2 presented the best performance, we run strategy M2 with an increased CPU time limit of 24 hours. We name this long run of strategy M2 as M2$^+$ from now on. Table 6 shows the figures of the seven instances in which M2$^+$ was able to improve the best solutions found. Figure 11 illustrates those improved solutions.

5.4 Summing up the best known results

All along the present section, we described several attempts to find a global solution to the 32 instances in Sets 1–3. At this point it is important to stress that the obtained solutions satisfy the precision requirements (22) and (23) and that, clearly, "better" solutions would be found by relaxing those requirements as well as some of the presented solutions would not be considered solutions under more rigorous tolerances. Independently of the optimization tolerances requirements (22) and (23), we also reported, for each presented solution, the actual value of f^ξ and $\|c^\xi\|_\infty$ in (24) and (25), respectively. However, those values combine two types of quantities that might be reported in separate. Those two quantities are:

Absolute maximum overlapping violation: strongly related to κ_{ij}^ξ in (19) and given by

$$\max_{i<j} \left\{ \left[2r - \sqrt{\{(1+(s_i-1)(b^2/a^2))u_i - (1+(s_j-1)(b^2/a^2))u_j\}^2 + \{s_i v_i - s_j v_j\}^2} \right]_+ \right\}. \tag{26}$$

Absolute maximum container violation: strongly related to g_i^ξ in (19) and given by

$$\max_{i} \left\{ \left[r - (1-s_i)\sqrt{(b^2/a^2)^2 u_i^2 + v_i^2} \right]_+ \right\}. \tag{27}$$

Every pair of circles' centers must be at a distance no smaller that $2r$ and (26) gives the largest absolute violation of this limit. Every circle must be inside the ellipse and at a distance not smaller than r to the ellipse's border. The largest absolute violation of this limit is given by (27). In fact, the exact calculation of the distance to the ellipse's border in (27) depends on the fulfillment of $h_i^\xi(u,v,s) = 0$ in (19). Therefore, the expression in (27) is an approximation of the maximum violation of the ellipse's border.

Table 7 summarizes these results pointing to the illustration of each solution within the text. In addition to the solutions' information described above, the solutions' density is also provided in the table. Detailed information of each solution (i.e. circles' centers) can be found at *http://www.ime.usp.br/∼egbirgin/packing/*. To develop a more efficient approach also capable of finding by itself all the best known solutions reported here (within the prescribed tolerances), and maybe even better solutions, remains an open problem.

Instance data			Solution found and computational effort						Extra effort
	Name	m_{lb}	m	f^{ξ}	$\|c^{\xi}\|_{\infty}$	#IG		CPU Time	#AIG
Set 1	s1i01	4	**5**	2.7D−18	2.3D−09	6	(6)	0.00	23460130
	s1i02	12	**13**	1.0D−16	9.8D−09	720	(720)	2.11	4659750
	s1i03	20	**24**	9.6D−17	1.1D−08	4	(1)	0.06	708338
	s1i04	36	**39**	2.9D−16	1.1D−08	158	(151)	18.13	282306
	s1i05	51	**57**	1.6D−16	1.1D−08	301	(296)	89.62	95656
	s1i06	74	**78**	1.2D−15	2.2D−08	209	(198)	162.08	37045
	s1i07	95	**103**	2.3D−12	9.9D−07	15	(8)	24.61	18221
	s1i08	126	**132**	9.6D−17	6.7D−09	87	(5)	274.92	9587
	s1i09	159	**164**	1.1D−17	1.7D−09	92	(84)	488.65	4862
	s1i10	192	198	3.8D−09	3.1D−05	135	(128)	808.40	2233
	s1i11	238	**239**	3.9D−16	1.4D−08	2	(2)	42.43	1351
	s1i12	273	**281**	2.2D−17	2.5D−09	77	(70)	2002.26	915
	s1i13	320	327	1.3D−16	6.1D−09	21	(13)	887.23	517
	s1i14	374	**378**	5.2D−17	5.5D−09	13	(6)	960.37	307
Set 2	s2i01	369	376	2.0D−17	3.6D−09	20	(14)	1220.52	383
	s2i02	367	**377**	6.1D−17	5.5D−09	17	(8)	871.21	353
	s2i03	363	**376**	1.2D−16	7.4D−09	28	(6)	1735.57	341
	s2i04	364	**378**	3.2D−16	1.3D−08	339	(224)	19106.20	53
	s2i05	361	375	2.1D−15	4.3D−08	80	(37)	3717.94	422
	s2i06	354	371	6.9D−16	2.2D−08	550	(234)	18711.12	91
	s2i07	346	373	2.8D−15	5.0D−08	239	(204)	10490.40	313
	s2i08	341	366	2.9D−11	4.8D−06	189	(59)	6061.91	884
	s2i09	331	**365**	7.7D−17	3.5D−09	297	(22)	6251.26	718
	s2i10	323	357	1.5D−15	3.1D−08	934	(157)	18528.08	199
	s2i11	296	**353**	5.1D−16	2.0D−08	540	(136)	7096.15	1112
	s2i12	265	324	1.8D−15	3.9D−08	2185	(242)	20997.28	147
Set 3	s3i01	87	118	1.4D−08	6.4D−05	18078	(9972)	10548.71	18630
	s3i02	56	57	2.0D−17	6.0D−09	5	(5)	0.13	605187
	s3i03	107	140	2.0D−14	1.9D−07	3136	(2333)	2414.27	25360
	s3i04	49	67	4.6D−16	2.7D−08	17826	(9775)	1919.16	169738
	s3i05	128	147	2.5D−16	9.5D−09	383	(150)	449.84	21535
	s3i06	59	77	1.8D−16	9.0D−09	361	(37)	61.32	129557

Table 3: Performance of multistart strategy M1.

Instance data			Solution found and computational effort						Extra effort
	Name	m_{lb}	m	f^{ξ}	$\|c^{\xi}\|_{\infty}$	#IG		CPU Time	#AIG
Set 1	s1i01	4	**5**	4.1D−19	7.8D−10	9	(9)	0.01	26068471
	s1i02	12	**13**	1.5D−16	1.5D−08	2	(2)	0.01	4774069
	s1i03	20	**24**	9.6D−17	1.1D−08	4	(1)	0.06	907210
	s1i04	36	**39**	2.7D−17	3.8D−09	49	(44)	5.06	261714
	s1i05	51	**57**	4.5D−17	9.1D−09	47	(42)	12.29	95777
	s1i06	74	**78**	7.2D−17	5.6D−09	8	(3)	10.23	37881
	s1i07	95	**103**	9.5D−17	7.6D−09	80	(73)	89.06	28480
	s1i08	126	**132**	1.0D−15	2.1D−08	313	(298)	762.19	10864
	s1i09	159	**164**	2.3D−16	1.2D−08	8	(3)	36.75	6807
	s1i10	192	**199**	1.8D−16	1.5D−08	44	(1)	359.77	4197
	s1i11	238	**239**	3.6D−16	2.3D−08	9	(9)	218.12	2823
	s1i12	273	280	2.8D−16	1.7D−08	7	(1)	98.31	2049
	s1i13	320	**328**	1.1D−09	2.1D−05	555	(543)	11780.04	583
	s1i14	374	**378**	1.7D−15	2.6D−08	52	(34)	3140.71	645
Set 2	s2i01	369	**377**	3.4D−16	1.2D−08	351	(342)	12567.15	348
	s2i02	367	376	3.4D−16	2.1D−08	9	(1)	160.42	1096
	s2i03	363	**376**	7.6D−16	1.9D−08	324	(291)	12559.34	284
	s2i04	364	**378**	7.0D−16	1.6D−08	390	(1)	11094.14	386
	s2i05	361	**379**	4.1D−16	1.4D−08	462	(250)	15327.75	280
	s2i06	354	**378**	3.8D−16	1.2D−08	278	(1)	8496.39	657
	s2i07	346	**375**	1.4D−09	2.0D−05	830	(548)	16983.64	284
	s2i08	341	**370**	3.2D−16	1.3D−08	227	(1)	6081.33	1118
	s2i09	331	363	4.1D−15	6.9D−08	733	(465)	8453.13	1269
	s2i10	323	**360**	3.2D−16	1.3D−08	1188	(521)	13996.47	899
	s2i11	296	350	2.6D−16	1.1D−08	1468	(1070)	9254.16	2355
	s2i12	265	**336**	1.1D−16	1.2D−08	5337	(4128)	20337.15	413
Set 3	s3i01	87	**139**	6.6D−16	1.1D−08	56217	(53195)	18557.22	10011
	s3i02	56	**61**	2.2D−17	2.7D−09	57007	(56811)	3050.48	443059
	s3i03	107	**141**	6.4D−15	1.1D−07	17797	(14492)	5538.99	53725
	s3i04	49	**78**	9.4D−20	4.0D−10	636	(105)	81.29	192575
	s3i05	128	**151**	3.6D−16	7.0D−09	3488	(2797)	2443.46	31326
	s3i06	59	**79**	4.4D−09	5.1D−05	8612	(8158)	835.34	240445

Table 4: Performance of multistart strategy M2.

Instance data			Solution found and computational effort						Extra effort
	Name	m_{lb}	m	f^{ξ}	$\|c^{\xi}\|_{\infty}$	#IG		CPU Time	#AIG
Set 1	s1i01	4	**5**	9.0D−19	7.9D−10	5	(5)	0.00	17482175
	s1i02	12	**13**	4.2D−17	7.7D−09	5	(5)	0.03	4059198
	s1i03	20	**24**	9.6D−17	1.1D−08	4	(1)	0.05	727197
	s1i04	36	**39**	1.5D−16	8.9D−09	26	(21)	2.24	195840
	s1i05	51	**57**	9.5D−16	3.2D−08	15	(10)	3.56	77726
	s1i06	74	**78**	1.3D−16	1.1D−08	16	(12)	20.14	29734
	s1i07	95	**103**	2.2D−16	1.0D−08	52	(45)	65.49	20173
	s1i08	126	**132**	2.9D−17	5.7D−09	243	(225)	713.19	7663
	s1i09	159	**164**	3.1D−16	1.2D−08	191	(124)	961.63	5963
	s1i10	192	**199**	3.7D−16	1.8D−08	214	(196)	1678.21	2906
	s1i11	238	238	—	—	—	—	—	2271
	s1i12	273	**281**	3.3D−16	1.1D−08	10	(3)	226.39	1906
	s1i13	320	327	1.8D−16	1.0D−08	291	(267)	7287.23	873
	s1i14	374	376	2.7D−15	3.3D−08	4	(3)	356.83	944
Set 2	s2i01	369	375	1.9D−15	5.7D−08	6	(1)	557.13	724
	s2i02	367	375	2.6D−16	1.7D−08	8	(1)	143.56	982
	s2i03	363	375	3.7D−16	1.8D−08	444	(178)	16654.77	162
	s2i04	364	377	1.8D−16	9.5D−09	162	(1)	7741.19	387
	s2i05	361	369	4.0D−17	6.8D−09	64	(1)	2913.40	328
	s2i06	354	373	4.6D−17	2.8D−09	327	(126)	12170.68	290
	s2i07	346	371	2.5D−16	1.2D−08	710	(261)	18706.98	160
	s2i08	341	366	1.0D−08	6.0D−05	836	(1)	17273.26	144
	s2i09	331	358	1.9D−15	5.6D−08	915	(132)	20169.86	140
	s2i10	323	357	4.1D−16	1.1D−08	2416	(106)	21095.57	57
	s2i11	296	345	2.6D−16	1.6D−08	1266	(1)	19803.39	321
	s2i12	265	331	5.1D−17	3.9D−09	2167	(1)	16411.58	801
Set 3	s3i01	87	134	5.7D−17	5.3D−09	23054	(15296)	7248.47	58554
	s3i02	56	**61**	8.0D−18	1.4D−09	348243	(348074)	16652.31	122688
	s3i03	107	**141**	8.5D−16	3.8D−08	7052	(3)	2530.15	50832
	s3i04	49	**78**	5.2D−17	5.0D−09	111922	(105478)	11339.10	97496
	s3i05	128	150	4.1D−16	9.9D−09	11719	(10614)	8760.11	15730
	s3i06	59	**79**	1.8D−16	1.7D−08	70963	(66584)	7388.61	148024

Table 5: Performance of multistart strategy M3.

Instance data			Solution found and computational effort						Extra effort
	Name	m_{lb}	m	f^{ξ}	$\|c^{\xi}\|_{\infty}$	#IG		CPU Time	#AIG
S1	s1i09	159	**165**	2.5D−16	9.8D−09	15158	(15150)	47815.29	14636
	s1i14	374	**379**	1.3D−16	8.6D−09	2307	(2255)	64242.21	897
S2	s2i01	369	**378**	2.8D−17	5.3D−09	778	(427)	24005.88	2668
	s2i04	364	**379**	4.1D−16	1.9D−08	883	(493)	24240.07	2842
	s2i12	265	**339**	8.2D−16	1.3D−08	21974	(1)	69153.22	6409
S3	s3i03	107	**142**	4.3D−16	8.2D−09	142408	(124611)	43021.63	145596
	s3i06	59	**80**	7.5D−17	5.7D−09	593092	(584480)	51709.06	80329

Table 6: Performance of multistart strategy M2 with a CPU time limit of 24 hours.

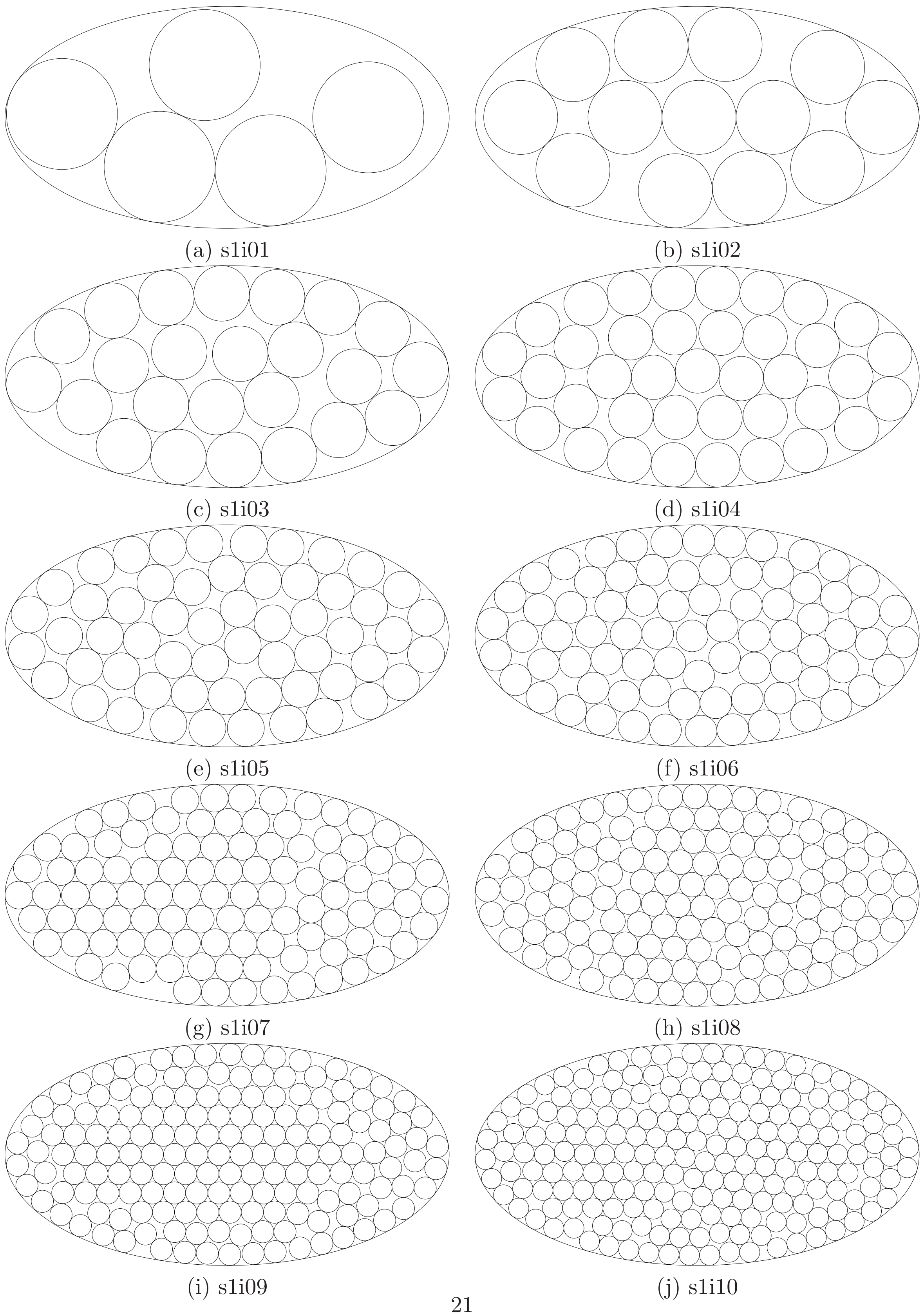
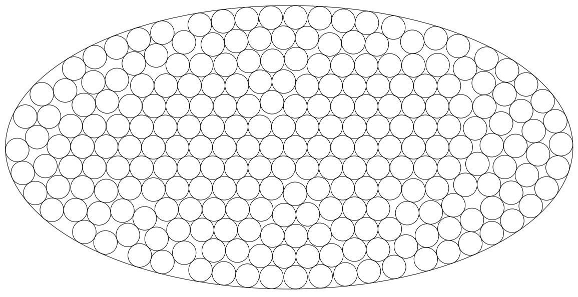
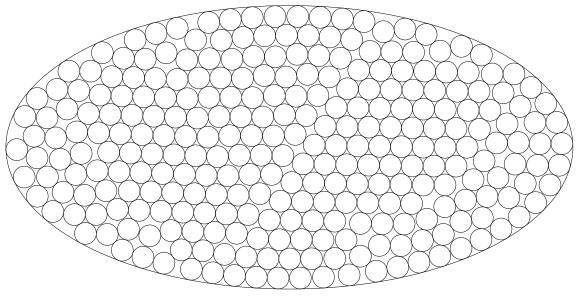

(a) s1i01 (b) s1i02 (c) s1i03 (d) s1i04 (e) s1i05 (f) s1i06 (g) s1i07 (h) s1i08 (i) s1i09 (j) s1i10

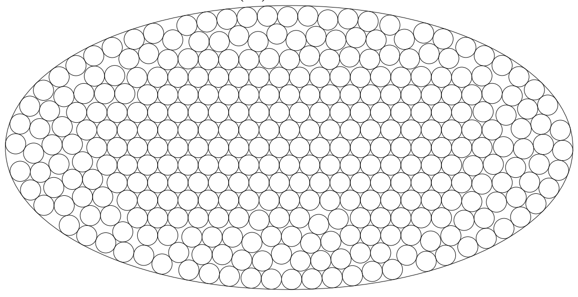
Figure 6: Graphical representation of the solutions obtained by strategies M1 and/or M2 to instances s1i01–s1i10 of Set 1.

Instance data		Solution found							
	Name	m	f^ξ	$\|c^\xi\|_\infty$	AMOV	AMCV	density	Strategy	Graphic
Set 1	s1i01	5	4.1D−19	7.8D−10	0.0D+00	0.0D+00	0.6250	R4/M1/M2/M3	Fig. 6(a)
	s1i02	13	1.5D−16	1.5D−08	4.8D−10	8.4D−11	0.7222	R4/M1/M2/M3	Fig. 6(b)
	s1i03	24	9.6D−17	1.1D−08	9.4D−10	1.1D−10	0.7500	R4/M1/M2/M3	Fig. 6(c)
	s1i04	39	2.7D−17	3.8D−09	2.8D−09	2.1D−10	0.7800	R4/M1/M2/M3	Fig. 6(d)
	s1i05	57	4.5D−17	9.1D−09	9.1D−09	8.8D−11	0.7917	M1/M2/M3	Fig. 6(e)
	s1i06	78	7.2D−17	5.6D−09	1.4D−09	2.6D−10	0.7959	M1/M2/M3	Fig. 6(f)
	s1i07	103	9.5D−17	7.6D−09	2.3D−09	3.8D−10	0.8047	M1/M2/M3	Fig. 6(g)
	s1i08	132	1.0D−15	2.1D−08	2.7D−09	6.4D−10	0.8148	M1/M2/M3	Fig. 6(h)
	s1i09	165	2.5D−16	9.8D−09	1.6D−09	3.5D−10	0.8250	M2$^+$	Fig. 11(a)
	s1i10	199	1.8D−16	1.5D−08	1.6D−09	3.2D−10	0.8223	M2/M3	Fig. 6(j)
	s1i11	239	3.6D−16	2.3D−08	2.1D−09	5.7D−10	0.8299	M1/M2	Fig. 7(k)
	s1i12	281	2.2D−17	2.5D−09	6.7D−10	1.5D−10	0.8314	M1/M3	Fig. 7(l)
	s1i13	328	1.1D−09	2.1D−05	2.3D−06	7.0D−07	0.8367	M2	Fig. 7(m)
	s1i14	379	1.3D−16	8.6D−09	1.0D−09	2.1D−10	0.8422	M2$^+$	Fig. 11(b)
Set 2	s2i01	378	2.8D−17	5.3D−09	1.4D−09	9.0D−11	0.8400	M2$^+$	Fig. 11(c)
	s2i02	377	6.1D−17	5.5D−09	8.9D−10	1.2D−10	0.8378	M1	Fig. 8(b)
	s2i03	376	7.6D−16	1.9D−08	1.3D−09	4.1D−10	0.8356	M1/M2	Fig. 8(c)
	s2i04	379	4.1D−16	1.9D−08	1.3D−09	2.5D−10	0.8422	M2$^+$	Fig. 11(d)
	s2i05	379	4.1D−16	1.4D−08	1.5D−09	3.6D−10	0.8422	M2	Fig. 8(e)
	s2i06	378	3.8D−16	1.2D−08	1.4D−09	3.1D−10	0.8400	M2	Fig. 8(f)
	s2i07	375	1.4D−09	2.0D−05	3.1D−06	7.6D−07	0.8333	M2	Fig. 8(g)
	s2i08	370	3.2D−16	1.3D−08	2.5D−09	5.7D−10	0.8222	M2	Fig. 8(h)
	s2i09	365	7.7D−17	3.5D−09	1.7D−09	2.4D−10	0.8111	M1	Fig. 9(i)
	s2i10	360	3.2D−16	1.3D−08	2.9D−09	1.2D−09	0.8000	M2	Fig. 9(j)
	s2i11	353	5.1D−16	2.0D−08	4.7D−09	1.4D−09	0.7844	M1	Fig. 9(k)
	s2i12	339	8.2D−16	1.3D−08	5.8D−09	2.2D−09	0.7533	M2$^+$	Fig. 11(e)
Set 3	s3i01	139	6.6D−16	1.1D−08	7.9D−09	2.7D−09	0.6950	M2	Fig. 10(a)
	s3i02	61	2.2D−17	2.7D−09	2.7D−09	6.6D−10	0.5422	M2/M3	Fig. 10(b)
	s3i03	142	4.3D−16	8.2D−09	4.4D−09	1.4D−09	0.7100	M2$^+$	Fig. 11(f)
	s3i04	78	9.4D−20	4.0D−10	0.0D+00	6.0D−12	0.6933	M2/M3	Fig. 10(d)
	s3i05	151	3.6D−16	7.0D−09	3.6D−09	1.0D−09	0.7550	M2	Fig. 10(e)
	s3i06	80	7.5D−17	5.7D−09	5.7D−09	9.8D−10	0.7111	M2$^+$	Fig. 11(g)

Table 7: Summary of the best known solutions.


(k) s1i11


(l) s1i12


(m) s1i13

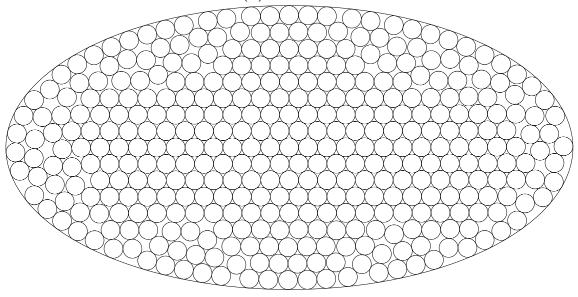

(n) s1i14

Figure 7: Graphical representation of the solutions obtained by strategies M1 and/or M2 to instances s1i11–s1i14 of Set 1.

6 Conclusions

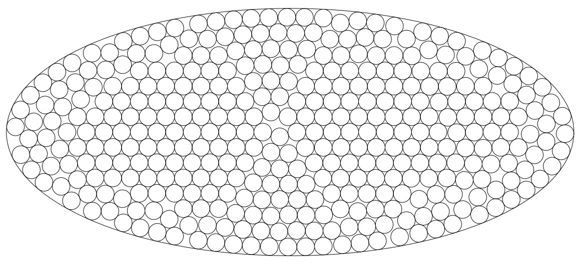
We introduced a new parametrization and optimization procedures for handling problems that involve packing balls within ellipses. We were able to handle instances with several hundreds of balls for the problem of packing the maximum number of unitary balls within a given ellipse. For instances with even more items, lattice-based solutions seem to provide high-quality approximations, pointing out that optimization procedures might not be the most adequate approach. We claim that careful analysis and experimentation with optimization procedures and heuristics for solving packing problems of increasing complexity should help to solve complicated real-life problems whose complicating characteristic could be irregularity of objects and domains.

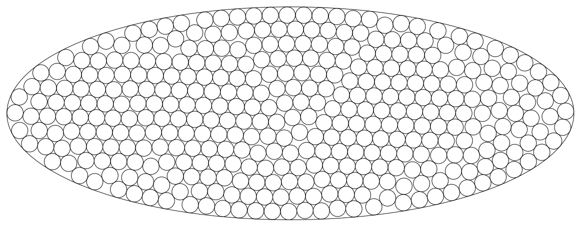
Acknowledgements. We are indebted to two annonymous referees for their helpful comments.

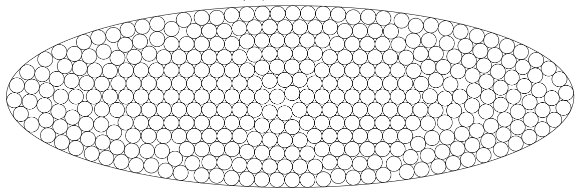
References

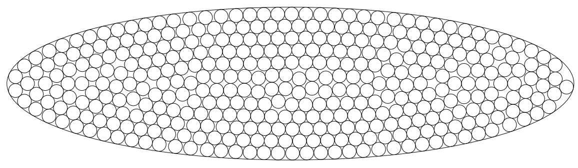
[1] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt, On Augmented Lagrangian Methods with general lower-level constraints, *SIAM Journal on Optimization* 18, pp. 1286–1309, 2007.

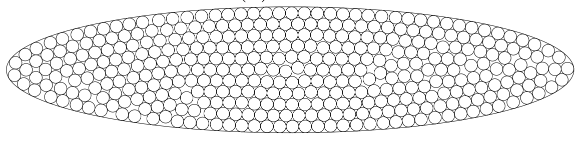
[2] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt, Augmented Lagrangian methods under the Constant Positive Linear Dependence constraint qualification, *Mathe-*

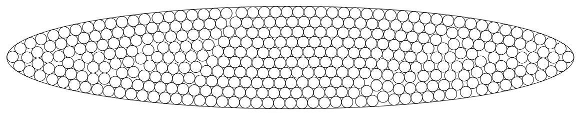

(a) s2i01

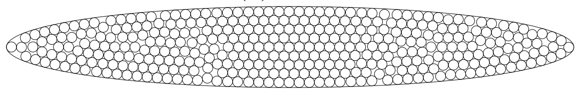

(b) s2i02


(c) s2i03


(d) s2i04


(e) s2i05


(f) s2i06


(g) s2i07

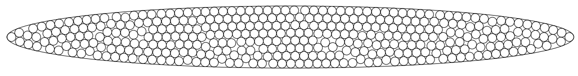
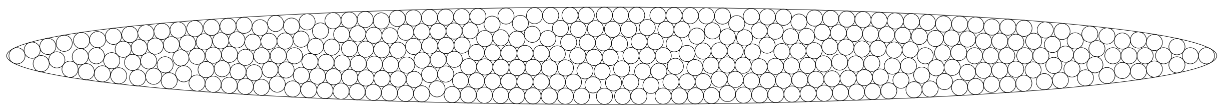
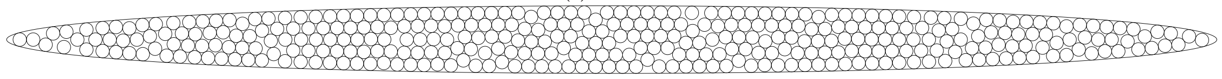
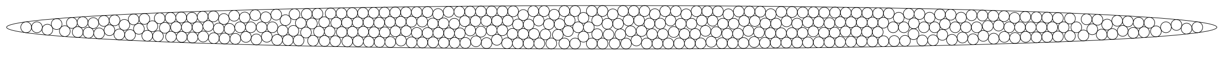

(h) s2i08

Figure 8: Graphical representation of the solutions obtained by strategies M1 and/or M2 to instances s2i01–s2i08 of Set 2.


(i) s2i09


(j) s2i10


(k) s2i11

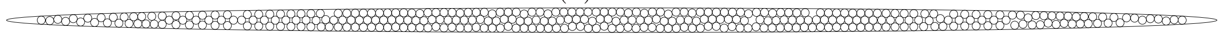

(l) s2i12

Figure 9: Graphical representation of the solutions obtained by strategies M1 and/or M2 to instances s2i09–s2i12 of Set 2.

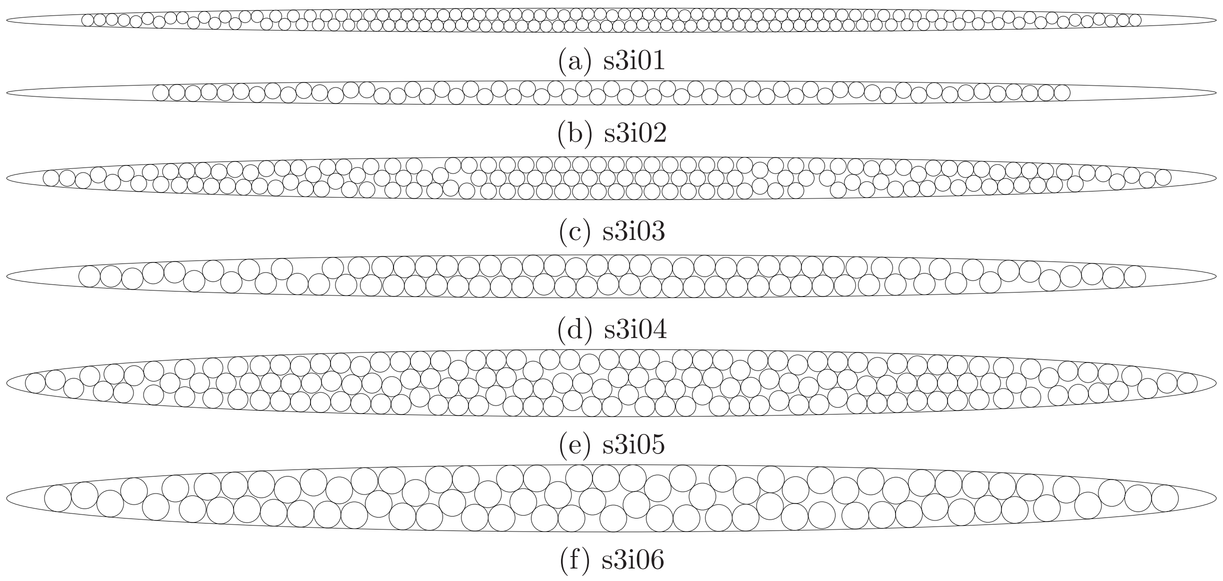

(a) s3i01
(b) s3i02
(c) s3i03
(d) s3i04
(e) s3i05
(f) s3i06

Figure 10: Graphical representation of the solutions obtained by strategies M1 and/or M2 to instances of Set 3.

matical Programming 111, pp. 5–32, 2008.

[3] C. Barron, S. Gómez, and D. Romero, Archimedean polyhedron structure yields a lower energy atomic cluster, *Applied Mathematics Letters* 9, pp. 75–78, 1996.

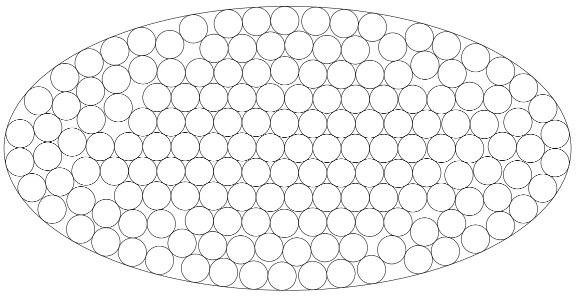
[4] E. G. Birgin, C. A. Floudas, and J. M. Martínez, Global minimization using an Augmented Lagrangian method with variable lower-level constraints, *Mathematical Programming* 125, pp. 139–162, 2010.

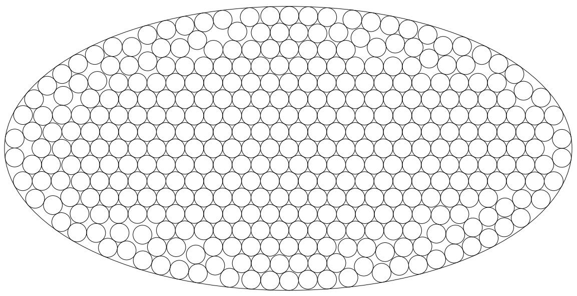
[5] E. G. Birgin and J. M. Gentil, New and improved results for packing identical unitary radius circles within triangles, rectangles and strips, *Computers & Operations Research* 37, pp. 1318–1327, 2010.

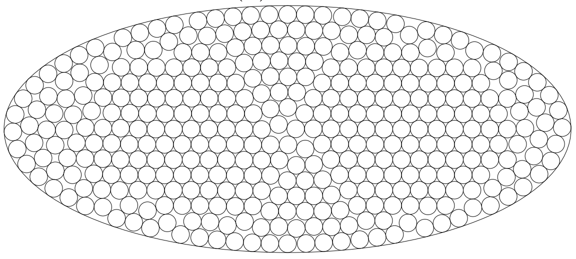
[6] E. G. Birgin and R. D. Lobato, Orthogonal packing of identical rectangles within isotropic convex regions, *Computers & Industrial Engineering* 59, pp. 595–602, 2010.

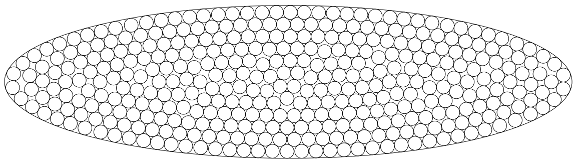
[7] E. G. Birgin and J. M. Martínez, Large-scale active-set box-constrained optimization method with spectral projected gradients, *Computational Optimization and Applications* 23, pp. 101–125, 2002.

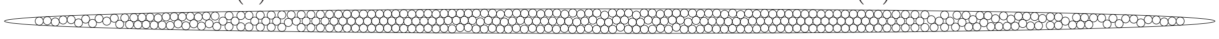
[8] E. G. Birgin, J. M. Martínez, F. H. Nishihara, and D. P. Ronconi, Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization, *Computers & Operations Research* 33, pp. 3535–3548, 2006.

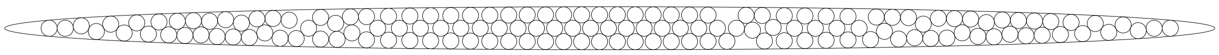

(a) s1i09


(b) s1i14


(c) s2i01


(d) s2i04


(e) s2i12


(f) s3i03

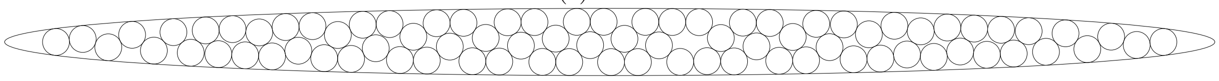

(g) s3i06

Figure 11: Graphical representation of the solutions obtained by strategy M2 using a 24 hours' CPU time limit that improved the best known solutions obtained by strategies M1–M3 using a CPU time limit of 6 hours.

[9] E. G. Birgin, J. M. Martínez, and D. P. Ronconi, Optimizing the Packing of Cylinders into a Rectangular Container: A Nonlinear Approach, *European Journal of Operational Research* 160, pp. 19–33, 2005.

[10] E. G. Birgin and F. N. C. Sobral, Minimizing the object dimensions in circle and sphere packing problems, *Computers & Operations Research* 35, pp. 2357–2375, 2008.

[11] L. H. Bustamante, *Stochastic global optimization strategies for packing circles within ellipses*, Ms Dissertation, Institute of Mathematics and Statistics, University of São Paulo, 2012.

[12] D. W. Cantrell, Private Communication, 2011.

[13] I. Castillo, F. J. Kampas, and J. D. Pinter, Solving circle packing problems by global optimization: Numerical results and industrial applications, *European Journal of Operational Research* 191, pp. 786–802, 2008.

[14] J. H. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups*, Springer-Verlag, New York, Berlin, Heidelberg, London, Paris, Tokyo, 1988.

[15] H. F. Callisaya, *Packing on Quadrics*, PhD Thesis, Institute of Mathematics, Statistics and Scientific Computing, University of Campinas, 2012.

[16] E. Friedman, *Erich's Packing Center*, http://www2.stetson.edu/∼efriedma/packing.html

[17] R. W. Hockney and J. W. Eastwood, *Computer Simulation using Particles*, McGraw Hill, New York, 1981.

[18] C. M. Hogan, Abiotic Factor, in *Encyclopedia of Earth*, E. Monosson and C. J. Cleveland (eds.), Environmental Information Coalition, National Council for Science and the Environment, Washington, DC, 2010.

[19] J. Kallrath, Cutting circles and polygons from area-minimizing rectangles, *Journal of Global Optimization* 43, pp. 299–328, 2009.

[20] A. V. Levy and A. Montalvo, The Tunneling Algorithm for the global minimization of functions, *SIAM Journal on Scientific and Statistical Computing* 6, pp. 15–29, 1985.

[21] L. Martínez, R. Andrade, E. G. Birgin, and J. M. Martínez, Packmol: A package for building initial configurations for molecular dynamics simulations, *Journal of Computational Chemistry* 30, pp. 2157–2164, 2009.

[22] J. M. Martínez and L. Martínez, Packing optimization for automated generation of complex system's initial configurations for molecular dynamics and docking, *Journal of Computational Chemistry* 24, pp. 819–825, 2003.

[23] D. V. Nichita and S. Gómez, Efficient location of multiple global minima for the phase stability problem, *Chemical Engineering Journal* 152, pp. 251–263, 2009.

[24] M. Srinivas, Implementation and evaluation of random tunneling algorithm for chemical engineering applications, *Computers and Chemical Engineering* 9, pp. 1400–1414, 2006.

[25] Y. G. Stoyan and G. Yaskov, A mathematical model and a solution method for the problem of placing various-sized circles into a strip, *European Journal of Operational Research* 156, pp. 590–600, 2004.

[26] Y. G. Stoyan and G. Yaskov, Packing equal circles into a circle with circular prohibited areas, *Interbational Journal of Computer Mathematics* 89, pp. 1355–1369, 2012.

[27] Y. G. Stoyan and G. Yaskov, Packing congruent hyperspheres into a hypersphere, *Journal of Global Optimization* 52, pp. 855–868, 2012.

[28] Y. G. Stoyan, M. V. Zlotnik, and A. M. Chugay, Solving an optimization packing problem of circles and non-convex polygons with rotations into a multiply connected region, *Journal of the Operational Research Society* 63, pp. 379–391, 2012.

[29] E. W. Weisstein, *Parallel Curves*, from MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/ParallelCurves.html

[30] E. W. Weisstein, *Circle Packing*, from MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/CirclePacking.html