European Journal of Operational Research 214 (2011) 512–525

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A heuristic for the circle packing problem with a variety of containers

C.O. López, J.E. Beasley *

Mathematical Sciences, Brunel University, Uxbridge UB8 3PH, UK

ARTICLE INFO

Article history:
Received 3 September 2010
Accepted 22 April 2011
Available online 30 April 2011

Keywords:
Circle packing
Formulation space search
Metaheuristic

ABSTRACT

In this paper we present a heuristic algorithm based on the formulation space search method to solve the circle packing problem. The circle packing problem is the problem of finding the maximum radius of a specified number of identical circles that can be fitted, without overlaps, into a two-dimensional container of fixed size. In this paper we consider a variety of containers: the unit circle, unit square, rectangle, isosceles right-angled triangle and semicircle. The problem is formulated as a nonlinear optimization problem involving both Cartesian and polar coordinate systems.

Formulation space search consists of switching between different formulations of the same problem, each formulation potentially having different properties in terms of nonlinear optimization. As a component of our heuristic we solve a nonlinear optimization problem using the solver SNOPT.

Our heuristic improves on previous results based on formulation space search presented in the literature. For a number of the containers we improve on the best result previously known. Our heuristic is also a computationally effective approach (when balancing quality of result obtained against computation time required) when compared with other work presented in the literature.

1. Introduction

The circle packing problem considered here is concerned with arranging a fixed number (n) of identical circles in a container without any overlap. In this paper we consider the two-dimensional variant of the problem, in particular the well-known case when the container is itself a circle. Other cases that are often encountered, and which will also be considered in this paper, are when the container is a square, rectangle, isosceles right-angled triangle or semicircle.

In terms of optimization there are two different (but equivalent) ways to view the problem, namely:

1. To maximise the radius associated with the n circles when the container size (area) is fixed.
2. To minimise the size (area) of the container to accommodate n circles of fixed radius.

Adopting the first viewpoint Fig. 1 is an example of how $n = 7$ circles can be placed with maximum radius into the unit circle (a circle of radius one), the unit square (a square with side one) and a rectangle of length five and width one.

These two viewpoints are in fact equivalent because there exists a one-to-one relationship (based on scaling) between them. Consider, for the purposes of illustrating this equivalence, the container being a circle.

With respect to the first view of the problem, to maximise the radius associated with the n circles when the container size (area) is fixed suppose (without loss of generality) that the container is the unit circle. Fig. 1 shows how $n = 7$ circles can be placed with maximum radius (R, say) into this containing unit circle.

With respect to the second view of the problem the equivalent problem is to minimise the size (area) of the containing circle to accommodate n circles of fixed radius. Suppose (without loss of generality) that the n circles are of unit radius. For the containing circle area is directly proportional to the square of the radius, so minimising area can be achieved by minimising the radius of the containing circle.

If we scale the unit circle solution seen in Fig. 1 by $1/R$ so that the n circles have the required unit radius then we will have a containing circle of radius $1/R$ with $n = 7$ unit circles contained within it. As the solution before scaling was optimal under the first viewpoint, it can be easily seen that the solution after scaling must be optimal under the second viewpoint (since if not we would have a contradiction).

The circle packing problem has a long history and a wide variety of applications. An introduction to its history can be found in Szabó et al. [22]. Industrial applications of the circle packing problem such as: circular cutting, container loading, cylinder packing, facility dispersion and communication networks, facility and dashboard layout, are considered in Castillo et al. [3].

This paper is organised as follows. In Section 2 we give a literature survey as to the circle packing problem and the approach used

* Corresponding author. Tel.: +44 1895 266219; fax: +44 1895 269732.
E-mail addresses: claudia.lopez@brunel.ac.uk (C.O. López), john.beasley@brunel.ac.uk (J.E. Beasley).

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.04.024

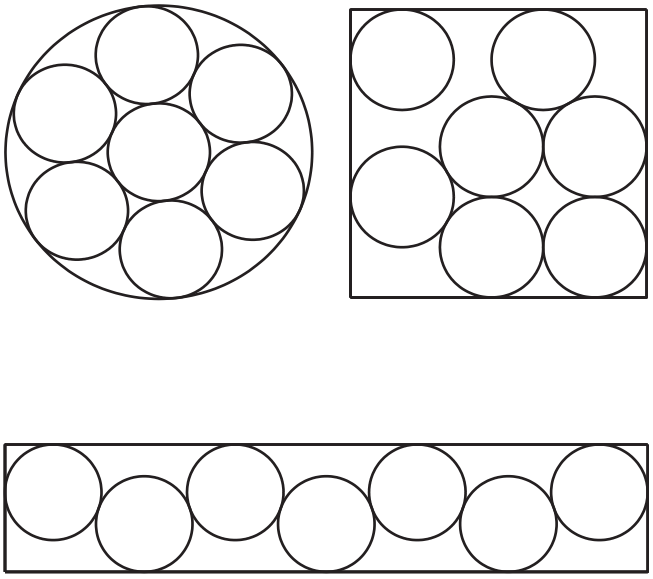

Fig. 1. Examples of the circle packing problem for different shaped containers, from [21].

in this paper, formulation space search. In Section 3 we present our formulation of the circle packing problem and in Section 4 we describe our heuristic. Computational results are presented in Section 5 and in Section 6 our conclusions are given.

2. Literature survey

Hifi and M'Hallah [11] have very recently provided an extremely comprehensive literature review of work relating to the circle packing problem considered in this paper. Hence, for reasons of space, we shall not provide a detailed literature survey here. Rather we will just review the few relevant papers that have appeared since that survey was done. We do however survey in this section papers that are especially relevant to our chosen approach, formulation space search (denoted by FSS).

2.1. Circle packing

Birgin and Gentil [1] consider the problem of packing identical circles of unit radius in a variety of containers (circles, squares, rectangles, equilateral triangles and strips of fixed height). Their approach is focused on examining the contacts that the circles make with each other and the container. In order to determine these contacts they make use of the solution given by the heuristic algorithm of Birgin and Sorbal [2]. To avoid any loss in accuracy (such as occurs in Birgin and Sorbal [2]) they include a correction step in their procedure that adjusts circle centres if any two circles are overlapping.

One issue that arises in their work is the treatment of loose circles (''rattlers'') that may have few contacts. One such loose circle (with zero contacts and potentially many possible positions) can be seen in the top-left of the square in Fig. 1.

Birgin and Gentil [1] give computational results for problems involving up to 50 circles. They compare their results with those given at the Packomania web site of Specht [21]. That website, first established in 1999, is periodically updated with improved results from a continuously running search, as well as with improved results communicated by other researchers. As such this website represents the ''gold standard'' against which results from any approach should be compared. In this paper we do compare the results from our formulation space search heuristic to this ''gold standard''.

Grosso et al. [5] use an approach based on monotonic basin hopping [14,23], extended to population basin hopping [6]. Monotonic basin hopping is essentially a single solution heuristic consisting of the application of a local search descent procedure starting from a perturbation of a locally optimal solution. Population basin hopping is a modification where a population of solutions are maintained. They give computational results for their approach for the problem of packing up to 100 identical circles of unit radius into a containing circle, as well as for the problem of packing up to 162 non-identical circles into a containing circle.

Liu et al. [16] use an approach based on energy landscaping paving [8]. This is an iterative approach whereby the centre of a chosen circle is repositioned at each iteration. They give computational results for their approach for the problem of packing up to 100 identical circles of unit radius into a containing circle, as well as for the problem of packing up to 162 non-identical circles into a containing circle.

Liu et al. [17] extend Liu et al. [16] by introducing gradient descent into the approach. They give computational results for their approach for the problem of packing up to 50 non-identical circles into a containing circle, as well as for the problem of packing up to 50 identical spheres of unit radius into a containing sphere.

Liu et al. [15] make a number of modifications to the algorithm of Liu et al. [17]. They give computational results for their approach for the problem of packing up to 17 non-identical circles into a containing circle, as well as for the problem of packing up to 91 identical circles into a containing circle.

2.2. Formulation space search (FSS)

Formulation space search is based on the observation that when considering a nonlinear optimization problem different formulations of the same problem may have different properties. In particular when using a nonlinear solver with a particular formulation we may reach a stationary point in the solution space. However this point may not be a stationary point with respect to a different formulation of the problem.

Hence, by taking the same point in the solution space, but by utilizing a different formulation, we may be able to use the nonlinear solver to improve the solution. This leads naturally to the idea of switching between formulations and repeating the process until we have a solution that is stationary with respect to all of the formulations considered.

Formulation space search was first proposed by Mladenovic et al. [18] in the context of the circle packing problem. They consider packing identical circles into the unit circle and unit square. They alternate between two nonlinear formulations: one where all circle centres are expressed in Cartesian coordinates; the other where all circle centres are expressed in polar coordinates. They give computational results for their approach (which they call reformulation descent) for problems involving packing up to 100 identical circles into the unit circle and unit square. They also give computational results for approaches that work with a single formulation of the problem.

Mladenovic et al. [19] build upon Mladenovic et al. [18] in several respects. Firstly, instead of having all circle centres expressed in the same coordinate space (Cartesian or polar) they have a mixed strategy, with some circle centres being expressed in Cartesian coordinates, the remainder in polar coordinates. They change (in a systematic fashion) the number of centres expressed in polar coordinates. Secondly, they reduce the number of constraints by ignoring overlap constraints for circles whose centres (at the initial solution, before optimization) are sufficiently far apart. They present results for the problem of packing up to 100 identical circles into the unit circle which show that they improve upon the quality of the results achieved as compared to Mladenovic et al. [18].

Formulation space search is a new and relatively unexplored idea in the literature. It has been applied to only a few problems in the literature additional to circle packing (e.g. timetabling, Kochetov et al. [13]). More discussion as to formulation space search can be found in Hansen et al. [7]. A related approach is variable space search, which has been applied to graph colouring [9,10].

3. Formulation

In this paper we formulate the circle packing problem from the viewpoint of maximising the radius associated with the n circles when the container size (area) is fixed. The circle packing problem can be formulated as a nonlinear optimization problem and a variety of formulations of the problem are given in the literature. Here we adopt a slightly unusual formulation in that we use both Cartesian and polar coordinates. The reason for this is that it will lead naturally to the heuristic we present later.

Below we formulate the problem when the container is the unit circle (circle of radius one) which is centred at the origin of the two-dimensional plane. We discuss in later sections below how this formulation needs to be (slightly) modified for differing containers. Let:

- C be the set of circles whose centres are expressed in Cartesian coordinates, so for circle $i \in C$ its centre is at (x_i, y_i) in Cartesian coordinates.
- P be the set of circles whose centres are expressed in polar coordinates, so for circle $i \in P$ its centre is at (r_i, θ_i) in polar coordinates (where $C \cap P = \emptyset$ and $C \cup P = \{1,\ldots,n\}$).
- Q be the set of all pairs $\{(i,j) \mid i = 1,\ldots,n;\ j = 1,\ldots,n;\ i \neq j\}$
- R be the radius associated with each of the n circles.
- $R_{overlap}$ be an upper bound on R, formally $R_{overlap}$ is the maximum radius that the circles can have before they must overlap due to area considerations, defined here by equating the area of the containing unit circle ($\pi 1^2$) to the total area of the n circles ($n\pi R_{overlap}^2$), so $R_{overlap} = 1/\sqrt{n}$.

Although we have above (using disjoint sets C and P) separated centres expressed in Cartesian and polar coordinates note here that the relationship between the two coordinate systems is that a point (x, y) in Cartesian space has equivalent coordinates (r, θ) in polar space where $x = r\cos(\theta)$ and $y = r\sin(\theta)$. Our formulation of the circle packing problem is:

$$\max \quad R \tag{1}$$

$$\text{subject to } x_i^2 + y_i^2 \leqslant (1-R)^2 \quad \forall\, i \in C, \tag{2}$$

$$r_i \leqslant 1 - R \quad \forall\, i \in P, \tag{3}$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 \geqslant 4R^2 \quad \forall (i,j) \in Q \text{ with } i \in C\ j \in C\ i < j, \tag{4}$$

$$(x_i - r_j\cos(\theta_j))^2 + (y_i - r_j\sin(\theta_j))^2 \geqslant 4R^2 \quad \forall (i,j) \in Q \text{ with } i \in C\ j \in P, \tag{5}$$

$$r_i^2 + r_j^2 - 2r_ir_j\cos(\theta_i - \theta_j) \geqslant 4R^2 \quad \forall (i,j) \in Q \text{ with } i \in P\ j \in P\ i < j, \tag{6}$$

$$-1 \leqslant x_i \leqslant 1 \quad \forall\, i \in C, \tag{7}$$

$$-1 \leqslant y_i \leqslant 1 \quad \forall\, i \in C, \tag{8}$$

$$0 \leqslant r_i \leqslant 1 \quad \forall\, i \in P, \tag{9}$$

$$0 \leqslant \theta_i \leqslant 2\pi \quad \forall\, i \in P, \tag{10}$$

$$0 \leqslant R \leqslant R_{overlap} \tag{11}$$

The objective, Eq. (1), maximises the radius associated with the circles. Eqs. (2) and (3) are the constraints which ensure that every circle is fully inside the container, in this case the unit circle. Notice here that whilst Eq. (2) is a nonlinear equation when expressed in Cartesian form, it is a linear equation, Eq. (3), when expressed in polar form.

Eqs. (4)–(6) ensure that no circles overlap each other. Eq. (4), for example, says that the Euclidean distance between the centres of circle i and circle j, $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ must be at least $2R$ (since each circle is of radius R). Note here that for computational reasons we (as is common in the literature) have squared both sides of this constraint to eliminate the square root. Eqs. (5), (6) are as Eq. (4), but where one or both of the circles has its centre expressed in polar coordinates.

Eqs. (7)–(11) are the limits on the variables. Note here that although Eqs. (7) and (8) can be deduced from Eq. (2), and Eq. (9) can be deduced from Eq. (3), they are given here for completeness. Eq. (10) limits the polar angle, so avoiding a multiplicity of equivalent solution values, $\theta_i + k(2\pi)$ for all integer values of k.

4. Formulation space search heuristic

4.1. Overview

Our FSS heuristic starts from the formulation of the circle packing problem, Eqs. (1)–(11), given above. We switch between different formulations by alternating the sets of circles whose centres are expressed in Cartesian/polar coordinates.

The difficulty with the formulation given above is clearly the number of nonlinear overlap constraints, Eqs. (4)–(6), which number in total $|Q| = n(n-1)/2$. In order to reduce these constraints we introduce additional linear constraints on the range of possible values for centre coordinates. This enables us to deduce whether or not two circles can overlap and hence reduce the size of Q.

One numeric issue that has to be addressed is that the results given at the Packomania web site of Specht [21] are given to a very high degree of decimal place accuracy (30 decimal places). In order to address this issue we have a step in our heuristic to ensure that solutions given by our nonlinear optimization solver (SNOPT which is the acronym for Sparse Nonlinear OPTimizer [4,12]) are corrected to be feasible when expressed with a high degree of accuracy. This is especially important since there are examples in the literature, e.g. [2] as noted in [1], where the results given are invalid due to a lack of sufficient precision.

In this section we present details of the main components of our FSS heuristic, together with pseudocode for it, focusing on the container being the unit circle. We also indicate the (minor) changes to our FSS heuristic that are necessary for differing containers, specifically: the square, rectangle, isosceles right-angled triangle and semicircle.

4.2. Overlap constraints

In order to reduce the number ($|Q|$) of nonlinear overlap constraints we need to reduce the size of Q. This can be done by, for each circle i, having a known point (X_i, Y_i) and not allowing the centre of the circle to move more than Δ (in any direction) from this point (all centres being regarded as being expressed in Cartesian coordinates, for simplicity of presentation). Given these restrictions the constraints on the centre coordinates of circle i become $X_i - \Delta \leqslant x_i \leqslant X_i + \Delta$ and $Y_i - \Delta \leqslant y_i \leqslant Y_i + \Delta$, i.e. $x_i \in [X_i - \Delta, X_i + \Delta]$ and $y_i \in [Y_i - \Delta, Y_i + \Delta]$.

Consider the nonlinear overlap constraint Eq. (4). This involves the term $(x_i - x_j)^2$ for two circles i and j, whose centres now have a restricted range, so we can deduce the minimum value that this term can take. Given the ranges, $x_i \in [X_i - \Delta,\ X_i + \Delta]$ and $x_j \in [X_j - \Delta,\ X_j + \Delta]$ these two ranges overlap (in other words x_i can equal x_j) if and only if one of the end points lies in the other range. In other words these ranges overlap if:

$$X_i - \Delta \in [X_j - \Delta, X_j + \Delta] \text{ or } X_i + \Delta \in [X_j - \Delta, X_j + \Delta] \text{ or } X_j - \Delta \in [X_i - \Delta, X_i + \Delta] \text{ or } X_j + \Delta \in [X_i - \Delta, X_i + \Delta]. \quad (12)$$

If the ranges overlap then the minimum value of $(x_i - x_j)^2$ is zero. If the ranges do not overlap then the minimum value of $(x_i - x_j)^2$ will occur when x_i and x_j take values at the end of their ranges, so in this case the minimum value of $(x_i - x_j)^2$ will be:

$$\min\{[(X_i - \Delta) - (X_j - \Delta)]^2, [(X_i - \Delta) - (X_j + \Delta)]^2, [(X_i + \Delta) - (X_j - \Delta)]^2, [(X_i + \Delta) - (X_j + \Delta)]^2\}. \quad (13)$$

In a similar manner we can compute the minimum value of the $(y_i - y_j)^2$ term in Eq. (4) as zero if:

$$Y_i - \Delta \in [Y_j - \Delta, Y_j + \Delta] \text{ or } Y_i + \Delta \in [Y_j - \Delta, Y_j + \Delta] \text{ or } Y_j - \Delta \in [Y_i - \Delta, Y_i + \Delta] \text{ or } Y_j + \Delta \in [Y_i - \Delta, Y_i + \Delta] \quad (14)$$

and as:

$$\min\{[(Y_i - \Delta) - (Y_j - \Delta)]^2, [(Y_i - \Delta) - (Y_j + \Delta)]^2, [(Y_i + \Delta) - (Y_j - \Delta)]^2, [(Y_i + \Delta) - (Y_j + \Delta)]^2\} \quad (15)$$

otherwise.

Fig. 2 illustrates the above graphically. In this figure we have two known points, (X_i, Y_i) and (X_j, Y_j), and the square surrounding each point indicates the range within which the centre's, (x_i, y_i) and (x_j, y_j), for circles i and j can be located. The minimum value of $(x_i - x_j)^2$ is determined by squaring of the length of the horizontal line A to B joining the left-hand square to the right-hand square, namely $[(X_i - \Delta) - (X_j + \Delta)]^2$. The minimum value of $(y_i - y_j)^2$ is zero (since the ranges overlap).

Now if the sum of these minimum values for $(x_i - x_j)^2$ and $(y_i - y_j)^2$ is $\geqslant 4R^2$ (compare Eq. (4)) then there is no need to impose a nonlinear overlap constraint for circles i and j, because they cannot overlap each other by definition given the ranges imposed.

With reference to Fig. 2 then given the square ranges allowed for circles i and j the best possible position for these circles (each of radius R) in any attempt to have them overlap would be centre them at A and B, respectively. If the distance from A to B is $\geqslant 2R$ (equivalently the squared distance is $\geqslant 4R^2$) then there is no need to impose a nonlinear overlap constraint for circles i and j.

Clearly we do not know R, since that is a variable in the optimization, but we can replace it by any valid upper bound which we do know, here we use $R_{overlap}$. Hence if the sum of these minimum values is $\geqslant 4R^2_{overlap}$ circles i and j cannot overlap and the pair (i, j) need not be included in Q. Conversely if this sum is $< 4R^2_{overlap}$ the pair (i,j) does need to be included in Q.

Hence, in summary here, we take all pairs (i, j) of circles, do the calculation (Eqs. (12)–(15)) outlined above (which is computationally an easy task) and thereby identify the pairs of circles that do need to be included in Q.

We denote the above procedure as $OverlapSet(X, Y, \Delta, R_{overlap})$ and it returns the set Q of pairs of circles.

We would comment here that other authors in the literature have also attempted to reduce the number of overlap constraints. Mladenovic et al. [19] reduce the number of constraints by ignoring overlap constraints for circles whose centres (at the initial solution, before optimization) are sufficiently far apart. The disadvantage of their approach is that they do not constrain the movement of any circle centre in the optimization (unlike our approach above). An important difference therefore between our paper and earlier formulation space search work for circle packing (such as Mladenovic et al. [18] who did not reduce overlap constraints, and Mladenovic et al. [19]) is the use of constraints that limit the range of movement for circle centres.

Birgin and Gentil [2] use a distinctly different approach to reduce the number of overlap constraints. They first replace the $n(n-1)/2$ overlap constraints, Eqs. (4)–(6), by a single constraint which is the sum of $n(n-1)/2$ nonlinear terms. Then, in order to reduce the computational effort relating to evaluating this constraint (which they may need to do many times during the course of a nonlinear solution algorithm), they consider a partition of the container into regions in such a way that circles whose centres are not in the same (or an adjacent) region cannot contribute to the constraint. Essentially therefore their approach consists of replacing $n(n-1)/2$ overlap constraints, each of which is computationally inexpensive to evaluate numerically (requiring O(1) operations), by a single constraint that computationally requires $O(n(n-1)/2)$ operations to evaluate.

4.3. Optimization problem

In our FSS heuristic the sets C and P are of approximately equal size. Limited computational experience indicated that, whilst we impose range constraints on all circles in terms of computing Q (as described above), it was sufficient purely to impose range constraints on circles whose centres are expressed in Cartesian coordinates in terms of the optimization. This allows additional flexibility for positioning of circles whose centres are expressed in polar coordinates. Although this might result in some circles overlapping in the optimization solution any such overlaps are corrected after solution as explained in Section 4.4 below.

Hence (at each iteration) the nonlinear optimization problem that we solve is optimize (1) subject to (2)–(11) and

$$X_i - \Delta \leqslant x_i \leqslant X_i + \Delta \quad Y_i - \Delta \leqslant y_i \leqslant Y_i + \Delta \quad \forall i \in C. \quad (16)$$

We denote this nonlinear optimization problem as $NLP(x^0, y^0, C, P, Q, X, Y, \Delta, R_{overlap})$ where x^0 and y^0 represent an initial solution for our optimization solver SNOPT. In the computational results reported later below we set an initial solution for this solver by randomly generating $x_i^0 \in (X_i - \Delta, X_i + \Delta)$ and $y_i^0 \in (Y_i - \Delta, Y_i + \Delta)$ $(i = 1, \ldots, n)$.

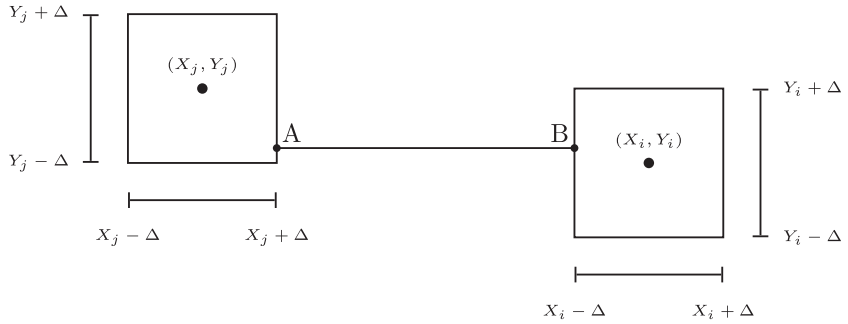

Fig. 2. Overlap constraint example.

4.4. Correction

As mentioned previously above one numeric issue that has to be addressed is that we need to ensure that solutions given by our nonlinear optimization solver when we solve $NLP(x^0, y^0, C, P, Q, X, Y, \Delta, R_{overlap})$ are correct when expressed with a high degree of decimal place accuracy. Let the solution from the solver be $[(x_i, y_i), i = 1,\ldots,n]$ when expressed in Cartesian coordinates.

Our correction procedure has two parts. First we need to ensure that the solution (i.e. the circle centres) are inside the container. Then we need to find the radius that corresponds to the given circle centres such that the circles do not overlap and are fully inside the container. The procedure we adopted is set out below.

We first need to ensure that the solution $[(x_i, y_i), i = 1,\ldots,n]$ from the solver is inside the unit circle. So if $x_i^2 + y_i^2 > 1$ (for any i) we (arbitrarily) set $x_i = x_i^0$ and $y_i = y_i^0$ (if that point (x_i^0, y_i^0) is itself inside the unit circle). If (x_i^0, y_i^0) is itself outside the unit circle we first express (x_i, y_i) in polar form; keep the polar angle and (arbitrarily) set the polar radius to be a random number drawn from (0,0.99); then re-express the centre in Cartesian form.

Now we set:

$$R = \min\left\{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}/2 \,\middle|\, i = 1,\ldots,n;\ j = 1,\ldots,n; i < j\right\} \tag{17}$$

to ensure that the radius R is such so as to prevent circles from overlapping and then set:

$$R^* = \min\left\{R, \min\left\{1 - \sqrt{x_i^2 + y_i^2} \,\middle|\, i = 1,\ldots,n\right\}\right\} \tag{18}$$

to ensure that all the circles are fully inside the unit circle. Here R^* is the maximum radius associated with the optimiser solution and, by the nature of the procedure adopted above, R^* must be associated with a feasible solution to the problem. Note here that the value of R^* may (because of the procedure above) differ (albeit possibly only slightly) from the radius value returned by the solver as the maximum possible radius.

We denote the above procedure as $Correction(x^0, y^0, x, y)$ and it returns a value for radius that must (by construction) be associated with a feasible solution. In our computational implementation of this correction procedure we used a MATLAB function called vpa (which is the acronym for variable precision arithmetic) that gives as many digits of accuracy as we desire.

4.5. FSS pseudocode

Below we show the pseudocode for our FSS heuristic. Our heuristic needs a value for Δ which, Section 4.2, restricts the movement of circle centres and is used in the determination of the set of circles Q that can potentially overlap. Based on limited computational experience with five arbitrarily selected values of n we set $\Delta = \frac{2}{3}$(current radius R^*).

Our pseudocode is shown in Algorithm 1. In that pseudocode R_{best} is the maximum radius encountered over all iterations and t is an iteration counter. We define the sets C and P by randomly allocating each circle i ($i = 1,\ldots,n$) either to C or to P. We switch between different formulations by alternating the sets of circle whose centres are expressed in Cartesian/polar coordinates. After solution the circle centres become the positions from which we restrict movement by the factor Δ. Here if the current radius R^* is very small we use $R_{overlap}$ to set a value for Δ.

The first initial solution (X_i, Y_i) $i = 1,\ldots,n$ is generated by randomly placing n circles in the container. Here, for the unit circle, this is done using: choose r_i uniformly from [0,1]; choose θ_i uniformly from $[0,2\pi]$; express the centre in Cartesian coordinates (X_i, Y_i). Limited computational experience indicated that an appropriate termination condition was to terminate the heuristic once eighty iterations had been performed (i.e. when $t = 80$).

In the computational results reported later below we replicated our FSS heuristic 25 times, each time from a different initial starting solution. In such cases we (naturally) report the best (maximum) radius found over all replications.

Algorithm 1. Formulation space search pseudocode

Initialisation: $\Delta \leftarrow \frac{2}{3}R_{overlap}$ $R_{best} \leftarrow 0$ $t \leftarrow 0$
Randomly generate an initial solution (X, Y)
Iterative process:
WHILE not termination condition **do**
  $Q \leftarrow OverlapSet(X, Y, \Delta, R_{overlap})$ {find the overlap set Q}
  Solve $NLP(x^0, y^0, C, P, Q, X, Y, \Delta, R_{overlap})$ to give (x, y)
  $R^* \leftarrow Correction(x^0, y^0, x, y)$ {correct the radius}
  $R_{best} \leftarrow \max\{R_{best}, R^*\}$ {update R_{best}}
  if $R^* \leqslant 0.001$ $\Delta \leftarrow \frac{1}{10}R_{overlap}$ else $\Delta \leftarrow \frac{2}{3}R^*$ {update Δ}
  $t \leftarrow t + 1$ {update iteration counter}
  $C \leftarrow P$ $P \leftarrow \{1,\ldots,n\} - C$ {swap the sets C and P}
  $(X,Y) \leftarrow (x, y)$ {set (X, Y) to the current solution}
end while

The algorithmic details given in Algorithm 1 are based on the container being the unit circle. For different containers some minor changes are needed as outlined below. For all of the containers considered below some changes are generic in nature and so (for reasons of space) are not given in detail for individual containers. These are:

- The first initial solution (X_i,Y_i) $i = 1,\ldots,n$ is generated by randomly placing n circles in the container.
- With respect to the correction procedure $Correction(x^0, y^0, x, y)$ we need to ensure that the circle centres $[(x_i, y_i), i = 1,\ldots,n]$ from the optimiser are inside the container. Our computational experience is that the optimiser solution being outside the container happens relatively infrequently and so we do not detail here the procedures we adopt (essentially move any circle centres that are outside the container inside), merely note that procedures to account for this situation must be in place.

We would comment here that one of the advantages of our heuristic, as the sections below will indicate, is that it is easy to adapt for a variety of different containers.

4.6. Rectangular and square containers

For the case of a rectangular container we consider a rectangle of length L, width W, centred at the Cartesian origin. If the container is the square then (obviously) $L = W$, where the case of the unit square corresponds to $L = W = 1$. The modifications needed to change from the container being the unit circle (as considered above) are relatively minor. We detail them below.

4.6.1. Formulation changes

$R_{overlap}$, the upper bound on the radius, is now defined using $n\pi R_{overlap}^2 = LW$, so $R_{overlap} = \sqrt{LW/n\pi}$.

Eqs. (2) and (3) that ensure that the circles are inside the container must be changed. Eq. (2) is replaced by:

$$\begin{aligned} -L/2 \leqslant x_i + R \leqslant L/2 \quad & -W/2 \leqslant y_i + R \leqslant W/2 \ \forall i \in C, \\ -L/2 \leqslant x_i - R \leqslant L/2 \quad & -W/2 \leqslant y_i - R \leqslant W/2 \end{aligned} \tag{19}$$

The polar replacement of Eq. (3) is simply as Eq. (19), but with Cartesian variables replaced by their polar equivalents.

With respect to the constraints, Eqs. (7)–(11) setting variable limits then in the rectangular case Eqs. (7) and (8) are no longer valid and are deleted. Eq. (9), which bounds the polar radius, is replaced by:

$$0 \leqslant r_i \leqslant \sqrt{(L/2)^2 + (W/2)^2} \quad \forall i \in P. \tag{20}$$

4.6.2. FSS heuristic changes

With respect to the correction procedure *Correction*(x^0, y^0, x, y) and finding the radius that corresponds to the given circle centres such that the circles do not overlap and are fully inside the container then Eq. (17), which ensures that the circles do not overlap, is as before. Eq. (18), to ensure that the circles are fully inside the container, is replaced by

$$R^* = \min\{R, \min\{L/2 - |x_i|, W/2 - |y_i|\| \ i = 1, \ldots, n\}\}. \tag{21}$$

4.7. Triangular container

For the case of the triangular container we consider an isosceles right-angled triangle (as in Specht [21]) where the two equal sides are each of length L and they join at the origin of the Cartesian plane. Their intersections with the third side of the triangle are at coordinates $(0,L)$ and $(L,0)$.

4.7.1. Formulation changes

$R_{overlap}$, the upper bound on the radius, is now defined using $n\pi R^2_{overlap} = L^2/2$, so $R_{overlap} = L/\sqrt{2n\pi}$.

Eqs. (2) and (3) that ensure that the circles are inside the container must be changed. Eq. (2) is replaced by:

$$x_i \geqslant R \quad y_i \geqslant R \quad x_i + y_i + \sqrt{2}R \leqslant L \quad \forall i \in C. \tag{22}$$

Eq. (3) is replaced by the equivalent polar expression of (22).

With respect to the constraints, Eqs. (7)–(11), setting variable limits then Eqs. (7) and (8) are no longer valid and are deleted. Eqs. (9) and (10) are replaced by

$$0 \leqslant r_i \leqslant L \quad 0 \leqslant \theta_i \leqslant \pi/2 \quad \forall i \in P \tag{23}$$

4.7.2. FSS heuristic changes

Eq. (18), to ensure that the circles are fully inside the container, is replaced by

$$R^* = \min\left\{R, \min\left\{x_i, y_i, \{(L - x_i - y_i)/\sqrt{2}\} | i = 1, \ldots, n\right\}\right\}. \tag{24}$$

4.8. Semicircular container

Here we consider the semicircular container as the upper half of the unit circle centred at the origin of the Cartesian plane.

4.8.1. Formulation changes

$R_{overlap}$, the upper bound on the radius, is now defined using $n\pi R^2_{overlap} = \pi 1^2/2$, so $R_{overlap} = 1/\sqrt{2n}$.

To ensure that no circle lies outside the container the following constraint must be added

$$y_i \geqslant R \quad \forall i \in C. \tag{25}$$

Eq. (8) is replaced by $0 \leqslant y_i \leqslant 1 \quad \forall i \in C$ and Eq. (10) is replaced by $0 \leqslant \theta_i \leqslant \pi \ \forall i \in P$.

4.8.2. FSS heuristic changes

Eq. (18), to ensure that the circles are fully inside the container, is replaced by

$$R^* = \min\left\{R, \min\left\{y_i, \{1 - \sqrt{x_i^2 + y_i^2}\} \middle| i = 1, \ldots, n\right\}\right\}. \tag{26}$$

5. Computational results

The results given in this section for our FSS heuristic were produced using an Intel Core 2 pc (2.26 GHz, 4 GB RAM). Our heuristic was coded in MATLAB 7.0 and as a subroutine used the nonlinear optimization solver SNOPT [4,12].

In this section we present results for six different containers: the unit circle, the unit square, a rectangle of length five and width one, a rectangle of length 10 and width one, a right-angled isosceles triangle where both equal sides are of length one and a semicircle of radius one. We compare our results against the results given at the Packomania web site of Specht [21], as well as against the results presented by others in the literature.

5.1. Comparison with the results given at the Packomania web site of Specht [21]

The results presented in Table 1 correspond to three cases: when the container is the unit circle, the unit square and a rectangle of length $L = 5$ and width $W = 1$. In that table we give (for each value of n):

- The best known result (radius) as from Specht [21].
- The percentage deviation for our FSS heuristic calculated using $100(R_{bk} - R_{best})/R_{bk}$, where R_{bk} is the best known radius from [21] and R_{best} is the best (maximum) radius found by our FSS heuristic.
- The total time taken (over all iterations/replications) by our FSS heuristic (in seconds).

Note here that the best known results given at [21], seen in Table 1, are periodically updated and as such they represent the ***accumulated best results from all previous work***. Hence in comparing our FSS heuristic against these results we are implicitly comparing (at least with respect to quality of result) our heuristic against all previous work.

The results given at the Packomania web site of Specht [21] are given with a high degree of precision (decimal places). Here we have given results to twelve decimal places of accuracy (but note that in Section 4.4 we can, using MATLAB, compute results to any desired level of accuracy).

Table 2 presents the same information as Table 1, but for the cases when the container is a rectangle of length $L = 10$ and width $W = 1$, a right-angled isosceles triangle with equal sides of length $L = 1$ and a semicircle of radius one.

The zero percentage deviations seen for some cases associated with $n = 250, 500$ in Tables 1 and 2 arise because no solutions have been reported to date at [21] for problems of this size. In these cases the values given as the best known result are the radii given by our FSS heuristic.

Considering Tables 1 and 2 it is clear that (on average) our FSS heuristic produces low percentage deviations. Computationally it is clear that the two rectangular container cases require the most time (on average).

5.2. Improvements on best known results

In Table 2 it can be seen that for $n = 30, 60, 65, 90, 100$ when the container is the right-angled isosceles triangle, a negative percentage deviation is given. This implies that our FSS heuristic, for these cases, found better solutions than had been found and reported to date in Specht [21]. A similar situation occurs when $n = 50, 60, 65, 70, 80, 85, 95, 100, 125, 150, 175$ when the container is a semicircle.

Table 1
FSS results for the unit circle (UC), unit square (US) and rectangle of length 5, width 1 (R 5 × 1).

n	Best known			% Deviation from best known			Total time in seconds		
	UC	US	R 5 × 1	UC	US	R 5 × 1	UC	US	R 5 × 1
10	0.262258924190	0.148204322565	0.061850317545	0.000000000734	0.000000621851	0.000000305559	41	61	85
15	0.221172539086	0.127166547515	0.055050511413	0.000000000607	0.000000777009	0.000006940783	37	50	63
20	0.195224011019	0.111382347512	0.050000000000	0.000000004298	0.000000260200	0.000000000830	47	65	48
25	0.173827661421	0.100000000000	0.042269960955	0.000000018720	0.000000038086	0.000009718337	65	78	86
30	0.161349109065	0.091671057986	0.039233338586	0.000000025979	0.000000137572	0.000039122479	82	99	108
35	0.149316776635	0.084290712122	0.037613330478	0.000000040758	0.000002663682	0.000003367028	112	155	159
40	0.140373604203	0.079186752517	0.035938232560	0.000000006896	0.000002170115	0.000178274824	109	169	171
45	0.132049594252	0.074727343687	0.033333333333	0.000001395740	0.000032714746	0.000000000504	151	237	176
50	0.125825489530	0.071377103865	0.031176464698	0.000000000578	0.000006738751	0.000049629119	173	276	220
55	0.121786324528	0.068055360559	0.029740240387	0.000000000012	0.000002074958	0.000038511750	219	286	251
60	0.115657480133	0.065030412648	0.028871754277	0.000000001134	0.000004672682	0.000007587388	246	350	309
65	0.110896743723	0.063203957072	0.028201133568	0.000000003455	0.000021480102	0.000015234239	289	487	401
70	0.107001616606	0.060596693631	0.027777800548	0.000000223052	0.000366546626	0.000004355026	347	517	495
75	0.103390915666	0.058494535281	0.026341555892	0.000006073472	0.006893494284	0.002590920313	408	621	608
80	0.100319499416	0.057370684147	0.025130533303	0.000029120955	0.000000153060	0.105048177729	479	750	692
85	0.098395063693	0.055680181768	0.024324423414	0.000000003464	0.004203366373	0.000082010404	621	978	771
90	0.094822059587	0.053749948307	0.023720963436	0.000000047733	0.000076671301	0.000451444998	586	1179	927
95	0.092249177761	0.052420366496	0.023239142833	0.000066576254	0.002243175022	0.000284289214	693	1423	1110
100	0.090235200288	0.051401071774	0.022881112120	0.000176232121	0.002496218249	0.000082619869	810	1317	1259
125	0.080852343329	0.045978336543	0.020322511578	0.000012372689	0.080402416541	0.131617836613	1325	2452	2263
150	0.074289754450	0.042145465901	0.018961762331	0.001774641536	0.004007236044	0.000001807992	2226	4107	3942
175	0.068792158147	0.039061099168	0.017327936687	0.013553921619	0.037072062171	0.063223806328	3183	6374	6717
200	0.064669354186	0.036612798904	0.016412404337	0.005922426972	0.297451068212	0.000258283549	4815	8790	9463
250	0.057927485801	0.032876318472	0.014634379134	0.000909771442	0.086609551716	0.195706941504	9519	18111	20465
500	0.041437143525	0.023445486440	0.010467091580	0.294917207409	0.958434336421	0	86117	133443	241386
Average				0.012694804705	0.059213225831	0.019988047455	4508	7295	11687

Table 2
FSS results for rectangle of length 10 width 1 (R 10 × 1), a right-angled isosceles triangle (RIT) and the semicircle (Semi).

n	Best known			% Deviation from best known			Total time in seconds		
	R 10 × 1	RIT	Semi	R 10 × 1	RIT	Semi	R 10 × 1	RIT	Semi
10	0.050000000000	0.106222361897	0.188262725163	0.000000000791	0.000000000004	0.000000000029	40	25	26
15	0.035985190969	0.087610065690	0.158625813391	0.000030458043	0.000000000019	0.000000000000	63	35	34
20	0.031090744863	0.076378991823	0.139095851256	0.000006929841	0.000000002581	0.000000000617	76	49	49
25	0.028853797639	0.068552444544	0.124584600939	0.000068789016	0.000000048185	0.000000000002	140	66	61
30	0.027652934016	0.063061266976	0.113965603325	0.000068201937	−0.001165535662	0.000000065539	123	99	82
35	0.026935896591	0.058702760457	0.105783385013	0.000021893812	0.000000005577	0.000000025327	141	127	104
40	0.025000000000	0.055284785872	0.099066823854	0.000000013247	0.000001650712	0.000000002691	122	147	115
45	0.022652281116	0.052151704853	0.093666768138	0.002067414584	0.000000483758	0.000000755100	167	186	139
50	0.021334674870	0.049782888809	0.089091135483	0.000336133678	0.000000319942	−0.002968787898	208	204	172
55	0.020409946124	0.047519033968	0.085034616682	0.000202835652	0.000000027816	0.000000000243	268	270	204
60	0.019735625997	0.045598212366	0.081483098139	0.000324334792	−0.006967705670	−0.027920773791	325	329	236
65	0.019243525573	0.043922929492	0.078766537837	0.000162313593	−0.018486414074	−0.000664263633	445	376	289
70	0.018840914067	0.042484939929	0.076054925571	0.000278522656	0.000007711674	−0.008030258153	538	422	338
75	0.018548177253	0.041032850591	0.073302177704	0.000302642166	0.000002708098	0.000055006140	692	617	388
80	0.018340582007	0.039844578067	0.071317546646	0.000003090232	0.000023146464	−0.004662584738	817	610	472
85	0.017386101597	0.038664292458	0.069234787247	0.058162962322	0.028203667502	−0.002975050491	1038	704	608
90	0.016690752817	0.037690460705	0.067565909065	0.129024971802	−0.005324986587	0.000000051773	1241	799	695
95	0.016150028511	0.036698643855	0.065576922261	0.174914864812	0.011275376312	−0.063255319818	1333	942	739
100	0.015663144828	0.035803370275	0.064024532014	0.000352847096	−0.000309141609	−0.004580062767	1640	1379	804
125	0.014317183180	0.032204794946	0.057573421546	0.000286694222	0.026348941178	−0.004229064951	3192	1875	1462
150	0.013241342248	0.029530381065	0.052710530051	0.046662355718	0.016861113424	−0.047905776162	6195	3132	2374
175	0.012065933672	0.027421932141	0.048878016771	0.433602673350	0.130313787041	−0.099125966251	11735	5202	3891
200	0.011479462700	0.025690735580	0.045925579503	0.117472679141	0.038518284457	0.019425121464	19004	7533	5848
250	0.010115291778	0.023050433323	0.041124237375	0	0	0.085800210527	56617	14942	11126
500	0.005734874928	0.016403344552	0.029293578377	0	0	0	218047	113086	83242
Average				0.038574144900	0.008772139646	−0.006441466768	12968	6126	4540

Fig. 3 shows the solutions associated with Table 2 for $n = 30$ when the container is the right-angled isosceles triangle and for $n = 50$ when the container is the semicircle.

Tables 1 and 2 only give results for certain values of n. In order to ascertain whether, or not, our FSS heuristic could discover new and improved solutions for other values of n we used it to solve for values of $n = 2,3,\ldots,150$ for all of the six containers (unit circle; unit square; rectangle of length $L = 5$ and width $W = 1$; rectangle of length $L = 10$ and width $W = 1$; right-angled isosceles triangle with equal sides of length $L = 1$; semicircle of radius one) considered in Tables 1 and 2.

For reasons of space we do not present here the detailed results obtained when considering these six different containers. Rather we present a summary of our results as in Table 3. For computational reasons we did not venture above $n = 150$, other than for the specific values of n shown in Tables 1 and 2. Obtaining the

Table 3
FSS improved results for $2 \leqslant n \leqslant 150$.

	Unit circle (UC)	Unit square (US)	Rectangle 5×1	Rectangle 10×1	Triangle (RIT)	Semicircle (Semi)
Number of improved cases	–	–	9	6	18	69
Average % deviation for improved cases	–	–	−0.026620256247	−0.021473329600	−0.005939179656	−0.031634857204
Values of n for improved cases	–	–	37 49 74 83 101 131 132 141 148	39 43 76 82 109 139	13 27 30 36 44 48 56 57 60 61 65 68 90 92 100 111 116 130	19 37 41 46 47 49 50 52 53 54 60 64 65 66 68 70 71 74 79 80 81 82 83 85 86 87 88 91 92 93 94 95 97 98 100 104 105 106 107 108 112 113 115 116 118 119 120 121 122 123 124 125 127 130 131 132 133 134 135 136 140 141 142 144 146 147 148 149 150
Average % deviation all values of n	0.0111	0.0228	0.0263	0.0242	0.0227	−0.0071
Average time (seconds) all values of n	637	1115	1098	1518	869	695

results seen in Table 3 required, in total, approximately 246 h (just over 10 days) of computation. In Table 3, for $2 \leqslant n \leqslant 150$, we show:

- The number of improved cases (an improved case being one where we improved on the best known result reported in Specht [21]).
- The average % deviation from (previously) best known for these improved cases.
- The values of n for these improved cases.
- The average % deviation (from previously best known result) over all the n values considered.
- The average time taken in seconds (averaged over all the n values considered).

It can be seen from Table 3, for example, that the rectangle of length 5 and width 1 has 9 values of n for which an improved result was found; these corresponded to $n = 37, 49, 74, 83, 101, 131, 132, 141, 148$; for these values of n the average percentage deviation was −0.026620256247 (each individual percentage deviation also being negative as it corresponds to an improved result); the average percentage deviation for $2 \leqslant n \leqslant 150$ was 0.0263 and the average time was 1098 seconds.

Fig. 4 shows the solutions associated with Table 3 for $n = 37$ when the container is the rectangle of length 5 and width 1 and for $n = 39$ when the container is the rectangle of length 10 and width 1.

Considering Table 3 it is clear that the FSS heuristic we have presented in this paper is able to produce results better than previously best known results. Since Specht [21] is updated with results from all authors (indeed some of the results reported in this paper have already been communicated to Specht [21] and can be seen on the Packomania web site) this is a significant achievement.

Whilst, for the unit circle and unit square, our FSS heuristic has been unable to out-perform the best results found over all previous work (there being no values of n in $2 \leqslant n \leqslant 150$ for which we produced a better quality result), the average percentage deviation of our FSS heuristic for those cases is low, 0.0111% for the unit circle, 0.0228% for the unit square, as is the time taken per value of n (on average approximately 11 min for the unit circle, 19 min for the unit square). Moreover, as can be seen from the algorithmic description given above as to the modifications needed to deal with different containers, one advantage of our FSS heuristic is that it is easy to adapt for a variety of different containers.

We would stress here that Tables 1–3 do not compare our FSS heuristic with individual papers presented in the literature, rather they compare it with the best results found over all such work. We later below make individual comparisons between our FSS heuristic and other algorithms presented in the literature.

5.3. Alternative strategies

The results presented above are based on one particular algorithm. Clearly if we vary the algorithm (even in the extreme change just a single parameter value) we may get different, perhaps better, results. In this section we outline the different FSS strategies that we explored to see if better results could be obtained.

The key algorithmic strategy steps in our FSS heuristic are:

- How we initially separate the n circles into the set C whose centres are expressed in Cartesian coordinates (any circles not in C being in P so having their centres expressed in polar coordinates).
- How we update C and P at each iteration of the algorithm.

In order to explore the influence of these steps on the results obtained we considered three different basic strategies. These strategies depend upon a parameter m, as detailed below, and we examined $m = \frac{n}{3}, \frac{n}{4}, \frac{n}{5}, \frac{2n}{3}, \frac{2n}{4} = \frac{n}{2}, \frac{2n}{5}, \frac{3n}{4}, \frac{3n}{5}, \frac{4n}{5}$. This examination of alternative strategies was (for computational reasons) restricted to the unit circle case and for values of $n = 10, 15, \ldots, 100$.

Strategy 1 is called ***fix and swap FS*** (m). It consists of fixing the initial cardinality of C to m, where we start by randomly allocating m circles to C. At each iteration we update C and P by swapping them ($C \leftarrow P$, $P \leftarrow \{1, \ldots, n\} - C$). In this strategy the cardinality of C will alternate, first m, then $n - m$, then m, etc.

Strategy 2 is called ***fix and random FR*** (m). It consists of fixing the initial cardinality of C to m, where we start by randomly allocating m circles to C. At each iteration we update C by randomly choosing m circles to be in C ($P \leftarrow \{1, \ldots, n\} - C$). In this strategy the cardinality of C will always be m.

Strategy 3 is called ***swap outer SO*** (m). It consists of fixing the initial cardinality of C to m, where we start by randomly allocating m circles to C. At each iteration we update C and P by swapping only those circles i whose distance from the origin *(0,0)* of the unit circle to their centre is greater than 0.75 (i.e. $\sqrt{x_i^2 + y_i^2} \geqslant 0.75$). In this strategy the cardinality of C will vary. Here we used 0.75 as it divides the unit circle into two portions of (approximately) equal area, an inner area where the centre coordinates are left unchanged at each iteration and an outer area where they are swapped.

Whilst there are (potentially) an large number of possible strategies that can be examined the three considered here seemed to us to capture different elements:

- in *fix and swap FS(m)* we have a fixed alternating cardinality and circles swap between Cartesian and polar coordinates;
- in *fix and random FR(m)* we have a fixed cardinality for C, but circles are randomly allocated to C at each iteration;
- in *swap outer SO(m)* we use the location of a circle to govern whether or not its coordinate representation changes (and hence C has variable cardinality).

Note here that the results presented above in Tables 1–3 and Figs. 3 and 4 are for a fix and swap strategy FS(M), but with M generated by randomly allocating each circle either to C or to P (so M will be randomly distributed around $n/2$). With three basic strategies, each with nine values for $m\left(\frac{n}{3}, \frac{n}{4}, \frac{n}{5}, \frac{2n}{3}, \frac{n}{2}, \frac{2n}{5}, \frac{3n}{4}, \frac{3n}{5}, \frac{4n}{5}\right)$, we have 27 different strategies to compare with our existing strategy FS(M).

As these strategies, e.g. FR(m), may involve the use of random numbers we adopted a statistical hypothesis testing approach to comparing them against our existing strategy in order to judge whether, or not, there was sufficient statistical evidence to conclude that an alternative strategy was better than our existing strategy.

In our hypothesis testing the null hypothesis was:

H0: The average percentage deviation from our current strategy is equal to the average percentage deviation given by a specific alternative strategy.

The alternative hypothesis was:

H1: The average percentage deviation from our current strategy is greater than the average percentage deviation given by a specific alternative strategy (equivalently that the alternative strategy gives a lower percentage deviation than our current strategy).

This is a one-sided hypothesis test. If we reject H0 (accept H1) then we (statistically) have evidence that there exists a better strategy than our current strategy.

We could use a paired t-test here, but since that technically assumes normality, we chose to use a Wilcoxon signed rank test. We computed the p-values for our one-sided hypothesis test (using the R language for statistical computing). For those unfamiliar with hypothesis testing the p-value is the probability that the result we observe would occur by chance if the null hypothesis was true. If the p-value is small then this provides evidence that the null hypothesis H0 is not true and so the alternative hypothesis H1 should be accepted. In judging whether the p-value is small the standard approach is to compare against values such as 0.05 and 0.01 (significance levels of 5% and 1%, respectively).

For reasons of space we do not present here the p-values we found. However for each of our 27 different strategies the

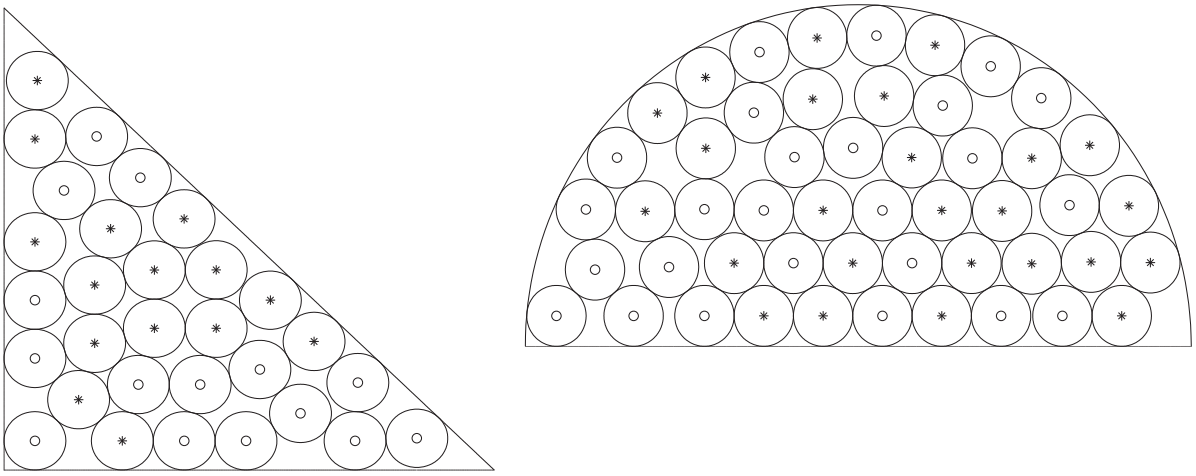

Fig. 3. Improved solutions for $n = 30$ circles in a right-angled isosceles triangle and for $n = 50$ circles in a semicircle. Circles shown with the centre as a small circle are in P, circles shown with the centre as an asterisk are in C. Radius for $n = 30$ circles in the right-angled isosceles triangle from [21] is 0.0630612669763349, the radius shown here is 0.0630620019778907. Radius for $n = 50$ circles in a semicircle from [21] is 0.0890911354834337, the radius shown here is 0.089093780410282.

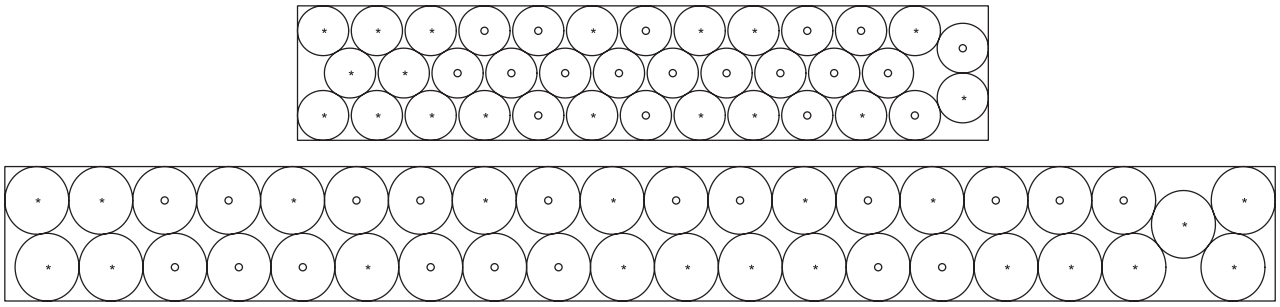

Fig. 4. Improved solutions for $n = 37$ circles in a rectangle of length 5 and width 1 and for $n = 39$ circles in a rectangle of length 10 and width 1. Circles shown with the centre as a small circle are in P, circles shown with the centre as an asterisk are in C. Radius for $n = 37$ circles in the rectangle of length 5 and width 1 from [21] is 0.037018969085497, the radius shown here is 0.0370215204767758. Radius for $n = 39$ circles in a rectangle of length 10 and width 1 from [21] is 0.025139633102023, the radius shown here is 0.0251633377935921.

associated p-value was always greater than 0.05, indicating that the null hypothesis should be accepted at the 5% significance level. Based on this we would conclude that there is no evidence that our current strategy is out-performed by any of the 27 alternative strategies we examined.

Of course we are aware that we could have investigated further here. For example we could have investigated results for containers other than the unit circle and/or examined further values for m in fix and swap FS(m), fix and random FR(m) and swap outer SO(m). We would simply comment here that readers familiar with algorithmic work such as we present in this paper will be aware that there are always different choices to be made, and it is impossible to examine all alternatives in detail. However we believe that the quality of the results seen in Tables 1–3, plus the quality of our FSS heuristic when considered against individual papers in the existing literature (as will be seen later below), mean that the fix and swap FS(M) strategy adopted in this paper is justified.

5.4. Comparison with other formulation space search approaches

In this section we compare (in detail) the results from our FSS heuristic against the work of Mladenovic et al. [18,19], who also used a formulation space search approach.

5.4.1. Comparison with Mladenovic et al. [18]

Mladenovic et al. [18], using an approach which they call reformulation descent (RD), considered both the unit circle and unit square cases. RD works with the Minos nonlinear solver. They also give results for two different formulations of the circle packing problem: one a pure Cartesian formulation (denoted by M_C), the other a pure polar formulation (denoted by M_P), both using Minos. They also give results for a pure Cartesian formulation using the Spenbar nonlinear solver (denoted by SP).

Table 4 shows the results they obtained and is for the case where the container is the unit circle. That table also shows the performance of our FSS heuristic (for the same values of n as considered in [18]).

In Table 4 we give a "Best known" result as taken directly from [18]. The values seen are given as the inverse of the circle radius. The reader should be aware here that there are a number of complications in terms of comparing circle packing results presented in the literature:

- Some papers give results in terms of the circle radius, others in terms of the inverse of that radius.
- The "Best known" value is not a static value, rather it is dynamic as it may change over time as [21] is updated with improved results.
- Different papers use different degrees of precision (number of decimal places given in the published results).
- Some papers present results that are on closer inspection invalid (due to a lack of sufficient numeric precision, typically detected by comparing the detailed results given in the paper with [21]).

The column labeled Best known† in Table 4 refers to the inverse of best known circle radius at that time and is given here precisely as in [18] with six decimal place accuracy. In computing the percentage deviation from this value the results shown for the four approaches (RD, M_C,M_P, SP) are calculated as: $100\left(R_{best}^{-1} - R_{bk}^{-1}\right)/\left(R_{bk}^{-1}\right)$ where R_{bk}^{-1} is the inverse of the best known radius [so here the value used for this inverse is as shown in the table under Best known†] and R_{best}^{-1} is the inverse of best radius found by an approach. The percentage deviations for our FSS heuristic as seen in this table are also calculated in this way.

The computation times (seconds) shown in Table 4 for our FSS heuristic are total computation times, that is the total time for 25 replications from different initial solutions (2.26 GHz Intel Core 2 pc). The computation time given for the approaches in [18] is the average time per replication (over 50 replications) from different initial solutions (in seconds, 1800 MHz Pentium 4 pc).

Considering Table 4 it is clear that our FSS heuristic produces results of better quality (lower percentage deviations) than any of the four approaches of [18]. Note here that the negative percentage deviation values seen for FSS indicate that it produced a result better than the Best known† value seen in Table 4. This reflects one of the complications referred to above, in that best known results are not static values, but rather change over time.

Table 4
Comparison with Mladenovic et al. [18] for the unit circle.

n	Best known†	% Deviation from best known					Total time	Average computation time as reported in [18]			
		FSS	RD	M_C	M_P	SP	FSS	RD	M_C	M_P	SP
10	3.813026	0.00	0.00	0.00	0.00	0.00	41	0.00	0.02	0.01	0.29
15	4.521357	0.00	0.00	0.13	0.13	0.00	37	0.01	0.03	0.02	1.87
20	5.122307	0.00	0.00	0.00	0.00	0.00	47	0.04	0.11	0.08	5.21
25	5.752824	0.00	0.00	0.00	0.00	0.00	65	0.08	0.37	0.19	17.14
30	6.197741	0.00	0.00	0.00	0.00	0.00	82	0.16	0.52	0.29	41.69
35	6.697171	0.00	0.00	0.01	0.02	0.03	112	0.90	1.84	1.73	81.98
40	7.123847	0.00	0.00	0.00	0.00	0.00	109	1.11	2.92	1.91	179.69
45	7.572912	0.00	0.10	0.11	0.04	0.07	151	1.47	3.08	2.19	300.41
50	7.947515	0.00	0.06	0.03	0.00	0.02	173	3.19	5.16	4.41	503.78
55	8.211102	0.00	0.00	1.13	1.57	1.56	219	3.37	6.73	5.15	902.59
60	8.646220	0.00	0.03	0.10	0.57	0.00	246	4.71	7.54	6.00	1526.40
65	9.017397	0.00	0.00	0.47	0.44	0.31	289	16.24	12.94	10.43	2118.60
70	9.346660	−0.01	0.10	0.55	0.32	0.27	347	19.56	17.61	14.54	3484.63
Av($n \leqslant 70$)		0.00	0.02	0.19	0.24	0.17	148	3.91	4.53	3.61	704.94
75	9.678344	−0.07	0.10	0.22	0.44		408	26.46	22.67	17.16	
80	9.970588	−0.02	0.10	0.41	0.29		479	39.15	30.99	23.62	
85	10.163112	0.00	0.72	1.43	1.10		621	38.79	29.85	24.04	
90	10.546069	0.00	0.02	0.02	0.45		586	96.82	47.19	47.70	
95	10.840205	0.00	0.18	0.26	0.48		693	147.35	59.51	41.84	
100	11.082528	0.00	0.30	0.52	0.38		810	180.32	64.96	45.02	
Av($n \leqslant 100$)		−0.01	0.09	0.28	0.33	0.17	290	30.51	16.53	12.96	704.94

Note here that (to two decimal places) for *all* values of n examined our FSS heuristic produces a better (or equal) quality result than any of the four approaches of Mladenovic et al. [18].

With respect to the computation times seen in Table 4 the times given for the four approaches of [18] relate to different hardware than we have used. From the times given though we can reasonably conclude that the time required for our FSS heuristic is not excessive (the largest time seen for FSS in Table 4 is less than 14 minutes). Moreover our FSS heuristic seems to scale better with increasing n, for example in moving from $n = 75$ to $n = 100$ FSS increases by a factor of (810/408) = 1.99. It is clear that SP does not scale and the equivalent values for RD, M_C and M_P are 6.81, 2.87 and 2.62, respectively (although note here that of these three approaches only RD, with a scaling factor of 6.81, approaches our FSS heuristic in terms of solution quality). Mladenovic et al. [18] did not present results for $n > 100$, by contrast we have presented in Table 1 results for values of n up to 500.

Table 5 presents results for the case where the container is the unit square. This table, which has the same format as Table 4, clearly indicates that for this case our FSS heuristic produces results of much better quality than any of the three approaches (RD, M_C, M_P) of [18]. As for the unit circle case we have that (to two decimal places) for *all* values of n examined our FSS heuristic produces a better (or equal) quality result than any of the three approaches of Mladenovic et al. [18]. Note here that, as for the unit circle case, computation time for our FSS heuristic is not excessive (a maximum time of less than 24 minutes).

5.4.2. Comparison with Mladenovic et al. [19]

Table 6 presents, for the unit circle, a comparison between results of our FSS heuristic and those reported in Mladenovic et al. [19], where the authors compare their percentage deviation from Best known† results with two methods: RD (as in [18]), and a formulation space search heuristic (denoted as FSS-M in Table 6). The approach used in [19] was described in the literature survey section above and that description will not be repeated here.

The results for FSS in Table 6 are as in Table 4, but are repeated here for ease of comparison. The results for RD in Table 6 are the same in percentage deviation terms as in Table 4, but the computation time (seconds) given relates to a 900 MHz Pentium 3 pc. The results for RD and FSS-M in Table 6 are for 40 replications, and the computation time given for RD and FSS-M are taken from [19] being the average time per replication.

Considering Table 6 it is clear that in terms of quality of results FSS-M outperforms RD, although computationally it requires (188.87/52.36) = 3.61 times more effort. Our FSS heuristic outperforms FSS-M, for all 11 values of n examined our FSS heuristic produces a better (or equal) quality result than FSS-M.

Table 5
Comparison with Mladenovic et al. [18] for the unit square.

n	Best known†	% Deviation from best known				Total time	Average computation time as reported in [18]		
		FSS	RD	M_C	M_P	FSS	RD	M_C	M_P
10	6.74757140	0.00	0.00	0.00	0.00	61	0.01	0.01	0.02
15	7.86370315	0.00	0.54	0.54	0.00	50	0.03	0.04	0.05
20	8.97808315	0.00	0.00	1.56	0.00	65	0.05	0.11	0.10
25	10.00000000	0.00	0.00	0.00	0.00	78	0.10	0.24	0.28
30	10.90856809	0.00	0.63	0.63	0.58	99	0.26	0.61	0.57
35	11.86370360	0.00	0.32	0.32	0.59	155	0.38	1.04	1.11
40	12.62837533	0.00	0.09	0.09	0.19	169	1.10	1.94	1.97
45	13.38198309	0.00	0.16	0.16	0.11	237	1.24	2.26	2.54
50	14.01009567	0.00	0.28	1.04	0.28	276	1.87	4.00	3.64
55	14.69391977	0.00	0.61	0.61	0.37	286	3.27	6.15	5.22
60	15.37742112	0.00	0.38	0.38	0.53	350	5.17	8.11	7.13
65	15.82179344	0.00	0.93	0.93	1.09	487	7.50	12.26	10.04
70	16.50255154	0.00	0.36	0.92	0.80	517	13.43	13.12	11.92
75	17.09561268	0.01	0.67	0.73	0.55	621	17.01	18.04	15.37
80	17.43050631	0.00	1.45	1.50	0.77	750	24.95	23.65	23.37
85	17.96028299	0.00	1.39	1.23	1.05	978	33.11	30.44	26.05
90	18.60466847	0.00	0.77	1.25	1.13	1179	43.62	35.85	27.81
95	19.07658639	0.00	0.80	0.94	0.49	1423	51.02	43.49	35.48
100	20.00000000	−2.72	0.00	0.00	0.00	1317	80.80	61.15	47.68
Average		−0.14	0.49	0.68	0.45	479	15.00	13.82	11.60

Table 6
Results for the unit circle comparing our FSS heuristic with the RD and FSS-M approaches in [19].

n	Best known†	FSS		RD		FSS-M	
		% Dev	Time	% Dev	Time	% Dev	Time
50	7.947515	0.00	173	0.06	3.19	0.00	80.54
55	8.211102	0.00	219	0.00	3.37	0.00	72.81
60	8.646220	0.00	246	0.03	4.71	0.00	84.39
65	9.017397	0.00	289	0.00	16.24	0.00	108.25
70	9.346660	−0.01	347	0.10	19.56	0.01	151.64
75	9.678344	−0.07	408	0.10	26.46	0.02	164.51
80	9.970588	−0.02	479	0.10	39.15	0.04	229.49
85	10.163112	0.00	621	0.72	38.79	0.18	256.17
90	10.546069	0.00	586	0.02	96.82	0.02	294.77
95	10.840205	0.00	693	0.18	147.35	0.07	308.34
100	11.082528	0.00	810	0.30	180.32	0.12	326.67
Average		−0.01	443	0.15	52.36	0.04	188.87

In FSS-M Mladenovic et al. [19] ignore overlap constraints for circles whose centres (at the initial solution, before optimization) are sufficiently far apart. Since FSS outperforms FSS-M it seems clear that the strategy adopted in our FSS algorithm of using constraints that limit the range of movement for circle centres is a superior one to use.

5.4.3. Single formulations

It is of interest to investigate whether (or not) switching between formulations (Cartesian and Polar) is in fact making a difference to the results obtained, or whether our algorithm would do better just to consider a single formulation. Technically the Cartesian only formulation corresponds to $C = \{1,\ldots,n\}$ and $P = \emptyset$ (with no swapping of the sets C and P in Algorithm 1). The Polar only formulation corresponds to $P = \{1,\ldots,n\}$ and $C = \emptyset$ (with no swapping of the sets C and P in Algorithm 1).

To investigate this, as well as to investigate whether using a different nonlinear solver such as Minos (which was used in [18,19]) would make a difference, we present Table 7. In that table we show the average percentage deviation from the best known results when our heuristic solves the unit circle case. Here the averages are taken over values of $n = 10, 15, \ldots, 80$ (larger values of n gave difficulties for Minos).

Considering Table 7 it is clear that Minos is not effective when compared against Snopt. With regard to using just a single formulation (Cartesian or Polar) it is clear that a much lower average percentage deviation is obtained when we use both formulations (as in our FSS heuristic as presented in this paper).

5.5. Comparison with other work

Above we have compared (in detail) the results from our FSS heuristic against the work of Mladenovic et al. [18,19], who also used a formulation space search approach.

Table 7
Results for the unit circle comparing our FSS heuristic with single formulations using Minos and Snopt.

Formulation	Average percentage deviation	
	Minos	Snopt
FSS – Cartesian only	0.123182312	0.001664567
FSS – Polar only	77.480177257	0.001616629
FSS – Cartesian and Polar, as Algorithm 1	53.757778475	0.000002461

Results have also been presented in the literature by other authors against which we can compare our FSS heuristic. For reasons of space we, rather than give a series of detailed tables of comparison, present in Table 8 a summary. In that table we show:

- The paper being considered.
- The container used and the values of n for which we are making a comparison.
- The average percentage deviation, utilizing the best known value given in the paper being considered, this average percentage deviation being calculated using either radius or its inverse, depending upon the approach adopted in the paper.
- Average computation time (seconds).
- Average percentage deviation for our FSS heuristic.
- Average computation time (seconds) for our FSS heuristic (Intel Core 2 pc (2.26 GHz, 4 GB RAM)).
- The nonlinear programming solver used.

With respect to Table 8 it is clear that the results given by Pintér [20] are not competitive with the other results shown.

For the container being a square it appears that our FSS heuristic dominates, both with respect to quality of result and computation time, the approach of Birgin and Gentil [1].

For the container being a circle it appears that our FSS heuristic dominates, both with respect to quality of result and computation time, the approach of Grosso et al. [5].

For the container being a circle and the approaches given in [1,16] our FSS heuristic has a higher average percentage deviation, but much lower computation times. It would seem reasonable to conclude therefore that our FSS heuristic is a computationally effective approach (when balancing quality of result obtained against computation time required) when compared with Birgin and Gentil [1] and Liu et al. [16].

We would also comment here that none of the papers seen in Table 8 have presented results for $n > 100$. By contrast in this paper we have presented results for our FSS heuristic for values of n up to 500.

Finally we would comment here that with respect to Table 8 it is clear that authors have used a specific nonlinear solver in conjunction with the algorithms they have presented. Because of the nature of the circle packing problem many papers in the literature solve a nonlinear program as a component part of the overall algorithm presented (as indeed we also do in the FSS heuristic presented in this paper).

Table 8
Comparison with other work.

Paper	Container values of n	Average % deviation	Average time (seconds)	FSS average % deviation	FSS average time (seconds)	Solver used
Birgin and Gentil [1]	Circle 10, 15, 20,…,50	0	1019	1.66×10^{-7}	91	ALGENCAN
Birgin and Gentil [1]	Square 10, 15, 20,…,50	0.00189281	1878	5.12×10^{-6}	132	ALGENCAN
Grosso et al. [5]	Circle 30, 35, 40,…,100	0.00059028	3784	−0.00049533	355	SNOPT
Liu et al. [16]	Circle 35, 40, 45,…,100	3.10664×10^{-10}	79,870	2.00×10^{-5}	375	SNOPT
Pintér [20]	Circle 10, 15, 20,…,60	0.59949089	195	1.36×10^{-7}	117	LGO

For Birgin and Gentil [1] the computation time is for a 2.4 GHz Intel Core 2 Quad, 4 GB RAM pc. They start from the solution given by the heuristic algorithm of Birgin and Sorbal [2]. The time given above does not include the time for this heuristic, which [2] indicates was one hour (2 GHz AMD Opteron 244 processor, 2 GB RAM) for all values of n.

For Grosso et al. [5] the computation time is for a Pentium IV 2.4 GHz, 1 GB RAM pc.

Liu et al. [16] use five replications, but only give the time for the replication that resulted in the best solution found. To account for this we have multiplied the time for this replication by five. Their computation time is for a Pentium IV 1.6 GHz, 512 MB RAM pc.

Pintér [20] gives results for two approaches, the results given above are for the LGO + CONCOPT approach which (effectively) dominates the other approach presented. His computation time is for an AMD Athlon 64 3200 + 2 GHz pc.

As our results in Table 7 above indicated exactly the same algorithm can give very different results if a different nonlinear solver is used. In this respect therefore comparison tables such as Table 8, although necessary when comparing different algorithms from the literature, need to be considered in the light of the fact that we typically do not know how an algorithm would have performed had a different nonlinear solver been used.

5.6. Contribution

In this section we outline what we believe to be the contribution of this paper to the existing literature.

Firstly we would note that we have in this paper put forward a heuristic algorithmic framework that can be applied (with only minor adaptations) to a variety of differing containers. In this paper we have considered six different containers: the unit circle, the unit square, a rectangle of length five and width one, a rectangle of length 10 and width one, a right-angled isosceles triangle where both equal sides are of length one and a semicircle of radius one. By contrast the majority of papers in the literature consider just a single container.

Secondly we would note that, as evidenced in Tables 1–3, our heuristic is capable of producing good quality results (with a low average percentage deviation from previously best known results from Specht [21]) in a reasonable computation time. Moreover for a number of the containers we improve on previously best known results. Since the Packomania web site of Specht [21] is updated with results from all authors, and hence represents the collective achievement of all previous work, this is a significant achievement.

Thirdly, when we compare our heuristic individually against papers previously published in the literature we can draw the following conclusions:

- *For all values of n examined*, our FSS heuristic produces a better (or equal) quality result (lower percentage deviation) than:
  - Any of the four approaches of Mladenovic et al. [18] for the unit circle (Table 4).
  - Any of the three approaches of Mladenovic et al. [18] for the unit square (Table 5).
  - The approach of Mladenovic et al. [19] for the unit circle (Table 6).
- From Table 8 our FSS heuristic dominates, both with respect to quality of result and computation time, the approaches of:
  - Birgin and Gentil [1] for the unit square.
  - Grosso et al. [5] for the unit circle.
  - Pintér [20] for the unit circle.
- From Table 8 our FSS heuristic has a higher average percentage deviation, but much lower computation times, than:
  - Birgin and Gentil [1] for the unit circle.
  - Liu et al. [16] for the unit circle.

6. Conclusions

In this paper we presented a heuristic based on formulation space search for the circle packing problem with respect to a variety of different containers: the unit circle, unit square, rectangle, isosceles right-angled triangle and semicircle.

Computational results were presented for all of these containers involving up to 500 circles. Problems of this size are much larger than those considered by most other authors. These indicated that our heuristic improves on previous results based on formulation space search presented in the literature. An important difference between our paper and earlier formulation space search work for circle packing is the use of constraints that limit the range of movement for circle centres.

Computational results indicated that our heuristic dominates with respect to the quality of result a number of other heuristics presented in the literature. With respect to other work where our heuristic is not dominant results indicate that our heuristic is a computationally effective approach (when balancing quality of result obtained against computation time required).

For a number of the containers we considered our heuristic improves on the best result previously known.

One of the advantages of our heuristic is that it is easy to adapt for a variety of different containers. Computational results, for six different containers, indicate that it gives good quality solutions irrespective of the container being considered.

Acknowledgments

The first author is studying at Brunel with grant support from CONACYT, the Mexican National Council for Science and Technology.

References

[1] E.G. Birgin, J.M. Gentil, New and improved results for packing identical unitary radius circles within triangles, rectangles and strips, Computer & Operations Research 37 (7) (2010) 1318–1327.
[2] E.G. Birgin, F.N.C. Sobral, Minimizing the object dimensions in circle and sphere packing problems, Computer & Operations Research 35 (7) (2008) 2357–2375.
[3] I. Castillo, F.J. Kampas, J.D. Pinter, Solving circle packing problems by global optimization: Numerical results and industrial applications, European Journal of Operational Research 191 (3) (2008) 786–802.
[4] Gill, P.E., Murray, W., Saunders, M.A., 2008. User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming. Available at <http://www.ccom.ucsd.edu/~peg/papers/sndoc7.pdf>.
[5] A. Grosso, A.R.M.J.U. Jamali, M. Locatelli, F. Schoen, Solving the problem of packing equal and unequal circles in a circular container, Journal of Global Optimization 47 (1) (2010) 63–81.
[6] A. Grosso, M. Locatelli, F. Schoen, A population-based approach for hard global optimization problems based on dissimilarity measures, Mathematical Programming Series A 110 (2) (2007) 373–404.
[7] Hansen, P., Mladenović, N., Brimberg, J., Perez, J.A.M., 2010. Variable neighborhood search. In: Gendreau, M., Potvin, J.-Y. (Eds.), Handbook of Metaheuristics, Springer. International Series in Operations Research & Management Science, vol. 146, pp. 61–86.
[8] U.H.E. Hansmann, L.T. Wille, Global optimization by energy landscape paving, Physical Review Letters 88 (6) (2002) (Article 068105).
[9] A. Hertz, M. Plumettaz, N. Zufferey, Variable space search for graph coloring, Discrete Applied Mathematics 156 (13) (2008) 2551–2560.
[10] A. Hertz, M. Plumettaz, N. Zufferey, Corrigendum to variable space search for graph coloring [Discrete Appl. Math. 156 (2008) 2551–2560], Discrete Applied Mathematics 157 (7) (2009) 1335–1336.
[11] M. Hifi, R. M'Hallah, A literature review on circle and sphere packing problems: Models and methodologies, Advances in Operations Research (2009). Article ID 150624 <http://downloads.hindawi.com/journals/aor/2009/150624.pdf>.
[12] Holmström, K., Göran, A.O., Edvall, M.M., 2008. User's Guide for TOMLAB/SNOPT. Available at <http://tomopt.com/docs/TOMLAB_SNOPT.pdf>.
[13] Y. Kochetov, P. Kononova, M. Paschenko, Formulation space search approach for the teacher/class timetabling problem, Yugoslav Journal of Operations Research 18 (1) (2008) 1–11.
[14] R.H. Leary, Global optimization on funneling landscapes, Journal of Global Optimization 18 (4) (2000) 367–383.
[15] Liu, J., Wang, Y., Pan, J., 2009. Efficiently packing circles into a larger containing circle. In: Zhang, W., Chen, Z., Douglas, C.C., Tong, W. (Eds.), High Performance Computing and Applications", Revised Selected Papers from the Second International Conference, HPCA 2009, Shanghai, China, August 10–12. Lecture Notes in Computer Science, vol. 5938, pp. 250–256.
[16] J. Liu, S. Xue, Z. Liu, D. Xu, An improved energy landscape paving algorithm for the problem of packing circles into a larger containing circle, Computers & Industrial Engineering 57 (3) (2009) 1144–1149.
[17] Liu, J., Yao, Y., Zheng, Y., Geng, H., Zhou, G., 2009. An effective hybrid algorithm for the circles and spheres packing problems. In: Proceedings of the Third International Conference on Combinatorial Optimization and its Applications, COCOA 2009, Huangshan, China, June 10–12. Lecture Notes in Computer Science, vol. 5573, pp. 135–144.
[18] N. Mladenović, F. Plastria, D. Urošević, Reformulation descent applied to circle packing problems, Computer & Operations Research 32 (9) (2005) 2419–2434.
[19] Mladenović, N., Plastria, F., Urošević, D., 2007. Formulation space search for circle packing problems. In: Proceedings of the International Workshop on

Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics, SLS 2007, Brussels, Belgium, September 6–8, 2007. Lecture Notes in Computer Science, vol. 4638, pp. 212–216.
[20] J.D. Pintér, Nonlinear optimization with GAMS/LGO, Journal of Global Optimization 38 (1) (2007) 79–101.
[21] Specht, E., 2010. Packomania. Available at <http://www.packomania.com>.
[22] P.G. Szabó, M.C. Markót, T. Csendes, E. Specht, L.G. Casado, I. Garcia, New approaches to circle packing in a square: With program codes, Springer Optimization and its Applications 6 (2007).
[23] D.J. Wales, J.P.K. Doye, Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms, The Journal of Physical Chemistry A 101 (28) (1997) 5111–5116.