Minimizing the object dimensions in circle and sphere packing problems

E. G. Birgin * F. N. C. Sobral †

October 12, 2006

Abstract

Given a fixed set of identical or different-sized circular items, the problem we deal with consists on finding the smallest object within which the items can be packed. Circular, triangular, squared, rectangular and also strip objects are considered. Moreover, 2D and 3D problems are treated. Twice-differentiable models for all these problems are presented. A strategy to reduce the complexity of evaluating the models is employed and, as a consequence, instances with a large number of items can be considered. Numerical experiments show the flexibility and reliability of the new unified approach.

Key words: Packing of circles and spheres, models, algorithms, nonlinear programming.

1 Introduction

Several papers deal with the packing problem of, given an object and a fixed set of items, minimize the object dimension (or equivalently maximize the items dimension) subject to placing the items within the object without overlapping. In particular, several papers focus on this problem using nonlinear optimization models and techniques.

In [26] the problem of packing identical circular items within a square is treated. A max-min model to maximize the minimum distance between the centers of the circular items is introduced with the aim of maximizing the items diameter. A classical nonlinear reformulation of the max-min problem is solved using MINOS [30] and the GAMS modeling language [17]. A multi-start strategy is used to enhance the probability of finding global solutions. Problems up to 30 items are solved attaining the best known solutions reported in the literature and yielding some new configurations. A different nonlinear programming formulation of the same problem is also introduced in [31]. Basically, the energy function $\sum_{i \neq j} (\lambda / d_{ij}^2)^m$, where d_{ij} represents

*Department of Computer Science, IME, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brazil. This author was supported by PRONEX-CNPq/FAPERJ E-26/171.164/2003 - APQ1, CNPq (PROSUL 490333/2004-4) and FAPESP (Grants 03/09169-6 and 01/04597-4). Corresponding author. FAX: +55(11)3091-6134. e-mail: egbirgin@ime.usp.br

†Department of Computer Science, IME, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brazil. This author was supported by CNPq. e-mail: fsobral@ime.usp.br

the Euclidean distance between the centers of items i and j, λ is a scaling factor and m is a positive integer, is minimized subject to the items fitting inside the object. An unconstrained reformulation of that problem is solved using an hybrid line-search algorithm that uses gradient-type directions at the beginning and Newton-type directions near the solution. The solution obtained by the optimization algorithm is then used as a starting point to solve a system of nonlinear equations with the aim of improving the accuracy of the solution. Problems with up to 50 items are solved, finding some alternative solutions and improving some results presented in [18, 21]. In [25], the same problem is studied. The problem is modeled as a quadratic optimization problem. Two properties satisfied by at least an optimal solution are introduced and a clever and efficient branch-and-bound algorithm is developed. The algorithm is used to prove the ϵ-optimality of solutions with up to 35 items, 38 and 39 items. Moreover, new solutions for 32 and 37 items are found. Finally, the authors of [25] highlight that, concerning the problem of maximizing the items diameter (instead of minimizing the object size), while most of the approaches are capable of obtaining a lower bound, their approach provides an upper bound on the solution.

The problem of packing equal circles within a circle is addressed in [29]. The nonlinear formulation is one of the most natural ones and maximizes the items radius while requires that the items do not overlap and fit inside the circular object. Two equivalent formulations, using cartesian and polar coordinates, are presented. A heuristic method called Reformulation Descent that iteratively alternates from one model to the other is presented and tested. The authors claim that these switches may reduce the probability of the method to stop at undesired stationary points. Numerical experiments show that the method is much faster than classical nonlinear programming methods and that, for instances with up to 100 items, the best known solution is found in 40% of the cases while in the other cases the error never exceeds 1%.

The problem of packing different-sized circles into a strip is considered in [35]. A natural nonlinear model is introduced and deeply studied. As pointed out by the authors, some peculiarities of the model allow to conclude that optimal solutions are attained at extreme points of the feasible region. A clever approach (based on Lagrange multipliers) to jump from a local minimizer to a better one, interchanging the positions of two non-identical circles, is devised. The numerical experiments in [35] show that this strategy, that can be viewed as a tunneling method for escaping from local minimizers, together with an efficient local solver [34] and a multi-start strategy, yields high-quality solutions. A comparison against a branch-and-bound algorithm using instances with up to 35 items show the advantages of the presented approach. Similar techniques are considered in [36], where the problem of packing various solid spheres into a parallelepiped with minimal height is addressed and numerical results with up to 60 spheres are given.

Nonlinear models have also been successfully used in the problem of, given an infinite number of identical items and an object, all with fixed dimensions, pack as many items as possible within the object. In [15] the authors deal with circular items within rectangular and circular containers. In [12, 11] the problems of packing orthogonal rectangles and free-rotated rectangles within arbitrary convex regions are considered, respectively. The Method of Sentinels introduced in [11, 28] can also be used for packing arbitrary non-identical polygons (with internal angles not smaller than $\pi/2$) within arbitrary convex regions. In all the cases the nonlinear models are solved using ALGENCAN [2, 1, 7, 10], an Augmented Lagrangian method for the minimization of a smooth function with general constraints (freely available at the TANGO Project web page [6]).

All the nonlinear-model-based strategies described above have a point in common: the number of non-overlapping constraints between the items is $O(N^2)$, where N is the number of items being packed. Based on efficient algorithms developed to reduce the asymptotic computational complexity of the N-body problem [22], we develop a methodology (and their corresponding data structures) to reduce the computational complexity of evaluating the nonlinear models. By using this strategy, packing problems with a large number of items are solved. See [27], where a similar strategy was successfully used for the generation of initial configurations for molecular dynamics.

This paper is organized as follows. In Section 2 we introduce a variety of nonlinear models for packing circles and spheres in several kinds of objects. The strategy to reduce the complexity of the non-overlapping constraints is described and analyzed. Numerical experiments are shown in Section 3. In Section 4 we state some conclusions and directions for future research.

2 Nonlinear models

The nonlinear models considered in this paper have the following structure (see [33]):

$$\begin{array}{ll} \text{Minimize} & \text{the object dimension} \\ \text{subject to} & \text{fitting the items inside the object,} \\ & \text{non-overlapping of items.} \end{array} \tag{1}$$

Problem (1) has three main ingredients: (i) the object dimension to be minimized; (ii) the constraints of the items being placed within the object; and (iii) the non-overlapping constraint between the items. Clearly, the first one depends just on the object and the last one depends just on the items, while the other one depends on both, the object and the items. In the present section we will show how to deal with each model ingredient in the cases of circular or spherical items and a wide range of objects.

2.1 Non-overlapping constraints

In this subsection we describe the non-overlapping constraints. Moreover, we describe a methodology, based on strategies developed for the N-body problem [22], to reduce the complexity of computing the non-overlapping constraints.

Let $c_i, i = 1, \dots, N$, be the centers of N circular or spherical items with radii $r_i, i = 1, \dots, N$. The non-overlapping between the items can be modeled as

$$d(c_i, c_j)^2 \geq (r_i + r_j)^2, \; \forall \; i < j, \tag{2}$$

or

$$\sum_{i<j} \max\{0, (r_i + r_j)^2 - d(c_i, c_j)^2\}^2 = 0, \tag{3}$$

where $d(\cdot, \cdot)$ is the Euclidean distance (see Figure 1). Both (2) and (3) are related to the distances among the centers of the $N(N-1)/2$ pairs of items. Squares in (2) and (3) are used to make

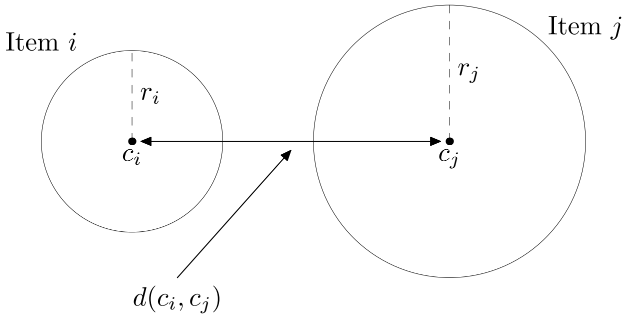


Figure 1: Two items do not overlap if and only if the distance among their centers is greater than or equal to the sum of their radii.

them differentiable. Clearly, (2) and (3) are equivalent and any set of items that satisfy (2) or (3) has no overlapping.

However, (3) has a great advantage over (2) in practice: not all the $O(N^2)$ distances need to be computed to evaluate the left hand side of the constraint at a given point. Basically, if two circular items k_1 and k_2 are far one from the other, the contribution of the term $\max\{0, (r_{k_1} + r_{k_2})^2 - d(c_{k_1}, c_{k_2})^2\}^2$ to the sum in (3) is null. In fact, if the items are more or less well distributed, the number of pairs that contribute to the sum in (3) is $O(N)$. Moreover, these pairs can also be identified in $O(N)$ operations. This strategy has been successfully used in molecular conformation problems in which the number of items can be very large (see [27]).

The N-body problem consists on computing the gravitational force between N particles in the 3D space, where each particle exerts a force on all the other particles, implying pairwise interactions. This kind of physical system occurs in several fields, like celestial mechanics, plasma physics, fluid mechanics and semiconductor device simulations [22]. Several efficient algorithms were proposed to reduce the asymptotic computational complexity from $O(N^2)$, in the naive approach, to $O(NlogN)$ (see [3, 5]) and to $O(N)$ (see [19] for particles in two dimensions and [20, 38] for particles in three dimensions). In [24] a study of the performance of these methods in parallel computers is analyzed. Based on these ideas, we develop a strategy to reduce the computational complexity of evaluating (3).

It is hard to realize a procedure to detect which pairs *will contribute* to the sum in (3) without computing all the $O(N^2)$ distances. However, it is easy to detect which pairs *will not contribute* and just to compute the distances for the other pairs (which may contribute or not). Consider a partition of the object in regions in such a way that circular items whose centers are in non-adjacent regions can't overlap. For the case of identical circular items with radius r it is easy to see that a partition in squared regions of side $\delta = 2r$ has such property (see Figure 2). For the case of circular items with radii r_i, $i = 1, \dots, N$, the size of the squared regions should be $\delta = 2\max_i\{r_i\}$. Now, given an item, it is just necessary to compute the distances from its center to the centers of the items which are in the same region or in an adjacent region.

Considering a squared object, the total number of squared regions needed to cover it is N_{reg}^2, where $N_{reg} = \lceil L_{ub}/\delta \rceil$ and L_{ub} is an upper bound on the squared object side. The squared regions are then surrounded by unbounded regions just to make sure that any point in the space belongs to a region, i.e., we are considering a partition of the whole space (see Figure 3). Given

$x \in I\!R^n$ ($n = 2, 3$ are the natural choices) the region to which it belongs can be computed in constant time as

$$Reg(x) = (p(x_1), \ldots, p(x_n)), \tag{4}$$

where

$$p(a) = \max\{0, \min\{\lfloor a/\delta \rfloor, N_{reg} + 1\}\}. \tag{5}$$

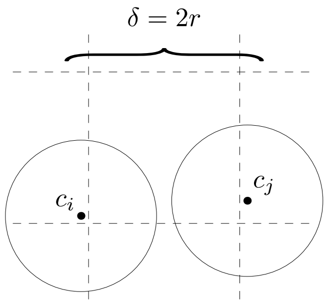


Figure 2: If two items are in non-adjacent regions then they do not overlap. In the picture, $d(c_i, c_j) \geq \delta = 2r$, so the items do not overlap.

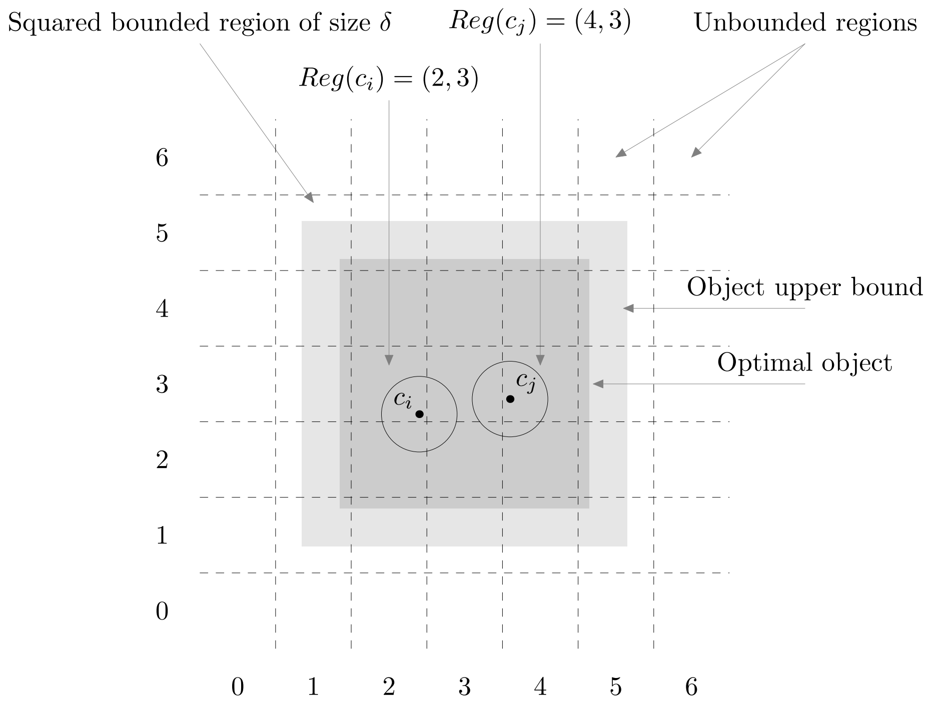


Figure 3: The space is partitioned into regions in such a way that the object is covered by squared regions. Items whose centers are in non-adjacent regions can't overlap. So, to verify overlappings, we just need to consider pairs of items in the same region or in adjacent regions.

The method starts with an empty data structure that represents the partition. Each region has a list of the items whose centers are in the region. There is a matrix with $(N_{reg}+2)^2$ elements with pointers to these lists. There is also a list with the non-empty regions (regions that have at least an item). See Figure 4. All these lists start empty. Given the centers $c_1, c_2, \ldots, c_N$ of the circular items, these are the steps to compute the left hand side of (3):

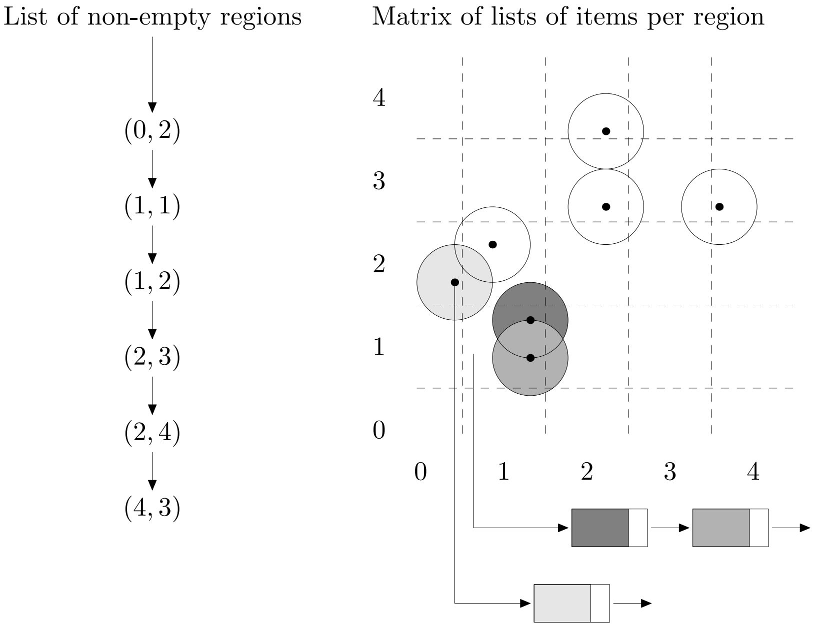


Figure 4: Data structure for the overlapping evaluation. Each region has a list of the items within it. There is also a list of the non-empty regions.

Overlapping evaluation.

Step 0: $sum \leftarrow 0$.

Step 1: For each item $i = 1, \ldots, N$ do:

- **Step 1.1:** Determine its region $Reg(c_i)$ given by (4)–(5).
- **Step 1.2:** Add item i to the list of items whose centers are in $Reg(c_i)$.
- **Step 1.3:** If this is the first item assigned to $Reg(c_i)$, add $Reg(c_i)$ to the list of non-empty regions.

Step 2: For each non-empty region R and for each item i in R do:

- **Step 2.1:** For each item j in R or in a region adjacent to R do:
  - **Step 2.1.1:** Compute $sum \leftarrow sum + \max\{0, (r_i + r_j)^2 - d(c_i, c_j)^2\}^2$.

Step 3: Clear the data structure and return sum.

Remark: At Step 2.1, in order to avoid computing all distances twice, two actions are taken: (i) Inside the same region, compute the distances from item i just to the items that appears after i in the list of items of region $Reg(c_i)$; (ii) Regarding adjacent regions, just a subset of the adjacent regions needs to be considered, as the other ones will be considered when the regions interchange their roles (see Figure 5).

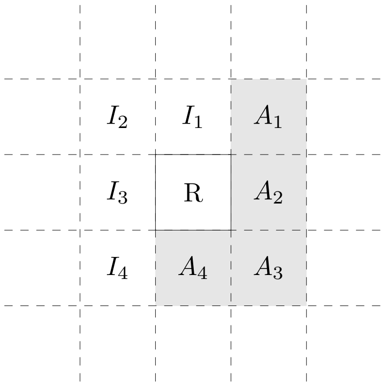


Figure 5: When the items in region R are being considered, just the distances to the items in regions A_1, A_2, A_3 and A_4 need to be computed. The distances to the items in regions I_1, I_2, I_3 and I_4 are computed when the items in $I_k, k = 1, 2, 3, 4$, play the central role. In $I\!R^3$ the number of adjacent regions to consider is 13. The total number of adjacent regions in $I\!R^3$ and $I\!R^2$ are 26 and 8, respectively.

Assuming that the number of items in each region is constant (independent of N) then, clearly, the complexity of the whole process is $O(N)$. The assumption of the constant number of items per region is very reasonable as, when the method approaches the solution or even in a random uniformly distributed initial configuration, the items are well distributed in the space.

In order to assess the practical impact of the partitioning approach in the overlapping evaluation, we compare its CPU time against the CPU time of the naive approach for an increasing number of randomly distributed identical circular items. Figure 6 shows the results. As expected, while the CPU time of the partitioning approach can be linearly approximated by $Time(N) \approx 0.00026943\ N\ ms$, the CPU time of the naive approach can be quadratically approximated by $Time(N) \approx 0.0000028301\ N^2\ ms$.

We also performed the same experiment considering randomly distributed different-sized circular items with uniformly distributed radii within the interval $[R_{\min}, R_{\max}]$ and $R_{\max}/R_{\min} = 10, 100, 1000$. The figures were virtualy the same. The explanation for this behaviour relies on the fact that, although a larger number of comparisons is done, while N goes to infinity the number of neighbour items remains constant. Of course, extreme cases with just one large item and many small items such that the object is partitioned in almost a unique region will make the partitioning approach to behave as the naive approach (or even worse considering the overhead of the data structures manipulation).

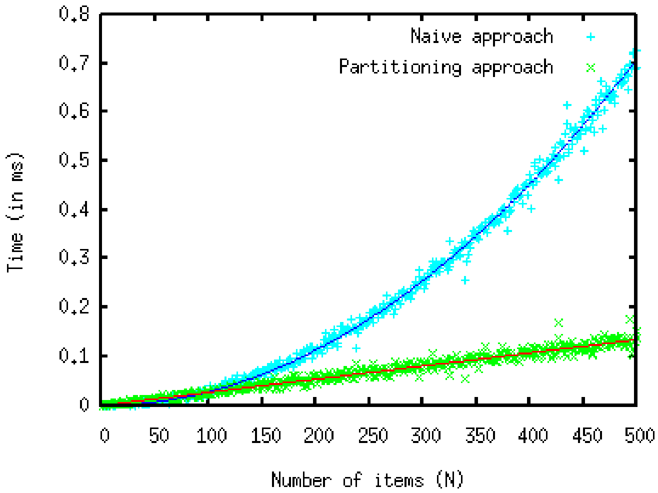


Figure 6: Practical performance of the naive and the partitioning approaches for the overlapping evaluation of an increasing number of randomly distributed identical items.

2.2 Object dimension and placing constraints

In this subsection we focus on the problem of fitting the items within the object while minimizing the object area or volume, and, in the cases in which it is not equivalent, the object perimeter or surface area. We consider several kinds of 2D and 3D objects.

If the object is a circle, to minimize its area is equivalent to minimize its radius R. Moreover, the fact that a circular item with radius r_i belongs to the circular object (which, without loss of generality, can be considered centered at the origin) can be modeled as $d(c_i, 0)^2 \leq (R - r_i)^2$ and $R \geq r_i$. In the case of a squared object, to minimize its area is equivalent to minimize its side L. Moreover, assuming that the bottom-left corner of the object is fixed at the origin and that its sides are parallel to the axis, the fitting of a circular item with radius r_i within the squared object with side L can be modeled as $r_i \leq c_i \leq L - r_i$. The cases in which the object is an equilateral triangle, a rectangle or a strip are analogous. To consider the 3D objects counterparts is also analogous. The rectangle, the cuboid and the cylinder are the only cases we consider for which their dimensions depend on more than one parameter and, consequently, minimizing the area or volume is not equivalent to minimizing the perimeter or surface area. Tables 1 and 2 give the model for each 2D and 3D problem, respectively, as well as the number of variables and constraints. The extension to consider convex objects represented by linear constraints is trivial. General convex 2D regions were also considered in [11, 12].

According to the typology recently introduced in [37], problems in Tables 1 and 2 (identified by object type) can be classified as follows: (i) two-dimensional circular ODP (Open Dimension Problem): circle, square, 2D strip, rectangle and equilateral triangle; (ii) three-dimensional spherical ODP: sphere, cube, 3D strip, cuboid, tetrahedron, pyramid and cylinder.

Object type	Model	# of variables	# of constraints
Circle	Min R s.t. $(c_i^x)^2 + (c_i^y)^2 \leq (R - r_i)^2, \ \forall\, i$ $R \geq r_{\max} \equiv \max_{i=1,\dots,N} \{r_i\}$ non-overlapping constraint (3)	$2N + 1$	$N + 2$ (1 is a box constraint)
Square	Min L s.t. $r_i \leq c_i^x \leq L - r_i, \ \forall\, i$ $r_i \leq c_i^y \leq L - r_i, \ \forall\, i$ non-overlapping constraint (3)	$2N + 1$	$4N + 1$ ($2N$ are box constraints)
Strip	Min W s.t. $r_i \leq c_i^x \leq L - r_i, \ \forall\, i$ $r_i \leq c_i^y \leq W - r_i, \ \forall\, i$ non-overlapping constraint (3)	$2N + 1$	$4N + 1$ ($3N$ are box constraints)
Rectangle	Min LW or Min $L + W$ s.t. $r_i \leq c_i^x \leq L - r_i, \ \forall\, i$ $r_i \leq c_i^y \leq W - r_i, \ \forall\, i$ non-overlapping constraint (3)	$2N + 2$	$4N + 1$ ($2N$ are box constraints)
Equilateral triangle	Min L s.t. $c_i^y \geq r_i, \ \forall\, i$ $6c_i^x + 2\sqrt{3}c_i^y \leq 3L - 4\sqrt{3}r_i, \ \forall\, i$ $-6c_i^x + 2\sqrt{3}c_i^y \leq 3L - 4\sqrt{3}r_i, \ \forall\, i$ non-overlapping constraint (3)	$2N + 1$	$3N + 1$ (N are box constraints)

Table 1: 2D models for minimizing the object dimension when the object is a circle, a square, a strip, a rectangle and an equilateral triangle. (In the nonlinear programming context, box (or bound) constraints are considered easy constraints.)

3 Numerical experiments

We are interested in finding a global solution of the proposed nonlinear programming models. To increase the probability of finding a global solution, we run a local solver starting from several random initial points. Given the number of items to be packed and the shape of the object, we first compute an upper bound for the parameter (or parameters) that defines the object dimension. Then, we randomly distribute the items within the overestimated object and run the local solver starting from this randomly generated configuration as initial guess. This process is repeated until a maximum allowed CPU time T is exceeded. The best local solution is returned as a solution.

We chose Algencan [2, 1, 10, 7] as the local solver. Algencan is a recently introduced Augmented Lagrangian method for smooth general-constrained minimization. The method is fully described in [2] where extensive numerical experiments assess its reliability. Algencan is available as a part of the Tango Project (see the web site [6]). In the present implementation Algencan uses Gencan [9] to solve the bound-constrained subproblems. Gencan is an active-set method for bound-constrained minimization. Gencan adopts the leaving-face criterion of [8], that employs the spectral projected gradients defined in [13, 14]. For the internal-to-the-face minimization Gencan uses a general algorithm with a line search that combines backtracking and extrapolation. In the present available implementation, Gencan employs, for the direction chosen at each step inside the faces, a truncated-Newton approach with incremental quotients to approximate the matrix-vector products and memoryless BFGS preconditioners [10].

All the experiments were run on a 2GHz AMD Opteron 244 processor, 2Gb of RAM memory

Object type	Model	# of variables	# of constraints
Sphere	Min R s.t. $(c_i^x)^2 + (c_i^y)^2 + (c_i^z)^2 \le (R - r_i)^2,\ \forall\, i$ $R \ge r_{\max} \equiv \max\limits_{i=1,\dots,N}\{r_i\}$ non-overlapping constraint (3)	$3N+1$	$N+2$ (1 is a box constraint)
Cube	Min L s.t. $r_i \le c_i^x \le L - r_i,\ \forall\, i$ $r_i \le c_i^y \le L - r_i,\ \forall\, i$ $r_i \le c_i^z \le L - r_i,\ \forall\, i$ non-overlapping constraint (3)	$3N+1$	$6N+1$ ($3N$ are box constraints)
3D strip	Min H s.t. $r_i \le c_i^x \le L - r_i,\ \forall\, i$ $r_i \le c_i^y \le W - r_i,\ \forall\, i$ $r_i \le c_i^z \le H - r_i,\ \forall\, i$ non-overlapping constraint (3)	$3N+1$	$6N+1$ ($5N$ are box constraints)
Cuboid	Min LWH or Min $LW + LH + WH$ s.t. $r_i \le c_i^x \le L - r_i,\ \forall\, i$ $r_i \le c_i^y \le W - r_i,\ \forall\, i$ $r_i \le c_i^z \le H - r_i,\ \forall\, i$ non-overlapping constraint (3)	$3N+3$	$6N+1$ ($3N$ are box constraints)
Tetrahedron	Min L s.t. $2\sqrt{2}c_i^x - 2\sqrt{6}c_i^y + 2c_i^z \le \sqrt{6}L - 6r_i\ \forall\, i$ $2\sqrt{2}c_i^x + 2\sqrt{6}c_i^y + 2c_i^z \le \sqrt{6}L - 6r_i\ \forall\, i$ $-2\sqrt{2}c_i^x + c_i^z + r_i \le 0,\ \forall\, i$ $c_i^z \ge 0,\ \forall\, i$ non-overlapping constraint (3)	$3N+1$	$4N+1$ (N are box constraints)
Pyramid	Min L s.t. $2c_i^x + \sqrt{2}c_i^z \le L - \sqrt{6}r_i,\ \forall\, i$ $-2c_i^x + \sqrt{2}c_i^z \le L - \sqrt{6}r_i,\ \forall\, i$ $2c_i^y + \sqrt{2}c_i^z \le L - \sqrt{6}r_i,\ \forall\, i$ $-2c_i^y + \sqrt{2}c_i^z \le L - \sqrt{6}r_i,\ \forall\, i$ $c_i^z \ge 0,\ \forall\, i$ non-overlapping constraint (3)	$3N+1$	$5N+1$ (N are box constraints)
Cylinder	Min R^2H or Min $R(R+H)$ s.t. $(c_i^x)^2 + (c_i^y)^2 \le (R - r_i)^2,\ \forall\, i$ $R \ge r_{\max} \equiv \max\limits_{i=1,\dots,N}\{r_i\}$ $r_i \le c_i^z \le H - r_i,\ \forall\, i$ non-overlapping constraint (3)	$3N+2$	$3N+2$ ($N+1$ are box constraints)

Table 2: 3D models for minimizing the object dimension when the object is a sphere, a cube, a 3D strip, a cuboid, a pyramid with equilateral triangles as faces and a square as base, a regular tetrahedron and a cylinder.

and Linux operating system. Codes are in Fortran77 and the compiler option "-O4" was adopted.

In a first set of experiments, we fixed $T = 1$ hour and $T = 1.5$ hours and solved all the 2D and 3D instances with up 50 unitary-radius items, respectively. In addition to the previous set of problems, we fixed $T = 4$ hours and solved all the instances with $55, 60, \dots, 95$ and 100 unitary-radius items. Tables 5, 6 and 7 show the solution the method found for each combination of number of items and 2D and 3D object type, respectively. When the object is a circle or a square, the obtained solutions coincide (up to a prescribed tolerance) with the ones reported in [32]. For equilateral triangles with up to 15 items, the obtained results also coincide with the ones reported in [16]. For all the other 2D and 3D objects there are no previously reported results. (The results reported in [32] for rectangular objects assume that weight/width = 0.1.) Figures 7 and 8 illustrates a few selected solutions (3D figures were generated using VMD [23] and Raster3D [4]).

To give an idea of the computational cost of the present approach, Table 8 presents a few figures related to the 2D problems with circular and squared objects. The table shows the total number of nonlinear programming problems that were solved and the elapsed CPU time until the best solution was found. (The remaining time, to complete the maximum allowed CPU time T, was spent just to confirm that a better solution could not be found.) The computational effort of the method deserves some explanation. Larger the number of items, larger the amount of (undesired) local minimizers of the models. So, when the number of items increases, the simple multi-start global optimization strategy needs more local minimizations to find the "global" minimizer. The combination of the present approach with more sophisticated global optimization techniques might improve the computational performance of the method.

To evaluate the quality of the obtained solutions for the problems that minimize the object area (volume), we compared them with a simple lower bound given by the sum of the items area (volume). To compute a lower bound on the perimeter (surface area) of an object, we proceeded as follows. First, a lower bound a_{lb} (v_{lb}) on its area (volume) was computed. Then, we analiticaly solved the problem of minimizing its perimeter (surface area) subject to the object area (volume) being greater than or equal to a_{lb} (v_{lb}) and fitting at least the largest item. Table 3 shows, for each type of object, the average relative distance to the lower bound, computed as

$$\text{relative distance} = \frac{\text{solution found} - \text{lower bound}}{\text{lower bound}},$$

and the average density of the packings obtained. Note that, as it is known that optimal solutions were found for circular and squared objects (and also for equilateral triangular objects up to 15 items), the comparison against the figures (average relative distance to the bound and average packing density) of those cases can be used to evaluate the quality of the other cases. However, the comparison must be done with care, as it is expected, for example, that the density of a packing within a square to be worse than the density of a packing within a rectangle.

In another set of experiments, we fixed $T = 24$ hours and tested the behaviour of the proposed method in the strip packing problems with different-sized circular and spherical items considered in [35] and [36], respectively. The number of items varies from 25 to 60 and the value of the fixed dimensions of the strips as well as the radii of the items can be found in [36, 35]. Table 4 shows the results. The table shows a lower bound based on the areas or volumes ratio, the solutions obtained in [36, 35] and by the present approach, the total number of nonlinear programming

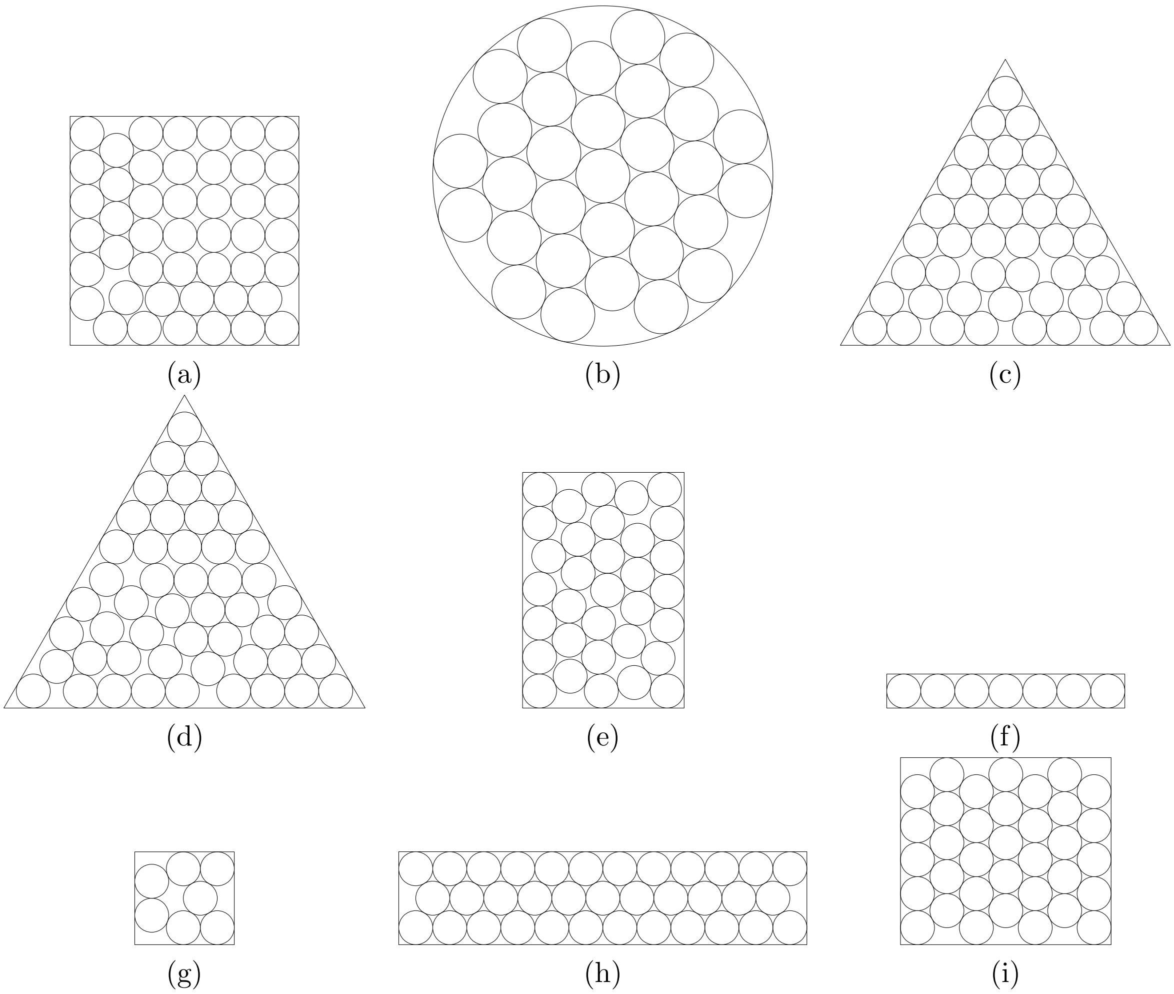
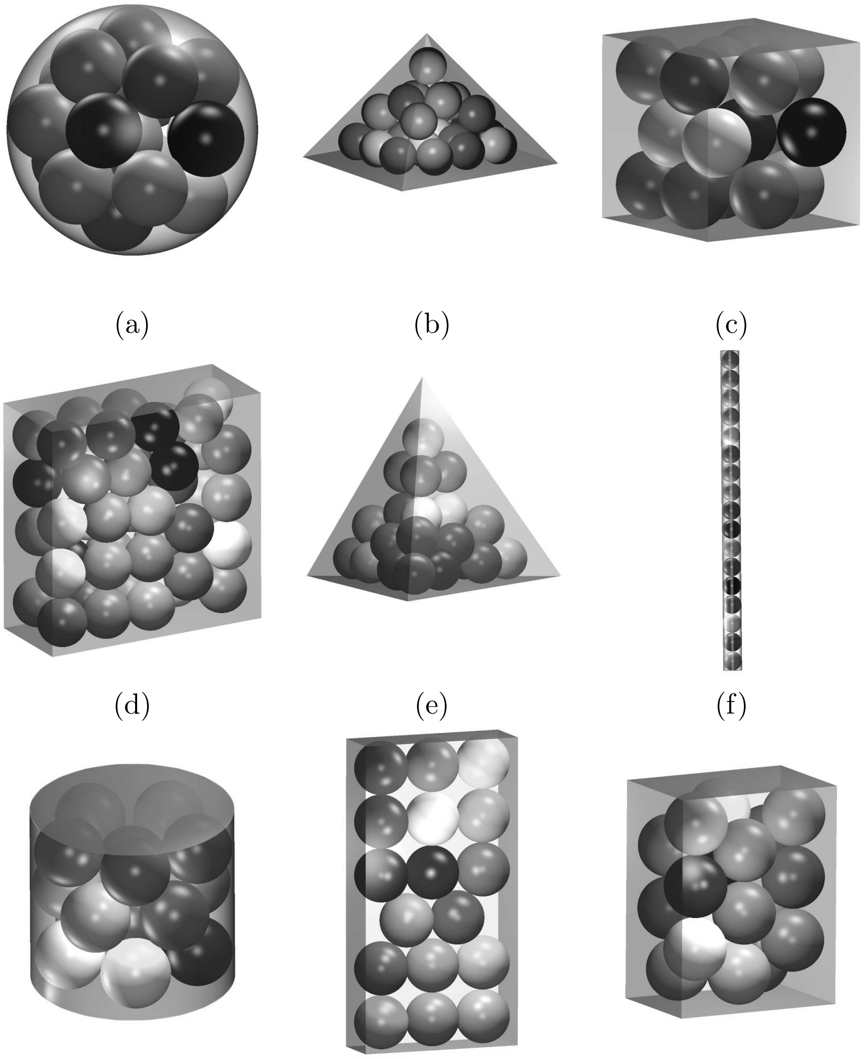


Figure 7: Selected pictures of 2D unitary-radius circle packing problems: (a) 46 circles in a square of size 13.4626063029, (b) 31 circles in a circle of radius 6.2907435849, (c)-(d) 42 and 50 circles in equilateral triangles of size 19.4045801371 and 21.2440452846, respectively, (e) 33 circles in a strip of fixed length 9.5 and width 13.8305751217, (f)-(g) 7 circles within a rectangle, minimizing area (13.998199825 $\times$ 2.0) and perimeter (5.8609504075 $\times$ 5.4637640106), respectively, (h)-(i) 35 circles within a rectangle, minimizing area (23.9970048173 $\times$ 5.463736504) and perimeter (12.3910375234 $\times$ 10.9989629363), respectively.

problems that were solved and the elapsed CPU time until the best solution was found. Figure 9 displays the graphical representation of the solutions. While the present approach failed to obtain good quality solutions in the two 2D problems, it was able to find better solutions in four over the six 3D problems.

As final examples of the wide range of applicability the present unified approach can have, Figure 10a shows the solution found for the problem of packing 100 different-sized spheres in a



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 8: Selected pictures of 3D unitary-radius sphere packing problems: (a) 17 spheres in a sphere of radius 3.2711271196, (b) 24 spheres in a squared pyramid of size 9.8116682857, (c) 18 spheres in a cube of size 5.3279843248, (d) 44 spheres in a three-dimensional strip of fixed length 9.5, fixed width 9.5 and height 3.9867870680, (e) 23 spheres in a regular tetrahedron of size 12.2090353736, (f)-(g) 17 spheres within a cylinder minimizing the volume (height 33.9972524429 and radius 1.0) and the surface area (height 5.2362405719 and radius 2.9018413949), respectively, and (h)-(i) 17 spheres within a cuboid minimizing the volume (11.4636013874 $\times$ 5.9997955157 $\times$ 2.0) and the surface area (5.4638212027 $\times$ 6.9742074355 $\times$ 3.7319493416), respectively.

	Object Type	Average distance to lower bound	Average density
2D	Circle	0.3143	0.7661
	Square	0.3091	0.7681
	Strip	0.4216	0.7494
	Rectangle (area)	0.2197	0.8206
	Rectangle (perimeter)	0.1259	0.7919
	Equilateral triangle	0.2878	0.7846
3D	Sphere	1.0598	0.4997
	Cube	1.0497	0.4960
	Strip	2.9543	0.4082
	Cuboid (volume)	0.8211	0.5501
	Cuboid (surface area)	0.5877	0.5364
	Tetrahedron	1.1965	0.4706
	Pyramid	1.2277	0.4622
	Cylinder (volume)	0.4999	0.6667
	Cylinder (surface area)	0.5742	0.5228

Table 3: Statistics related to the solutions found for the 2D and 3D instances with up 50 unitary-radius items.

tetrahedron. Finally, we also consider the case in which the object is given by a set of linear equations. In this case, we will define the objective of the problem as minimizing the object dimension (area, perimeter, volume or surface area) by preserving its "shape and proportion among sides or faces". In this case, the problem can be posed as

$$\begin{array}{ll} \text{Minimize} & L \\ \text{s.t.} & Ac_i + r_i u \le Lb, \ \forall\, i \\ & \text{non-overlapping constraint (3)} \end{array} \tag{6}$$

where $A \in I\!R^{k \times n}$, $b, u = (\|a_1\|_2, \ldots, \|a_k\|_2)^T \in I\!R^k$, a_j is the j-th row of A and k is the number of constraints that represent the object. As an example (see Figure 10b), consider the three-dimensional object given by:

$$A = \begin{pmatrix} -16 & & 4 \\ & 48 & 24 \\ 16 & & 24 \\ & -48 & 24 \\ & & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 160 \\ 96 \\ 32 \\ 96 \\ 0 \end{pmatrix}. \tag{7}$$

4 Final remarks

We introduced a large variety of twice-differentiable nonlinear programming models for the problem of minimizing the object dimension in 2D and 3D packing problems. Identical and

Problem		Lower bound	Solutions found		Effort measurements	
			In [36, 35]	Using GENPACK	Number of trials	CPU time is secs.
2D	(a)	12.2332	14.3785	14.9509618123	529	184.70
	(b)	14.5496	17.1968	18.0340842834	54221	27735.22
3D	(c)	5.0712	9.8667	9.7941763980	58563	29320.06
	(d)	5.7561	9.6220	11.0128604540	33918	52060.52
	(e)	4.8487	9.4728	9.3089999467	26851	41308.01
	(f)	5.8572	11.0862	11.0962093979	26806	85617.91
	(g)	6.1663	11.6453	11.6210793508	6418	28081.08
	(h)	6.8115	12.8415	12.7215414636	3829	19170.46

Table 4: Performance of the method in the strip packing problems with different-sized circular and spherical items from [36, 35].

different-sized items were considered. We implemented an efficient methodology to reduce the computational cost of computing the overlapping. A practical method was used to solve all the proposed models, attesting its applicability. Moreover, the presented methodology is fully parallelizable. The combination of the present approach with clever problem-dependent global optimization techniques like the one developed in [34, 36, 35] would be a line for future research.

The complete Fortran 77 sources codes of the algorithms and models presented in this paper are available in `http://www.ime.usp.br/~egbirgin/`.

Acknowledgements

The authors are thankful to Marcel Kenji, Renato Bortolatto and Ricardo Andrade for the careful reading of the manuscript. They are also indebted to two anonymous referees whose comments helped to improve this paper.

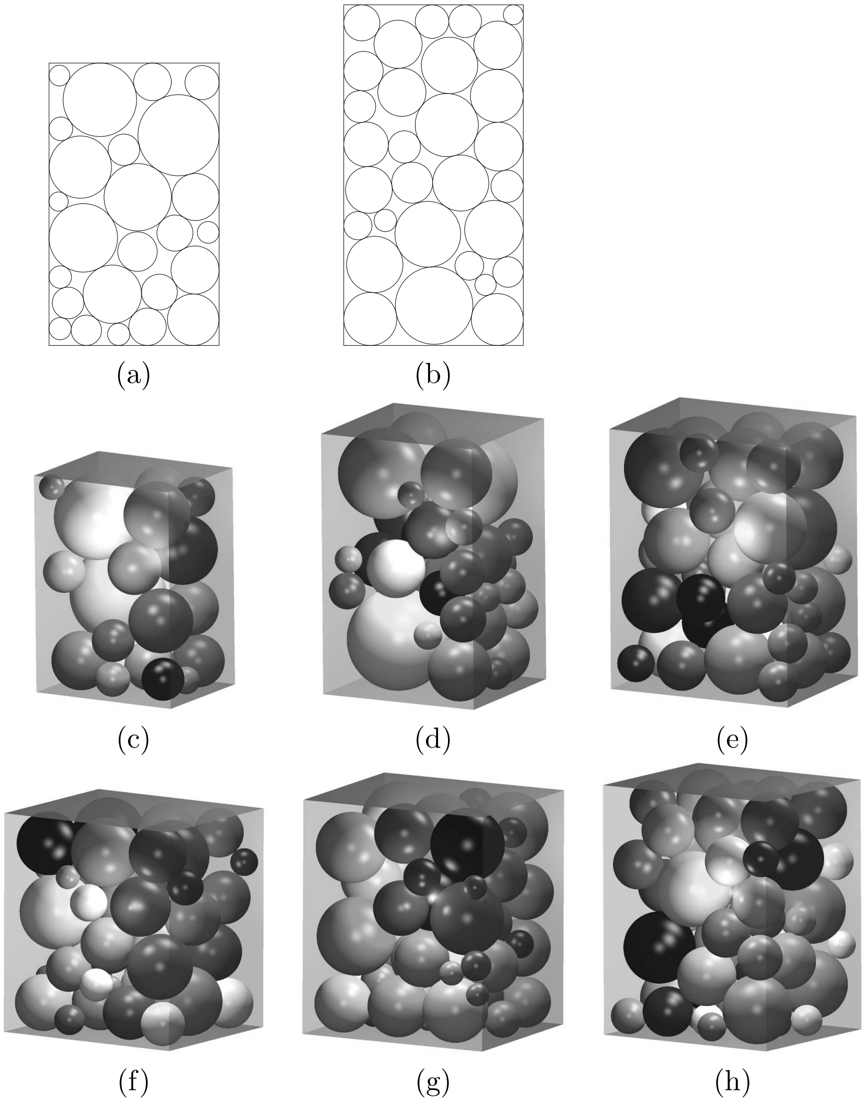
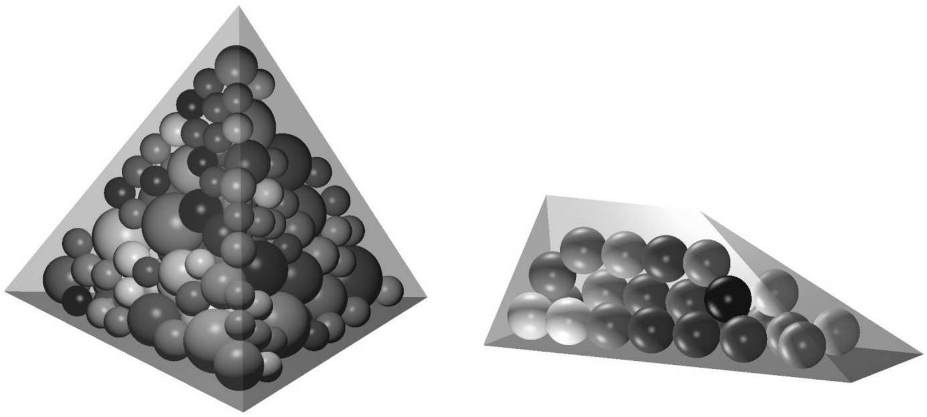


Figure 9: Problems of packing different-sized circular and spherical items within strips from [36, 35].



(a) (b)

Figure 10: (a) Packing of 100 different-sized spheres within a tetrahedron. There are 50 unitary-radius spheres and 50 spheres with radius $1.025, 1.050, \ldots, 2.250$. The dimension of the tetrahedron is given by $L = 27.3333464485$. (b) Packing of 25 unitary-radius spheres into a 3D object given by the arbitrary linear constraints (6–7). Solution: $L = 1.4999323248$.

Number of items	Circle R	Square L	Strip L	Triangle L	Rectangle (area) L	Rectangle (area) W	Rectangle (perimeter) L	Rectangle (perimeter) W
1	1.0000000000	1.9999986363	2.0000000000	3.4641016146	2.0000000000	2.0000000000	2.0000000000	2.0000000000
2	1.9996222546	3.4137477127	2.0000000000	5.4633621099	3.9994547642	2.0000000000	3.9992604948	2.0000000000
3	2.1544558927	3.9313808636	1.9999911315	5.4636533311	2.0000000000	5.9991345164	3.9995011744	3.7317273490
4	2.4139375051	3.9996299811	2.0000000000	6.9274637829	3.9996565400	3.9996565400	3.9996302816	3.9996302816
5	2.7010098451	4.8279613489	2.6952686399	7.4634468126	9.9986261610	2.0000000000	3.9998611307	5.4633052124
6	2.9996508504	5.3277596117	3.3224368208	7.4636709802	3.9997150028	5.9994299921	3.9997760936	5.9993282244
7	2.9997761921	5.7315971190	3.5608566245	8.9273837492	13.9981998250	2.0000000000	5.8609504075	5.4637640106
8	3.3044206851	5.8632353309	3.6884793600	9.2930737754	7.9993979180	3.9997993110	5.9995344103	5.4637633218
9	3.6127740710	5.9995518554	3.9993016842	9.4634763151	5.9996222843	5.9996222843	5.9995521684	5.9995521684
10	3.8126788316	6.7466961038	4.6951121788	9.4636358192	9.9992644508	3.9998161178	5.9996594681	7.1956196062
11	3.9228091423	7.0210502213	5.1211545417	10.7276943421	5.4634407024	7.9987047809	5.9991924047	7.4624830643
12	4.0286834176	7.1437936464	5.3765047895	10.9264246875	7.9986295356	5.9990863749	7.9982454190	5.9991228063
13	4.2349654525	7.4630289424	5.8523347064	11.4044366767	2.0000000000	25.9938432497	7.4627500826	7.4627500827
14	4.3275521837	7.7309165102	5.9980828464	11.4625024049	9.9984346514	5.4635325793	7.9984352858	7.1953665961
15	4.5206216997	7.8626415945	6.6937425787	11.4627797055	7.9988353651	7.4630923328	7.9986635705	7.4630647263
16	4.6143481044	7.9989556782	7.0649643624	12.7110751779	3.7318388122	16.9966762489	7.9986841453	7.9986841451
17	4.7912398107	8.5309003541	7.4495386792	12.9261773819	5.4635902166	11.9981939658	8.9131925338	7.9543346939
18	4.8627806310	8.6552029299	7.8517294049	13.2910260295	3.7318475438	18.9964440023	7.9990444062	8.9268910590
19	4.8629278433	8.9061300551	7.9976715063	13.4456687039	7.4632082242	9.9985864965	9.4625659570	7.9990156750
20	5.1212784390	8.9768656326	8.6935864074	13.4625490829	5.4636308244	13.9979711566	8.9272322541	8.9986390784
21	5.2512555209	9.3563084678	9.1690829563	13.4627429100	5.4636886104	14.9975772973	9.2167842122	9.4339701956
22	5.4390007858	9.4622675308	9.4490464873	14.6101271902	3.7318614400	22.9960005512	9.9407136592	8.9271510823
23	5.5445265607	9.7309291904	9.8510937710	14.8805620808	5.4636615016	15.9977610644	8.9273391176	9.9986129160
24	5.6508749142	9.8625920424	9.9971058011	14.9264967836	5.4637089641	16.9973842151	9.4629665387	9.9986287544
25	5.7520474056	9.9989092425	10.6930072130	15.2915870167	3.7318697115	25.9956839018	10.9982515685	8.9275611421
26	5.8274395455	10.3761296161	11.1687967336	15.4568486004	5.4636857741	17.9975609058	10.6588451987	9.9409595292
27	5.9054946032	10.4788058721	11.4484657320	15.4625744338	5.4637257927	18.9971982173	10.6591024240	9.9988547158
28	6.0141566559	10.6754220133	11.8505484609	15.4627172406	11.9985828103	8.9273321305	9.9989366878	10.9268949096
29	6.1377763523	10.8137503471	11.9966464202	16.6030248601	5.4637056254	19.9973689183	11.4625890050	9.9989080012
30	6.1970915627	10.9072687500	12.6932092975	16.7281252215	20.9970183919	5.4637400405	10.6592978781	10.9986516064
31	6.2907435849	11.1915352105	13.1685323354	16.9263595271	15.9979597338	7.4633846242	10.9270095378	11.4247556202
32	6.4287631231	11.3803881756	13.4480140426	17.2453120324	21.9971838652	5.4637222711	11.9404852062	10.6593828953
33	6.4856320243	11.4625012404	13.8472008358	17.4039294262	13.9984103697	8.9273990398	11.9985032730	10.6594384363
34	6.6101463641	11.7309249171	13.9962078766	17.4587344689	7.1956075610	17.9977228032	11.9986028314	10.9272272456
35	6.6964365742	11.8625361315	14.6929957747	17.4625898173	23.9970048173	5.4637365040	12.3910375234	10.9989629363
36	6.7459641444	11.9988792034	15.1690615764	17.4626981458	5.4637630915	24.9966743828	10.6595680742	12.9983311035
37	6.7579310827	12.1804216686	15.4475690694	18.5287831373	5.4637358338	25.8578787169	12.3031569983	11.9965783735
38	6.9610838204	12.2374434375	15.8270155555	18.7266870206	5.4637488666	25.9968310515	11.9817725699	12.3908769558
39	7.0571911486	12.2887512850	15.9957923056	18.9140027395	5.4637726295	26.9965090852	10.6596663838	13.9981945067
40	7.1231096220	12.6267382011	16.6927575874	18.9265358552	20.9974831288	7.1956410497	12.9268799669	11.9988610018
41	7.2592804090	12.7456257952	17.1686399573	19.2916298956	5.4637597441	27.9966619868	12.9406947738	12.3912989627
42	7.3462119818	12.8518646718	17.4477019156	19.4045801371	7.1956025496	21.9971608203	12.3913577634	12.9986607790
43	7.4192209173	13.0972405183	17.8098929473	19.4596912793	17.9980933123	8.9274901429	13.5989755952	12.3917670632
44	7.4972867738	13.1939383256	17.9953961237	19.4625983940	29.9964971470	5.4637694187	12.3914982006	13.9404577794
45	7.5722836395	13.3803865970	18.6925377387	19.4626504950	5.4637888824	30.9961898134	13.9814312021	12.3913995511
46	7.6494228038	13.4626063029	19.1682027497	20.5249300923	7.1956672420	23.9972574437	13.9985220814	12.6593249459
47	7.7233640079	13.6751033111	19.4475892668	20.7011666871	5.4637781024	31.9963361352	12.9271460134	13.9985889130
48	7.7905553343	13.8047806547	19.8069204856	20.8799581334	19.9979450130	8.9275232377	12.9989194323	14.1231541809
49	7.8860817787	13.9472637804	19.9948589623	20.9264402987	5.4637306400	33.9945655425	12.6173802384	15.0112526170
50	7.9468087710	14.0083589613	20.6923384169	21.2440452148	5.4637859581	33.9961786161	13.6373330997	14.3820364308
55	8.2101683532	14.6923955360	22.6835225197	21.4622516388	8.9275832375	22.9976175240	14.1524486932	14.9442580942
60	8.6454213199	15.3755552334	24.6785675081	22.9265606115	17.9983545775	12.3913421248	14.1235217033	15.9985449327
65	9.0166711986	15.8203131072	26.6507629243	23.4622014647	5.4638164895	43.9954346000	15.7232054364	15.9601095466
70	9.3451963876	16.4998033416	28.6306926662	24.8807696211	8.9276367873	28.9972283659	15.8552623767	16.9113958554
75	9.6711933041	17.0956713219	30.6445646992	25.4558282921	8.9276510682	30.9971052432	17.9965380801	15.8417107227
80	9.9673713500	17.4293700602	32.6135139169	26.5919337671	8.9276638633	32.9969839021	18.9231113668	15.9766418153
85	10.1624343053	17.9585744521	34.5997299255	27.2455664377	34.9968654011	8.9276756174	17.9969843318	17.8547723425
90	10.5452581347	18.6092727313	36.6010872992	27.4621886433	10.6597334412	30.9978987556	19.1320378453	17.9971314230
95	10.8393458264	19.0830329823	38.5727471051	28.7283473468	12.3916727778	27.9982222555	19.9950093368	18.0780528627
100	11.0816840023	19.4522055083	40.5514750264	29.3893128146	12.3916727778	27.9982222555	19.5888129645	19.3655902612

Table 5: Numerical results for the 2D problems.

Number of items	Sphere R	Cube L	Cuboid (volume) L	W	H	Cuboid (surface) L	W	H
1	1.0000005143	1.9999987021	1.9999994082	1.9999994082	1.9999994082	1.9999998015	1.9999998015	1.9999998015
2	1.9999202433	3.1546039826	2.0000000000	3.9998520179	2.0000000000	3.9998520181	2.0000000000	2.0000000000
3	2.1546210616	3.4141527373	2.0000000000	5.9997650951	2.0000000000	2.0000000000	2.0000000000	5.9997650951
4	2.2246870768	3.4141480640	3.9999067784	3.9999067784	2.0000000000	3.9999153028	2.0000000000	3.9999153028
5	2.4141267898	3.7887647862	9.9996271139	2.0000000000	2.0000000000	5.4639348176	2.0000000000	3.9999512713
6	2.4141694573	3.8855478230	3.9999226428	5.9998452847	2.0000000000	3.9999355322	2.0000000000	5.9998549455
7	2.5911991946	3.9977338382	3.9999400751	2.0000000000	7.4638804452	3.9999563663	2.0000000000	7.4638868064
8	2.6452822222	3.9999345267	3.9999353642	3.9999353642	3.9999353642	5.9998908944	2.0000000000	5.4640108186
9	2.7320034623	4.3093402273	5.9998707285	5.9998707285	2.0000000000	5.9998870712	2.0000000000	5.9998870712
10	2.8324167450	4.6666081946	3.9999370664	9.9997482632	2.0000000000	3.7320243621	3.7320243621	5.9998991367
11	2.9019556947	4.8159601052	2.0000000000	5.4639222153	7.9996484758	5.4638610588	2.0000000000	7.9994404199
12	2.9020105031	4.8280686639	3.9998824683	3.9998824683	5.9997649343	7.9994337904	2.0000000000	5.9996461265
13	2.9997977416	4.8282131208	2.0000000000	5.9997946298	8.9278206589	3.9999337537	3.9999337537	6.8277518754
14	3.0910230631	4.8282912625	2.0000000000	9.9995751531	5.4639471485	5.9997880092	3.9999467126	4.8281141461
15	3.1415101582	5.1997445225	7.4638276614	2.0000000000	7.9996838961	5.9997645604	3.7319723788	5.4638832882
16	3.2155339532	5.2964155013	3.7319816590	8.9996807054	3.7319816590	6.2423258359	3.4142320441	6.2423258359
17	3.2711271196	5.2996749577	5.9997955157	2.0000000000	11.4636013874	6.9742074355	3.7319493416	5.4638212027
18	3.3188360708	5.3279843248	9.9996565716	3.7319855151	3.7319855151	5.4639276546	3.7319915897	6.9996882715
19	3.3858513801	5.4586383844	9.4637456920	7.9997083203	2.0000000000	4.8282103240	3.9999557695	7.9996695729
20	3.4733633429	5.6048721029	3.7319889320	10.9996304432	3.7319889320	7.1958225538	3.7319840261	5.9998330126
21	3.4862092764	5.6431452803	5.4639638751	7.9997917071	3.7319840662	7.9996463417	3.7320008721	5.4639501784
22	3.5796881818	5.7710452688	5.9998723682	5.9998723682	4.8282875226	5.4639607819	3.9999031097	7.9996349464
23	3.6273889908	5.8199237336	7.6565874480	4.8283548413	4.8283548413	7.1956140331	3.7319880108	6.9995678216
24	3.6852536176	5.8633943713	5.4639744767	8.9997659859	3.7319892222	7.1958834279	3.7320005199	6.9997694833
25	3.6872896999	5.9589662351	5.9998979005	8.9279025044	3.7319795011	8.9277722105	3.7319917901	5.9998728297
26	3.7471886302	5.9952148260	3.7319762835	9.9742707123	5.4639484659	7.1959593106	3.7320172791	7.9658996899
27	3.8132974881	5.9998355203	9.9997461864	5.4639818590	3.7319928014	9.9995542616	3.7320141670	5.4639826430
28	3.8415149358	6.2421317695	7.9998295947	7.1959606322	3.7319897620	7.1959176362	3.7320092538	7.9997307495
29	3.8769556364	6.2423824504	3.7319826950	10.9742467685	5.4639615210	8.9278148456	3.7320060834	6.9997214200
30	3.9163640724	6.2424721518	10.9997251845	5.4639882208	3.7319958897	8.9278471493	3.7320065066	6.9998178874
31	3.9506163857	6.2425174557	7.9996954592	5.4639083563	5.4638915004	8.9996802550	3.7320300978	7.1959753668
32	3.9873139176	6.2425285827	5.4639777146	7.9998611848	5.4639777146	7.9997796243	5.4639770041	5.4639770041
33	4.0197789249	6.4689813902	11.9997062916	5.4639933032	3.7319983540	9.8130876582	3.7320307315	7.1960389489
34	4.0475629785	6.5735658875	5.4639912617	12.8589488963	3.7319936763	9.9664645648	3.7320116113	7.4612564245
35	4.0842793950	6.5931612342	5.4639946940	12.9996092372	3.7319987413	9.0706267144	4.8284135759	6.2425430006
36	4.1128778440	6.6970889746	5.4639977043	12.9996874801	3.7320004885	9.9996505797	3.7320211822	7.1959652997
37	4.1546179855	6.7083709308	7.6566693014	7.6566712985	4.8283733467	7.6565995398	4.8284183539	7.6565980201
38	4.1575074608	6.7093947495	4.8283780629	7.6566840081	7.6566840081	7.6566177480	4.8284198756	7.6566177480
39	4.2238096708	6.7739983433	5.4640014512	13.9996696629	3.7320023049	7.9997871781	5.4640126543	6.9978553162
40	4.2551873286	6.7998570644	7.1959946200	10.9997719786	3.7320006207	7.9997867024	5.2659037464	7.3884098761
41	4.2961778465	6.9039667243	5.4639942369	9.9998247954	5.4639942369	7.9996918020	5.4639887407	7.1958647739
42	4.3080119020	6.9906644142	7.1959834279	7.9998806718	5.4639906152	7.1959619143	5.4639991620	7.9998289682
43	4.3528504653	7.0610542561	11.9992433167	7.1959553361	3.7319926669	7.9998102754	5.4640058977	7.4638869282
44	4.3827055183	7.0991542522	11.9997554903	7.1960017830	3.7320029043	7.4638315875	5.7112633939	7.9998952839
45	4.4068820245	7.1269867567	10.9997985680	5.4640016367	5.4640016367	6.9998991121	5.4639942298	8.9279089439
46	4.4409860772	7.1396025069	7.1959915445	12.9657366193	3.7320002792	8.9944778512	5.4640390355	7.1960161968
47	4.4739825106	7.1447631846	12.9997061426	7.1960058286	3.7320057053	8.9996324205	5.4640009566	7.1959237566
48	4.4961576629	7.2254788705	7.1960079302	12.9997392813	3.7320048642	8.9997922699	5.4640138768	7.1959859052
49	4.5191083524	7.3396050716	11.9995595224	5.4639889656	5.4639845156	8.9279159160	5.2659430685	7.9998131188
50	4.5504157703	7.3606467872	11.9997927906	5.4640048144	5.4640048144	7.6566831702	6.2425762929	7.6566831702
55	4.6849920452	7.6497254384	11.9997854621	8.9280146257	3.7320062218	9.9997441915	5.4640300646	7.4639258867
60	4.7748065618	7.6567160689	10.9998345830	7.1960136241	5.4640104777	8.9998349213	5.4640250495	8.9279785504
65	4.9241573261	7.9389780072	8.9280289549	13.9997569867	3.7320096198	9.9994110181	5.4640152915	8.9278768296
70	5.0328821287	8.1572766782	7.6567464615	10.4850757364	6.2425743011	10.4849680177	6.2426124333	7.6567505843
75	5.1642307312	8.2424863321	10.9998561341	8.9280296531	5.4640168915	11.1805242617	5.5583422100	9.1634519725
80	5.2752317706	8.5400043137	7.6567555710	11.8992578456	6.2425798506	11.9991661607	5.4639641616	8.9277971955
85	5.3792538717	8.6924540639	11.9996488835	7.1959794230	7.1959936564	10.9993312884	5.4640567077	10.6601156192
90	5.4771826035	8.8678323052	13.3134414952	7.6567626911	6.2425842090	8.9269377677	7.1941376138	10.3636092004
95	5.5716172496	9.0184678703	8.9280290941	13.9997110994	5.4640176971	9.0875358264	6.3481408059	12.2608997182
100	5.6661018170	9.1663256540	8.9280659172	14.9594160981	5.4640342817	10.4850433554	7.6567796673	9.0709141038

Table 6: Numerical results for the 3D problems.

Number of items	3D strip	Tetrahedron	Pyramid	Cylinder (surface)		Cylinder (volume)	
	L	L	L	H	R	H	R
1	2.0000000000	4.8989794855	3.8637033034	2.0000000000	1.0000000000	2.0000000000	1.0000000000
2	1.9999999511	6.8987786397	5.2777419314	2.0000000000	1.9998851315	3.9999799164	1.0000000000
3	1.9999996826	6.8988829315	5.7954078635	2.0000000000	2.1546325115	5.9999681193	1.0000000000
4	1.9999997558	6.8989186607	5.8636236011	7.9995822321	1.0000000000	7.9999582245	1.0000000000
5	1.9999998012	8.1648125393	5.8636160349	3.6329467608	2.1546454674	9.9999598635	1.0000000000
6	1.9999998345	8.7817120034	6.6919872393	3.6329550792	2.1546531385	11.9999534257	1.0000000000
7	1.9999998574	8.8988016274	7.1456607739	3.9999562296	2.3228259410	13.9999474063	1.0000000000
8	1.9999998750	8.8988805540	7.2778333226	3.6817732264	2.4141690175	15.9999417139	1.0000000000
9	1.9999998889	8.8989049422	7.5660753093	5.2659119030	2.1546760712	17.9999362872	1.0000000000
10	1.9999998872	8.8989186535	7.7273237304	3.7012882426	2.7012496035	19.9999310827	1.0000000000
11	1.9999998908	10.1473178899	7.8588071332	3.6532929729	2.8209499930	21.9979915089	1.0000000000
12	1.9999999018	10.3327450473	7.8632345878	5.3634470841	2.4141288826	23.9978597508	1.0000000000
13	1.9999998503	10.5039904936	7.8634093587	3.7111744027	2.9998379486	25.9977315770	1.0000000000
14	1.9999999132	10.7497129642	7.8634553793	3.9994148462	2.9961624606	27.9976076192	1.0000000000
15	1.9999998995	10.8777393372	8.6303103460	5.4025004405	2.7011994854	29.9974864572	1.0000000000
16	1.9999998967	10.8983529191	8.8467717390	7.0451550000	2.4141488724	31.9973681489	1.0000000000
17	1.9999999316	10.8985312575	8.9751441419	5.2362405719	2.9018413949	33.9972524429	1.0000000000
18	1.9999999092	10.8986448409	9.1412510454	5.4201459445	2.9567222851	35.9971391244	1.0000000000
19	1.9999999091	10.8986690234	9.2769864393	5.4223037529	2.9998633645	37.9970280076	1.0000000000
20	1.9999999175	10.8987161340	9.2776202362	5.4223197427	2.9998806834	39.9969189309	1.0000000000
21	1.9999999449	11.9107290304	9.5209353533	5.9988407441	2.9896028065	41.9968117522	1.0000000000
22	1.9999999378	12.1644939511	9.5954475686	6.9596198256	2.8210553449	43.9967063459	1.0000000000
23	2.5159516472	12.2090353736	9.7466361361	6.7723208885	2.9612676153	45.9966026003	1.0000000000
24	2.6231977042	12.4099719051	9.8116682857	7.1300489686	2.9567312663	47.9965004154	1.0000000000
25	2.6956448382	12.5741811638	9.8544161043	7.4064098865	2.9232129228	49.9963997013	1.0000000000
26	2.9617066600	12.6747046630	9.8627741922	7.1334431350	2.9998954899	51.9963003768	1.0000000000
27	3.0684401131	12.7736673573	9.8633027680	7.1334608972	2.9999050186	53.9962023683	1.0000000000
28	3.1196255500	12.8600469119	9.8634416955	7.9968894897	2.9844231581	55.9961056087	1.0000000000
29	3.1913632493	12.8842225810	9.8634679168	7.9997982380	3.0407002526	57.9960100366	1.0000000000
30	3.2086893898	12.8980309149	9.8635090748	7.1505964007	3.3044734720	59.9959155958	1.0000000000
31	3.2859642224	12.8984986467	10.5305062334	7.1507270283	3.3046552989	61.9958222347	1.0000000000
32	3.3086949340	12.8986547111	10.6235488127	8.8348924555	2.9999147311	63.9957298179	1.0000000000
33	3.3178381842	12.8986989768	10.7625054766	8.8445826634	2.9999150187	65.9956385637	1.0000000000
34	3.3204768911	12.8987169455	10.8335071538	8.8446035999	2.9999206841	67.9955481688	1.0000000000
35	3.3224991941	12.8987348085	11.0171361880	9.9954742176	2.9760543713	69.9954586825	1.0000000000
36	3.3227701051	13.8016667112	11.0838846064	7.1616804229	3.6130019962	71.9953700694	1.0000000000
37	3.5030791690	14.0412319624	11.1960904966	7.1623926235	3.6381953396	73.9952822965	1.0000000000
38	3.5474588054	14.1657928132	11.2697974969	8.8673378976	3.3046009656	75.9951953326	1.0000000000
39	3.6033087937	14.2196073802	11.2890658063	8.8676233722	3.3046726654	77.9951528898	1.0000000000
40	3.6427538524	14.3444779177	11.4043727618	10.5557274333	2.9999277002	79.9950671884	1.0000000000
41	3.7221021534	14.4386612980	11.5140686475	10.5557491304	2.9999315502	81.9959862095	1.0000000000
42	3.8536666422	14.5640872361	11.6082371018	8.7907404838	3.5579717977	83.9959193185	1.0000000000
43	3.9249852736	14.6375723666	11.6669595312	8.8804026948	3.5693360047	85.9958535090	1.0000000000
44	3.9867870680	14.7494779827	11.7267093766	7.1694333250	3.9417801321	87.9957876533	1.0000000000
45	4.1307639555	14.8022341879	11.7669103726	8.8822182115	3.6130144699	89.9957228992	1.0000000000
46	4.2650163216	14.8666473324	11.8183969887	7.1833112011	3.9434227567	91.9956583385	1.0000000000
47	4.3919831741	14.8843583422	11.8389199288	12.2668751642	2.9999367635	93.9955940066	1.0000000000
48	4.4702282327	14.8888916898	11.8543855969	12.2668975375	2.9999394941	95.9955306328	1.0000000000
49	4.5479606952	14.8977952472	11.8602390935	8.8864887258	3.7543441279	97.9954674610	1.0000000000
50	4.6361497395	14.8980720507	11.8631629616	10.1548109139	3.6130894764	99.9954044402	1.0000000000
55	4.8415441305	14.8987433181	11.8635296721	13.9780304659	2.9999509413	109.9950972151	1.0000000000
60	5.3118320575	16.1509800447	12.7654145569	10.6125761426	3.8506973721	119.9947990625	1.0000000000
65	5.7279874805	16.5478815306	13.1924098309	10.6151070070	3.9236949232	129.9945087687	1.0000000000
70	5.9680473558	16.8200872202	13.4726646298	15.7352137761	3.3047042663	139.9942266038	1.0000000000
75	6.4032451608	16.8976376228	13.7256661130	12.3341346475	3.9235372581	149.9939509187	1.0000000000
80	6.9617540267	16.8986738106	13.8486607110	13.7717708962	3.9236584875	159.9936813790	1.0000000000
85	7.2908059511	17.6079959898	13.8632029690	13.7807630028	3.9237709088	169.9934168155	1.0000000000
90	7.4688349216	18.2065359049	13.8635411877	15.3806065880	3.9237168589	179.9931591781	1.0000000000
95	7.9994329832	18.5235744448	14.5689221176	15.1764940219	3.9237791996	189.9929003178	1.0000000000
100	8.5044325927	18.7913616667	14.9342481081	15.7490442120	3.9236938448	199.9926489863	1.0000000000

Table 7: Numerical results for the 3D problems (cont.).

	Circle		Square	
Number of items	Number of trials	CPU time in seconds	Number of trials	CPU time in seconds
1	1	0.00	1	0.00
2	9	0.14	1	0.00
3	978	10.81	12	0.01
4	49382	270.93	1	0.00
5	2194	19.15	1	0.00
6	7497	64.19	38	0.56
7	200231	1154.91	16	1.19
8	194651	1291.75	50	5.69
9	99678	809.78	4	0.74
10	1457	15.98	148	23.07
11	76924	650.97	14	1.95
12	10444	112.07	22	5.32
13	9003	95.65	191	32.21
14	14242	180.95	15	4.60
15	30803	417.10	812	148.67
16	96878	1390.54	1	0.00
17	112173	1555.73	615	131.41
18	6031	96.12	52	13.27
19	399	6.47	9	3.20
20	33934	694.50	13	4.31
21	5447	137.53	208	60.78
22	24829	679.90	107	22.41
23	89	3.41	38	9.34
24	84943	3049.42	974	223.51
25	201	9.16	2	0.03
26	39288	1731.99	10	2.35
27	35280	1865.71	268	70.48
28	4386	252.17	384	92.88
29	3575	223.01	50	16.81
30	2	0.09	79	23.47
31	1	0.02	375	102.02
32	2	0.13	257	53.67
33	1167	89.87	35	8.58
34	1029	83.38	228	50.41
35	46	3.48	4564	1029.51
36	3	0.22	31	16.93
37	2306	189.48	1011	297.64
38	25562	2430.00	4803	1318.76
39	7	0.64	215	83.36
40	20665	2191.25	56	20.97
41	96	10.51	166	64.72
42	246	29.63	179	59.08
43	527	70.99	6654	1698.42
44	139	37.10	173	54.16
45	3007	456.26	192	39.47
46	32	4.79	213	52.13
47	64	10.05	475	119.72
48	2422	425.56	2748	753.35
49	10859	2087.76	3335	1011.08
50	50	10.02	1315	437.59
55	102	24.64	2	0.19
60	179	56.58	512	204.70
65	5361	2049.69	6848	3559.31
70	2647	1245.92	11360	6237.46
75	13143	7411.86	18817	10339.88
80	628	457.77	8661	6661.87
85	16759	13277.81	12261	11589.59
90	10123	8783.84	9824	10765.19
95	12937	13321.23	5790	7871.10
100	11553	13760.70	987	1678.37

Table 8: Effort measurements for the packing problems with circular and squared objects.

References

[1] R. Andreani, E. G. Birgin, J. M. Martínez and M. L. Schuverdt, Augmented Lagrangian methods under the Constant Positive Linear Dependence constraint qualification, *Mathematical Programming*, to appear.

[2] R. Andreani, E. G. Birgin, J. M. Martínez and M. L. Schuverdt, On Augmented Lagrangian methods with general lower-level constraints, Technical Report MCDO-050303, Department of Applied Mathematics, UNICAMP, Brazil, 2005 (Available in Optimization On Line and in www.ime.usp.br/∼egbirgin/).

[3] A. W. Appel, An Efficient Program for Many-Body Simulations, *SIAM Journal on Scientific and Statistical Computing* 6, pp. 85–103, 1985.

[4] D. J. Bacon and W. F. Anderson, A Fast Algorithm for Rendering Space-Filling Molecule Pictures, *Journal of Molecular Graphics* 6, 219-220, 1988.

[5] J. Barnes and P. Hut, A Hierarchical $O(n \log n)$ force calculation algorithm, *Nature* 324, 1986.

[6] E. G. Birgin, `http://www.ime.usp.br/∼egbirgin/tango/`

[7] E. G. Birgin, R. Castillo and J. M. Martínez, Numerical comparison of Augmented Lagrangian algorithms for nonconvex problems, *Computational Optimization and Applications* 31, pp. 31–56, 2005.

[8] E. G. Birgin, J. M. Martínez, A box constrained optimization algorithm with negative curvature directions and spectral projected gradients, *Computing [Suppl]* 15, pp. 49–60, 2001.

[9] 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.

[10] E. G. Birgin and J. M. Martínez, Structured minimal-memory inexact quasi-Newton method and secant preconditioners for Augmented Lagrangian Optimization, *Computational Optimization and Applications*, to appear.

[11] E. G. Birgin, J. M. Martínez, W. F. Mascarenhas and D. P. Ronconi, Method of Sentinels for Packing Objects within Arbitrary Regions, *Journal of the Operational Research Society* 57, pp. 735–746, 2006.

[12] 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.

[13] E. G. Birgin, J. M. Martínez and M. Raydan, Nonmonotone spectral projected gradient methods on convex sets, *SIAM Journal on Optimization* 10, pp. 1196–1211, 2000.

[14] E. G. Birgin, J. M. Martínez and M. Raydan, Algorithm 813: SPG - Software for convex-constrained optimization, *ACM Transactions on Mathematical Software* 27, pp. 340–349, 2001.

[15] 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.

[16] E. Friedman, `http://www.stetson.edu/~efriedma/packing.html`

[17] A. Brooke, D. Kendrick and A. Meeraus, GAMS: A user's guide, Release 2.25, The Scientific Press, Redwood City, CA, 1992.

[18] M. Goldberg, The packing of equal circles in a square, *Mathematics Magazine* 43, pp. 24–30, 1970.

[19] L. Greengard and V. Rokhlin, A Fast Algorithm for Particle Simulations, *Comp. Phys.* 73, 1987.

[20] L. Greengard and V. Rokhlin, The rapid evaluation of potential fields in 3 dimensions, *Lecture Notes in Mathematics* 1360, pp. 121–141, 1988.

[21] C. de Groot, R. Peikert and D. Würtz, The optimal packing of ten equal circles in a square, IPS Research Report 90-12, ETH, Zürich, 1990.

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

[23] W. Humphrey, A. Dalke and K. Schulten, VMD -Visual Molecular Dynamics, *J. Molecular Graphics*, vol. 14, pp. 33–38, 1996.

[24] J. Katzenelson, Computational structure of the N-body problem, *SIAM Journal on Scientific and Statistical Computing* 10, pp. 787–815, 1989.

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

[26] C. D. Maranas, C. A. Floudas and P. M. Pardalos, New results in the packing of equal circles in a square, *Discrete Mathematics* 142, pp. 287–293, 1995.

[27] 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.

[28] W. F. Mascarenhas and E. G.Birgin, Using sentinels to detect intersections, *submitted*, 2006 (Available in www.ime.usp.br/~egbirgin/).

[29] N. Mladenović, F. Plastria and D. Urošević, Reformulation descent applied to circle packing problems, *Computers & Operations Research* 32, pp. 2419–2434, 2005.

[30] B. A. Murtagh and M. A. Saunders, MINOS 5.5 User's Guide, Report SOL 83-20R, Systems Optimization Laboratory, Stanford University (revised July 1998).

[31] K. J. Nurmela and P. R. J. Östergård, Packing up to 50 equal circles in a square, *Discrete & Computational Geometry* 18, pp. 111–120, 1997.

[32] E. Specht, `http://www.packomania.com/`

[33] Yu. G. Stoyan, Mathematical methods for geometric design, In: T. M. R. Ellis and O. J. Semenkoc (eds), *Advances in CAD/CAM, Proceedings of PROLAMAT 82*, Leningrad, Amsterdam, pp. 67–86, 1983.

[34] Yu. G. Stoyan and G. N. Yas'kov, Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints, *International Transactions in Operational Research* 5, pp. 45–57, 1998.

[35] Yu. G. Stoyan and G. N. Yas'kov, 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.

[36] Yu. G. Stoyan, G. N Yas'kov and G. Scheithauer, Packing spheres of various radii into a parallelepiped, Preprint MATH-NM-15-2001, TU Dresden, 2001.

[37] G. Wäscher, H. Haussner and H. Schumann, An improved typology of cutting and packing problems, *European Journal of Operational Research*, to appear.

[38] F. Zhao, An $0(N)$ algorithm for three-dimensional n-body simulations, Technical Report, AI-TR-995, MIT Al Lab, Cambridge, MA, 1987.