Computers and Operations Research 94 (2018) 106–117

Contents lists available at ScienceDirect

Computers and Operations Research

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

Packing unequal rectangles and squares in a fixed size circular container using formulation space search

C.O. López [a], J.E. Beasley [b,*]

[a] Faculty of Sciences, National Autonomous University of Mexico, Mexico City, Mexico
[b] Mathematical Sciences, Brunel University, Uxbridge UB8 3PH, UK and JB Consultants, Morden, UK

article info

Article history:
Received 11 October 2017 Revised 19 February 2018 Accepted 20 February 2018 Available online 21 February 2018

Keywords:
Formulation space search
Mixed-integer nonlinear program
Rectangle packing
Square packing

abstract

In this paper we formulate the problem of packing unequal rectangles/squares into a fixed size circu-lar container as a mixed-integer nonlinear program. Here we pack rectangles so as to maximise some objective (e.g. maximise the number of rectangles packed or maximise the total area of the rectangles packed). We show how we can eliminate a nonlinear maximisation term that arises in one of the con-straints in our formulation. We indicate the amendments that can be made to the formulation for the special case where we are maximising the number of squares packed. A formulation space search heuris-tic is presented and computational results given for publicly available test problems involving up to 30 rectangles/squares. Our heuristic deals with the case where the rectangles are of fixed orientation (so cannot be rotated) and with the case where the rectangles can be rotated through ninety degrees.

1. Introduction

In this paper we consider the problem of packing non-identical rectangles (i.e. rectangles of different sizes) into a fixed size circu-lar container. Since the circular container may not be large enough to accommodate all of the rectangles available to be packed there exists an element of choice in the problem. In other words we have to decide which of the rectangles will be packed, and more-over for those that are packed their positions within the container. The packing should respect the obvious constraints, namely that the packed rectangles do not overlap with each other and that each packed rectangle is entirely within the container. This pack-ing should be such so as to maximise an appropriate objective (e.g. maximise the number of rectangles packed or maximise the total area of the rectangles packed).

To illustrate the problem suppose we have ten rectangles with sizes as shown in Table 1 to be packed into a fixed sized circular container. The rectangles shown in Table 1 have been ordered into ascending area order.

Regarding the rectangles as being of fixed orientation, i.e. they cannot be rotated, then:

- If we are wish to maximise the number of rectangles packed Fig. 1 shows the solution as derived by the approach presented in this paper. In that figure we can see that seven of the ten rectangles have been packed, three rectangles are left unpacked.
- If we are wish to maximise the total area of the rectangles packed Fig. 2 shows the solution as derived by the approach presented in this paper. In that figure we can see that five of the ten rectangles have been packed.

If the rectangles can be rotated through ninety degrees then:

- If we are wish to maximise the number of rectangles packed Fig. 3 shows the solution as derived by the approach presented in this paper. In that figure we can see that seven of the ten rectangles have been packed, three rectangles are left unpacked.
- If we are wish to maximise the total area of the rectangles packed Fig. 4 shows the solution as derived by the approach presented in this paper. In that figure we can see that seven of the ten rectangles have been packed.

In Figs. 3 and 4 the letter r after the rectangle number indicates that the rectangle has been rotated through ninety degrees. Com-paring Figs. 1 and 3 we can see that they both involve the pack-ing of seven rectangles. Whilst allowing rotation through ninety degrees allows the possibility of a better solution as compared with the no rotation case this is by no means assured. Comparing Figs. 2 and 4 we can see that in this particular case an improve-ment in the total area of the rectangles packed has been made by making use of rotation.

The structure of this paper is as follows. In Section 2 we review the literature relating to the packing of rectangles. We discuss ap-

* Corresponding author.
E-mail addresses: claudia.lopez@ciencias.unam.mx (C.O. López), john.beasley@brunel.ac.uk, john.beasley@jbconsultants.biz (J.E. Beasley).

https://doi.org/10.1016/j.cor.2018.02.012
0305-0548/

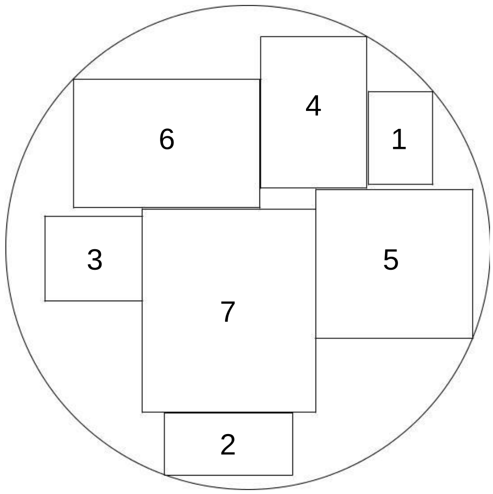

Fig. 1. Maximise the number of rectangles packed, no rotation, solution value 7.

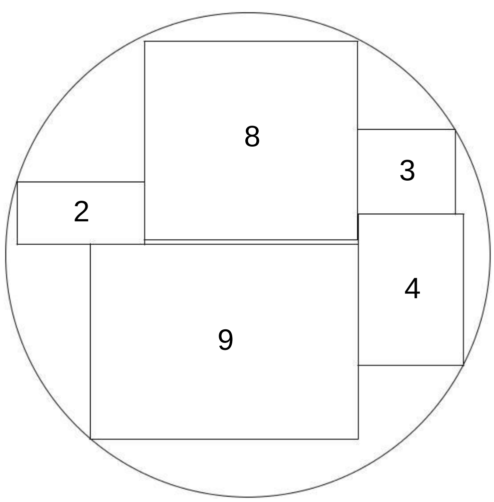

Fig. 2. Maximise the total area of the rectangles packed, no rotation, solution value 37.6878.

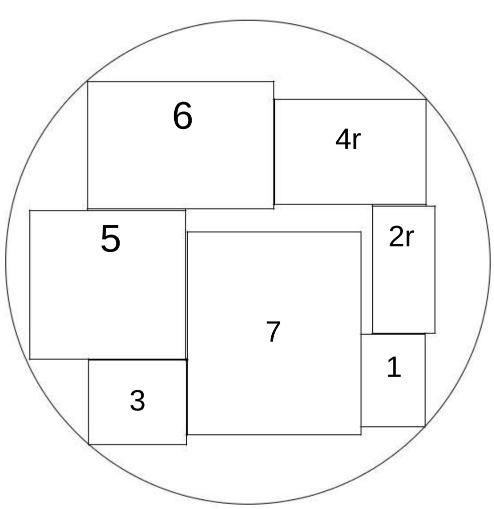

Fig. 3. Maximise the number of rectangles packed, rotation allowed, solution value 7.

Table 1
Rectangle packing example, circular container radius 4.18.

Rectangle	Length	Width
1	1.10	1.61
2	2.20	1.08
3	1.68	1.46
4	1.82	2.61
5	2.70	2.57
6	3.21	2.21
7	2.99	3.51
8	3.68	3.42
9	4.62	3.36
10	3.79	4.79

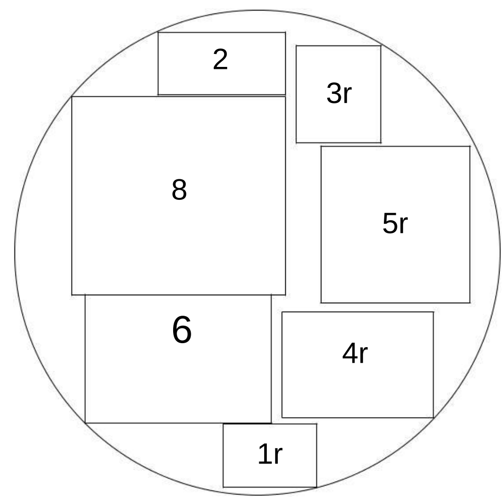

Fig. 4. Maximise the total area of the rectangles packed, rotation allowed, solution value 37.9687.

plication areas where rectangle packing problems arise. We also re-view the literature relating to the particular metaheuristic, formu-lation space search, used in this paper. In Section 3 we formulate the problem of packing unequal rectangles/squares into a fixed size circular container as a mixed-integer nonlinear program. We show how we can eliminate a nonlinear maximisation term that arises in one of the constraints in our formulation. We also show how we can deal with the case where rectangles can be rotated through ninety degrees. We indicate the amendments that can be made to the formulation for the special case where we are maximising the number of squares packed. Section 4 gives details of the formula-tion space search heuristic that we use to solve the problem. Com-putational results are presented in Section 5 for problems involving up to 30 rectangles/squares. In that section we give results both for maximising the number of rectangles/squares packed and for max-imising the total area of the rectangles/squares packed. Finally in Section 6 we present our conclusions.

2. Literature survey

In this section we first discuss the literature relating to the problem of packing rectangles and its applications. We then dis-cuss the literature relating to the particular metaheuristic, formu-lation space search, we use to solve the rectangle packing problem considered in this paper.

2.1. Rectangle packing

The majority of the work in the literature related to rectan-gle packing deals with packing rectangles/squares within a larger container that is either a square, or a rectangle, or a rectangular strip with one dimension fixed and the other dimension variable (e.g. fixed width, but variable length).

A common feature of such work is that it is assumed that all of the smaller rectangles have to be packed into the larger container, which leads to an optimisation problem relating to minimising the dimension of the container. For example for a square container a natural optimisation problem is to minimise the side of the square container (which also minimises its perimeter and area). For a rect-angular container one can examine minimising either its perimeter or its area. For a rectangular strip one can minimise the variable dimension. With respect to the packing of rectangles within a cir-cular container then the natural optimisation problem is to min-imise the radius of the container.

In our literature survey below we focus principally on papers that take a packing approach. The reader may be aware that a closely related problem to packing is cutting e.g. cutting rectangles from a larger stock rectangle. There has been a substantial amount of work presented in the literature dealing with cutting. However much of that work involves additional restrictions with regard to the cuts that are made. One such restriction might be that the cuts are guillotine cuts, a guillotine cut on a rectangle being a cut from one edge of the rectangle to the opposite edge which is parallel to the two remaining edges. Another such restriction might be to limit the cutting to a number of stages, where at each stage guil-lotine cuts are made, but in a direction opposite to that adopted in the previous stage. So for example in the first stage guillotine cuts are made parallel to the y-axis, then in the second stage guillotine cuts are made parallel to the x-axis, etc. Since the primary focus of the work presented in this paper is packing rectangles within a *circular container* we, for space reasons, exclude detailed consider-ation of work focused on cutting rectangles from rectangular con-tainers from the literature survey presented below.

Unless otherwise stated all of the work considered below deals with orthogonal packing, so rectangles/squares are packed without rotation.

Li and Cheng (1989) show that the problem of determining whether a set of squares can be packed into a larger rectangle is strongly NP-complete. In addition they show that the problem of determining whether a set of rectangles can be packed into a square is NP-complete. Leung et al. (1990) show that the prob-lem of determining whether a set of squares can be packed into a square is strongly NP-complete.

Picouleau (1996) considered the worst-case analysis of three fast heuristics for packing squares into a square container so as to minimise the size of the square. Murata et al. (1996) present a simulated annealing algorithm for the problem for packing rectan-gles into a rectangular container so as to minimise the size (area) of the container. Liu and Teng (1999) present a genetic algorithm for the problem of packing a set of rectangles into a strip of fixed width using minimum height.

Wu et al. (2007) present a heuristic attempting to pack ev-ery member of a set of rectangles inside a fixed size rectangular container. Caprara et al. (2006) discuss absolute worst-case perfor-mance ratios for lower bounds on packing rectangles/squares into a square container so as to minimise the size of the square container. They consider the case where the rectangles have fixed orientation and the case where they can be rotated through ninety degrees. Huang et al. (2007) present a heuristic approach to packing rect-angles within a fixed size rectangular container so as to maximise the total area of the rectangles packed where the rectangles can be rotated through ninety degrees.

Birgin et al. (2010) consider packing the maximal number of identically sized rectangles inside a rectangular container. Their ap-proach is based upon recursive partitioning and allows the rectan-gles to be rotated through ninety degrees. Korf et al. (2010) con-sider the problem of packing a set of rectangles (with and with-out ninety degree rotation allowed) in a rectangular container of minimal area. They adopt a constraint satisfaction approach to the problem. Maag et al. (2010) consider the problem of packing a set of rectangles in a rectangular container of minimal area. Their ap-proach is based on relaxing the constraint on rectangle overlap.

Huang and Korf (2012) consider the same problem as Korf et al. (2010) but adopt an approach based on first deciding x-coordinate values for each rectangle. Bortfeldt (2013) presents a number of heuristic approaches (based on solution methods for two-dimension knapsack and two-dimension strip packing) for packing rectangles into a rectangular container so as to minimise the size (area) of the container.

Martello and Monaci (2015) consider the problem of pack-ing rectangles/squares into a square container so as to minimise the size of the container. They present a linear integer pro-gramming formulation and an exact approach based on a two-dimensional packing algorithm as well as a metaheuristic. They deal with the case where the rectangles have fixed orientation and also the case where they can be rotated through ninety degrees. Delorme et al. (2017) present a Benders' decomposition approach to the problem of packing a set of rectangles (with ninety degree rotation allowed) into a strip of fixed width using minimum height. Their approach (as they discuss) can be easily applied to the prob-lem of packing rectangles/squares into a square container of mini-mal size.

It is important to note here that a number of the approaches given in the literature for the problem of packing rectangles within a rectangular container utilise the fact that rectangle position co-ordinates can be taken from a finite discrete set (e.g. by packing rectangles so that they are positioned at their lowest bottom-left position). For example see Delorme et al. (2017) and Martello and Monaci (2015). However in this paper we consider a circular con-tainer, and *the lack of rectangular sides to the container render such discretisation approaches invalid for the problem we consider*.

As far as we are aware the problem considered in this paper of packing unequal rectangles/squares into a circular container has only been considered by just a few papers in the literature pre-viously. Li et al. (2014) consider the problem of packing orthogo-nal unequal rectangles in a circular container with an additional constraint related to mass balance. Their objective function is to minimise the radius of the container. A heuristic algorithm is pre-sented.

Hinostroza et al. (2013) consider the problem of cutting rect-angular boards from a log, regarded as a circular container. They present a nonlinear formulation of the problem (based on Birgin et al., 2006b), and two heuristics, one based on order-ing the rectangles and the other on simulated annealing. Note here that, in our judgement, their formulation is flawed.

Work has been presented in the literature relating to packing rectangles/squares into arbitrary convex regions, and such work can be applied to a circular container. We discuss this work below.

Birgin et al. (2006a) introduce the concept of sentinels sets, which are finite subsets of the items to be packed such that, when two items are superposed, at least one sentinel of one item is in the interior of the other item. Using these sentinel sets they consider packing identical rectangles within both con-vex regions and a rectangular container, with and without rectan-gle rotation (both ninety degree rotation and arbitrary rotation). Birgin et al. (2006b) consider packing rectangles (with and with-out ninety degree rotation). Their objective is to feasibly pack all rectangles. Iteratively increasing the number of rectangles enables one to maximise the number of (identical) rectangles placed. Their approach is based on nonlinear optimisation.

Birgin and Lobato (2010) consider packing identical rectangles within an arbitrary convex region where a common rotation of

θ degrees (not restricted to θ = 90) of all the rectangles is al-lowed. In addition a rectangle can be rotated through ninety de-grees before a rotation of θ is applied. Their solution method is a combination of branch and bound and active-set strategies for bound-constrained minimization of smooth functions. Cassioli and Locatelli (2011) present a heuristic approach based on iterated local search for the problem of packing the maximum number of rect-angles of the same size within a convex region (where rectangle rotation through ninety degrees is allowed).

Andrade and Birgin (2013) present symmetry breaking con-straints for two problems relating to packing identical rectangles (with or without ninety degree rotation) in a polyhedron. They consider packing as many identical rectangles as possible within a given polyhedron as well as finding the smallest polyhedron of a specified type that accommodates a fixed number of identical rect-angles.

More generally Birgin (2016) considers the application of non-linear programming in packing problems. They note that nonlin-ear programming formulations and methods have been success-fully applied to a wide range of packing problems. In particular we in this paper, as in the formulation presented below, use a nonlin-ear model.

2.2. Applications

The problem of packing rectangular objects into a larger con-tainer (equivalently cutting rectangular objects from a larger con-tainer) appears in a number of practical situations. As noted in Dowsland and Dowsland (1992) the earliest applications were in glass and metal industries where smaller rectangular objects had to be cut from larger (typically rectangular) stock pieces. A further application they discuss occurs in pallet loading where rectangu-lar boxes have to be packed onto a wooden pallet for transport. Sweeney and Paternoster (1992) present an application-orientated research bibliography that lists some of the early work related to packing.

Lodi et al. (2002) present a literature survey relating to two-dimensional packing and solution approaches. They men-tion a number of practical applications relating to rectangle cut-ting/packing. These include the arrangement of articles and adver-tisements on newspaper pages and in the wood and glass indus-try cutting rectangular items from larger sheets of material. They also mention the placement of goods on shelves in warehouses. Wascher et al. (2002) also mention some practical applications (such as pallet loading) in their work presenting a typology of cut-ting and packing problems.

In relation to the specific problem considered in this paper of packing unequal rectangles/squares into a fixed size circular con-tainer we are aware of a number of practical applications.

For example in the forestry/lumber industry consider the cut-ting of rectangular wooden boards from timber logs made from trees that have been felled. Here, by approximating the shape of the timber log by a circle of known radius, we have the prob-lem considered in this paper, namely which of the rectangles (of known sizes) that we desire to cut should be cut from the circular log (Hinostroza et al., 2013).

A further practical example relates to the problem consid-ered in Li et al. (2014) which was concerned with packing or-thogonal unequal rectangles in a circular container with an addi-tional constraint related to mass balance. Here the container was a satellite and the rectangular objects related to items compris-ing the satellite payload. The mass balance constraint considered in Li et al. (2014) was a single nonlinear constraint that involved the (mass weighted) centres of each rectangle. Since, as will be-come apparent below, our formulation space search approach for packing rectangles into a fixed size circular container is based on a mixed-integer nonlinear program it is trivial to introduce into our approach a single additional nonlinear constraint (such as a mass balance constraint).

2.3. Formulation space search

When solving nonlinear non-convex problems with the aid of a solver, Mladenovic´ et al. (2005) observed that different formu-lations of the same problem may have different characteristics. Hence a natural way to proceed is by swapping between formu-lations. Under this framework Mladenovic´ et al. (2005) use formu-lation space search (henceforth FSS) for the circle packing problem considering two formulations of the problem: one in a Cartesian coordinate system, the other in a Polar coordinate system. Their algorithm solves the problem with one formulation at a time and when the solution is the same for all formulations the algorithm terminates. They consider packing identical circles into the unit cir-cle and the unit square.

In Mladenovic´ et al. (2007) they improve on Mladenovic´ et al. (2005) by considering a mixed formulation of the prob-lem. They set a subset of the circles in the Cartesian system whilst the rest of the circles were in the Polar system. López and Beasley (2011) use FSS for the problem of packing equally sized circles inside a variety of containers. They present computational results which show that their approach improves upon previous results based on FSS presented in the literature. For some of the containers considered they improve on the best result previ-ously known. López and Beasley (2013) use FSS to solve the pack-ing problem with non-identical circles in different shaped con-tainers. They present computational results which were compared with benchmark problems and also proposed some new instances. López and Beasley (2016) use FSS to solve the problem of packing non-identical circles in a fixed size container.

Essentially FSS exploits the fact that:

- because of the nature of the solution process in nonlinear op-timisation we often fail to obtain a globally optimum solution from a single formulation; and so
- perturbing/changing the formulation and then resolving the nonlinear program may lead to an improved solution.

Given the above it is a simple matter to construct iterative schemes that move between formulations in a systematic manner.

FSS has been applied to a few problems additional to circle packing (e.g. timetabling (Kochetov et al., 2008)). In López and Beasley (2014) FSS was used to solve some benchmark mixed-integer nonlinear programming problems. In a more general sense an adaptation to FSS was presented in Brimberg et al. (2014) for solving continuous location problems. More discussion as to FSS can be found in Hansen et al. (2010). A related approach is variable space search, which has been applied to graph colouring (Hertz et al., 2008; 2009). Other related approaches are variable formula-tion search which has been applied to the cutwidth minimisation problem (Duarte et al., 2016; Pardo et al., 2013) and variable objec-tive search which has been applied to the maximum independent set problem (Butenko et al., 2013).

As noted in Pardo et al. (2013) variable space search, vari-able formulation search and variable objective search contain sim-ilar ideas as originally expounded using FSS. At a slightly more general level FSS can be regarded as a variant of variable neigh-bourhood search, for example see Amirgaliyeva et al. (2017) and Hansen et al. (2017).

3. Formulation

In this section we first present our basic formulation for the problem of packing unequal rectangles in a fixed size circular container as a mixed-integer nonlinear program. We then show how we can eliminate a nonlinear maximisation term that arises in one of the constraints in our formulation. We indicate how we can deal with the case where rectangles can be rotated through ninety degrees. For the special case where we are maximising the number of squares packed we present the amendments that can be made to the formulation.

3.1. Basic formulation

O problema que consideramos é encontrar a embalagem ponderada máxima de n retângulos desiguais em um recipiente circular de tamanho fixo. Aqui temos a opção, para cada retângulo desigual, de optar por embalar ou não. Podemos formular esse problema da seguinte maneira: deixe que o contêiner circular de tamanho fixo seja do raio R e, sem perda de generalidade, deixe-o centrado na origem do plano Euclidiano. Temos n retângulos para construir uma embalagem, em que o retângulo i tem um lado horizontal de comprimento Li e um lado vertical de largura Wi e valor (se embalado) Vi. Em nossa fórmula básica, não permitimos rotação ao embalar retângulos, de modo que esses retângulos sejam embalados com suas bordas horizontais (comprimento) paralelas ao eixo x, suas bordas verticais (largura) paralelas ao eixo y.Claro estamos lidando com quadrados de embalagem, então Li = Wi. Identifique aqui os retângulos para que sejam ordenados em ordem crescente de tamanho (área) (ou seja, LiWi ≤ Li + 1Wi + 1 i = 1,..., N - 1). Usando um valor Vi aqui para cada retângulo, eu posso considere o número de problemas diferentes dentro da mesma formulação. Por exemplo, se considerarmos Vi = 1 i = 1,. . . , n então temos o problema de maximizar o número de retângulos compactados. Se tomarmos Vi = LiWi i = 1,. . . , n então temos o problema de maximizar a área total dos retângulos compactados. Alternativamente, o Vi i = 1,. . . , n podem ser atribuídos valores arbitrários. Em seguida, as variáveis são:

- α_i = 1 if rectangle i is packed, 0 otherwise; $i = 1, \dots, n$
- (x_i, y_i) the position of the centre of rectangle i; $i = 1, \dots, n$

With regard to the positioning (so (x_i, y_i)) of any unpacked rectangle i (for which $\alpha_i = 0$) our formulation forces all unpacked rectangles to be positioned at the origin. Let Q be the set of all rectangle pairs $[(i, j) \mid i = 1, \dots, n; \quad j = 1, \dots, n; \; j > i]$. The formulation is:

$$\max \sum_{i=1}^{n} \alpha_i V_i \qquad (1)$$

subject to

$$-\alpha_i\left(\sqrt{R^2 - W_i^2/4} - L_i/2\right) \le x_i \le \alpha_i\left(\sqrt{R^2 - W_i^2/4} - L_i/2\right) \quad i = 1, \dots, n \qquad (2)$$

$$-\alpha_i\left(\sqrt{R^2 - L_i^2/4} - W_i/2\right) \le y_i \le \alpha_i\left(\sqrt{R^2 - L_i^2/4} - W_i/2\right) \quad i = 1, \dots, n \qquad (3)$$

$$(x_i + L_i/2)^2 + (y_i + W_i/2)^2 \le \alpha_i R^2 + (1 - \alpha_i)(L_i^2/4 + W_i^2/4) \quad i = 1, \dots, n \qquad (4)$$

$$(x_i + L_i/2)^2 + (y_i - W_i/2)^2 \le \alpha_i R^2 + (1 - \alpha_i)(L_i^2/4 + W_i^2/4) \quad i = 1, \dots, n \qquad (5)$$

A função objetivo, Eq. (1), maximiza o valor dos retângulos empacotados. A Eq. (2) garante que, se um retângulo é empacotado (ie αi = 1), sua coordenada x está em [- ((R2 2 - Wi / 4) - pode ser / 2), + ((R2 - W 2 / 4) - L / 2)]. Esses limites são facilmente deduzidos de considerações geométricas, p. considere o valor da coordenada x central associado a um retângulo colocado com seu centro no eixo x e com dois de seus cantos tocando o contêiner circular. A principal característica da Eq. (2) é que, se o retângulo não for empacotado (ou seja, αi = 0), a coordenada x será forçada a ser zero. (3) é a restrição equivalente à Eq. (2) para a coordenada y.Eqs. (4) - (7) assegure-se de que, se um retângulo estiver empacotado (assim, para retângulos com αi = 1), seu centro seja posicionado adequadamente de modo que o retângulo inteiro fique dentro do recipiente circular. Para conseguir isso, precisamos garantir que todos os quatro cantos do retângulo estejam dentro do contêiner circular. Esses quatro cantos são (xi ± Li / 2, yi ± Wi / 2) e Eqs. (4) - (7) assegure-se de que a distância (ao quadrado) da origem de cada um desses cantos não seja mais do que o raio (ao quadrado) do recipiente. Observe que, se o retângulo estiver compactado (então αi = 1), o lado esquerdo das Eqs. (4) - (7) é R2. Se o retângulo não estiver empacotado (então αi = 0), a partir das Eqs. (2) e (3) o retângulo está posicionado na origem (então tem xi = yi = 0). Nesse caso, o lado esquerdo das Eqs. (4) - (7) torna-se Li2 / 4 + Wi2 / 4, assim como o lado direito e, portanto, as restrições são satisfeitas automaticamente. Eq. (8) garante que quaisquer dois retângulos iej que sejam empacotados (então αi = αj = 1) não se sobreponham. Essa restrição deriva da dada anteriormente em Christofides (1974). Afirma que dois retângulos de tamanho [Li, Wi] e [Lj, Wj] não se sobrepõem, desde que a diferença entre suas coordenadas x centrais seja pelo menos (LL) / 2 ou que a diferença entre suas coordenadas y centrais seja + em jleast (Wi + Wj) / 2 (ou ambos). Se um ou outro retângulo não estiver empacotado, o lado esquerdo da Eq. (8) torna-se zero devido ao termo do produto (αiαj), o que significa que a restrição é automaticamente satisfeita. Eq. (9) é a restrição de integralidade. Conforme discutido acima, nossa formulação posiciona qualquer retângulo não empacotado na origem. Para retângulos desempacotados i, a inclusão de um termo αi inapropriado no lado esquerdo da Eq. (8) garante que esse retângulo descompactado, embora posicionado na origem, não participe ativamente da restrição de sobreposição que deve ser aplicada entre todos os retângulos compactados. Nossa formulação (Eqs. (1) - (9)) é um programa não linear de número inteiro misto (MINLP). Os MINLPs computacionalmente são reconhecidos como muito exigentes, envolvendo tanto um elemento de escolha combinatória quanto a solução de um programa não-linear contínuo subjacente. Para o problema considerado neste artigo, a escolha combinatória refere-se à escolha do conjunto de retângulos a serem empacotados, e o programa não-linear contínuo subjacente refere-se a determinar onde posicionar de maneira viável dentro dos retângulos circulares do recipiente circular que estão empacotados.

3.2. Elimination of the maximisation term

$$(x_i - L_i/2)^2 + (y_i + W_i/2)^2 \leq \alpha_i R^2 + (1 - \alpha_i)(L_i^2/4 + W_i^2/4)$$

The overlap constraint (Eq. (8)) contains the expression $\max\{|x_i - x_j| - (L_i + L_j)/2, |y_i - y_j| - (W_i + W_j)/2\}$. For the particular problem considered in this paper this maximisation term can be eliminated, albeit by enlarging the size of the MINLP to be solved.

Introduce additional continuous variables β_{ij}, $\forall(i, j) \in Q$, defined by:

$$0 \le \beta_{ij} \le 1 \qquad \forall(i, j) \in Q \tag{10}$$

Then we can replace Eq. (8) by:

$$\alpha_i\alpha_j \; \beta_{ij}[|x_i - x_j| - (L_i + L_j)/2] + (1 - \beta_{ij})[|y_i - y_j| - (W_i + W_j)/2] \ge 0 \qquad \forall(i, j) \in Q \tag{11}$$

The logic here is that the $\alpha_i\alpha_j$ term ensures that the Eq. (11) is always satisfied when either $\alpha_i = 0$ or $\alpha_j = 0$ (as indeed it does in Eq. (8)). It only remains to check therefore the validity of replacing Eq. (8) with Eq. (11) in the case $\alpha_i = \alpha_j = 1$.

When $\alpha_i = \alpha_j = 1$ Eq. (11) becomes $\beta_{ij}[|x_i - x_j| - (L_i + L_j)/2] + (1 - \beta_{ij})[|y_i - y_j| - (W_i + W_j)/2] \ge 0$. Now the weighted sum on the left-hand side of this constraint *can only be non-negative provided that at least one of the two terms in it is itself non-negative*. In other words Eq. (11) will ensure that one (or both) of $[|x_i - x_j| - (L_i + L_j)/2]$ and $[|y_i - y_j| - (W_i + W_j)/2]$ will be non-negative. Since one or both of these terms are non-negative it is therefore true that the maximisation term in Eq. (8), $\max\{|x_i - x_j| - (L_i + L_j)/2, |y_i - y_j| - (W_i + W_j)/2\}$, must also be non-negative. This in turn implies that Eq. (8) is satisfied. Therefore it is valid to replace Eq. (8) by Eq. (11).

Note here that it is also valid to replace Eq. (8) by Eq. (11) if we define β_{ij} as binary (zero-one) variables. However we might well expect there to be computational benefit in defining these vari-ables as continuous, rather than binary, variables.

3.3. Rotation

As is common in the literature (e.g. Birgin et al., 2010; Caprara et al., 2006; Hinostroza et al., 2013; Huang and Korf, 2012; Korf et al., 2010; Lodi et al., 2002; Maag et al., 2010; Martello and Monaci, 2015) in the basic formulation presented above we did not allow any rotation when packing, so that the items to be packed (rectangles/squares) were packed with their horizontal (length) edges parallel to the x-axis, their vertical (width) edges parallel to the y-axis. If rotation of any item is allowed (which might be dependent on the practical problem being modelled) then the sit-uation becomes more complex, although obviously rotation might enable a better solution to be found.

In the literature rotation through ninety degrees is the most common situation modelled (e.g. Birgin et al., 2010; Caprara et al., 2006; Delorme et al., 2017; Huang and Korf, 2012; Huang et al., 2007; Korf et al., 2010; Li et al., 2014; Martello and Monaci, 2015; Murata et al., 1996; Wu et al., 2007). Clearly rotation through ninety degrees is irrelevant when we are packing squares (as they are the same under ninety degree rotation) and only relevant when we are dealing with unequally sized rectangles. Our formulation can be extended to deal with rotation through ninety degrees as discussed below. Rotation through an arbitrary angle cannot be dealt with by our approach.

If the rectangles can be rotated through ninety degrees then this is easily incorporated into our formulation. Suppose that rect-angle i can be rotated through ninety degrees. Then create a new rectangle (j say) that represents rectangle i if it is rotated, so that we have $L_j = W_i$, $W_j = L_i$, $V_j = V_i$. Add to the formulation:

$$\alpha_i + \alpha_j \le 1 \tag{12}$$

Eq. (12) ensures that we cannot use both the original rectangle i and its rotated equivalent j. Dealing with rectangle rotation there-fore requires creating a new rectangle for each original rectangle that can be rotated and adding a single constraint to the formula-tion.

In terms of the effect on the formulation then if all n rectan-gles can be rotated this only directly adds n constraints (Eq. (12)) to the formulation. However the creation of an additional n rotated rectangles doubles the number of rectangles to be considered for packing. This means that the number of linear constraints associ-ated with Eqs. (2) and (3) doubles, as does the number of non-linear constraints associated with Eqs. (4)–(7). The more signifi-cant effect is that the number of nonlinear constraints associated with Eq. (8) increases from $n(n-1)/2$ to $2n(2n-1)/2$ (so approx-imately increases by a factor of 4). This increase in the number of nonlinear constraints associated with Eq. (8) also carries through to increase the number of β_{ij} variables (Eq. (10)) that need to be considered by an (approximate) factor of 4. For this reason we would expect that, computationally, dealing with a problem with n rectangles with fixed orientation becomes much more challeng-ing if all n rectangles can be rotated.

3.4. Maximising the number of squares packed

From our previous work López and Beasley (2016) we know that when we are considering a packing problem where all the items to be packed can be ordered such that item i fits inside item j for all $j > i$ then, in the case where we are maximising the num-ber of items packed, the optimal solution consists of the first K items, for some K.

Clearly items can be ordered to fit inside each other if we are considering packing squares, i.e. order the squares in increasing size (length) order, but such an ordering is unlikely to be possible if we are packing rectangles. Hence we shall just consider square packing here.

In the case of square packing therefore, when we are maximis-ing the number of squares packed, we can impose the additional constraints:

$$\alpha_{i-1} \ge \alpha_i \qquad i = 2, \ldots, n \tag{13}$$

$$\alpha_k = 0 \qquad \text{if } \sum_{i=1}^{k} L_i^2 > \pi R^2 \quad k = 1, \ldots, n \tag{14}$$

Eq. (13) ensures that if α_i is one (so square i is packed) then α_{i-1} must also be one (so square $i-1$ is packed). If square i is not packed ($\alpha_i = 0$) then the right-hand side of this constraint is zero, so the constraint is always satisfied whatever the value for α_{i-1}. Collectively the $(n-1)$ inequalities represented in Eq. (13) ensure that the optimal solution consists of the first K squares, for some K.

In Eq. (14) we have that if we were to pack square k then we would have to pack all squares up to and including square k. If this packing exceeds the area of the container then clearly square k cannot be packed.

Aside from these additional constraints we can amend the over-lap constraint, Eq. (11). Note that Eq. (11) includes a $\alpha_i\alpha_j$ term and applies for $(i, j) \in Q$, where Q is defined to have $j > i$. Now if $\alpha_j = 1$ we automatically know that $\alpha_i = 1$ (since $j > i$) and hence that $\alpha_i\alpha_j = 1$. If $\alpha_j = 0$ then it is irrelevant what value α_i takes since we must have $\alpha_i\alpha_j = 0$. In other words the $\alpha_i\alpha_j$ term in Eq. (11) can be replaced by α_j so that the overlap constraint, Eq. (11), becomes:

$$\alpha_j \; \beta_{ij}[|x_i - x_j| - (L_i + L_j)/2] + (1 - \beta_{ij})[|y_i - y_j| - (W_i + W_j)/2] \ge 0 \qquad \forall(i, j) \in Q \tag{15}$$

Note here that we have used W_i and W_j in Eq. (15) for clarity of comparison with Eq. (11). Obviously since we are just considering square packing here we have $W_i = L_i$ $i = 1, \ldots, n$.

4. FSS algorithm

In this section we present our FSS algorithm for the problem. For simplicity we present our approach using the basic formulation of the problem, Eqs. (1)–(9), before the amendments as discussed above (i.e. elimination of the maximisation term and adaptions for packing squares). We discuss at the end of this section how we in-corporate a number of other constraints, presented in this section, into our approach.

4.1. Algorithm

Consider the formulation, Eqs. (1)–(9), given above. Letting δ be a small positive constant replace the integrality requirement, Eq. (9), by:

$$\sum_{i=1}^{n} \alpha_i (1 - \alpha_i) \leq \delta \tag{16}$$

$$0 \leq \alpha_i \leq 1 \quad i = 1, \ldots, n \tag{17}$$

If δ was zero these equations would force $[\alpha_i,\ i = 1, \ldots, n]$ to assume zero-one values. However given the capabilities of non-linear optimisation software simply replacing an explicit integral-ity condition by Eqs. (16) and (17) would not be computation-ally successful, since we would be hoping to generate a (globally optimal) solution to a continuous nonlinear optimisation problem with a very tight inequality constraint. Note that if δ is zero then Eq. (16) is effectively an equality constraint as the left-hand side is non-negative.

Accordingly we adopt a heuristic approach and have $\delta > 0$. Hence our original MINLP, Eqs. (1)–(9), has now become a contin-uous nonlinear optimisation problem, since we have relaxed the integrality requirement using Eqs. (16) and (17). This nonlinear op-timisation problem is optimise Eq. (1) subject to Eqs. (2)–(8),(16) and (17). We refer to this problem as the *continuous FSS relaxation* of the problem.

If we solve this nonlinear problem the $[\alpha_i,\ i = 1, \ldots, n]$ can de-viate (albeit only slightly, if δ is small) from their ideal zero-one values, but we can round them to their nearest integer value to recover an integer set of values. Given an integer set of values for $[\alpha_i,\ i = 1, \ldots, n]$ then the original formulation (Eqs. (1)–(9)) becomes a nonlinear feasibility problem. This nonlinear feasibility problem is to find positions (x_i, y_i) for each rectangle i that we have chosen to pack (so with $\alpha_i = 1$ in the rounded solution). Note here that this is a feasibility problem as the objective function, Eq. (1), is purely a function of the zero-one variables (and these have been fixed by rounding).

With just a single value for δ we have just a single nonlin-ear problem: optimise Eq. (1) subject to Eqs. (2)–(8),(16) and (17). However changing δ in a systematic fashion creates a series of different problems that can be given to an appropriate nonlinear solver in an attempt to generate new and improved solutions to our original MINLP. The idea here is that altering δ perturbs the nonlinear formulation and hence, given the nature of any nonlin-ear solution software, might lead to a different solution.

The pseudocode for our FSS algorithm for the rectangle pack-ing problem considered in this paper is presented in Algorithm 1. In this pseudocode let P denote the original MINLP (here optimise Eq. (1) subject to Eqs. (2)–(9)) and P^* denote the continuous FSS relaxation (here optimise Eq. (1) subject to Eqs. (2)–(8),(16) and (17)).

We first initialise values, here Z_{best} is the best feasible solu-tion found and t is an iteration counter. We then solve the original MINLP P.

In this pseudocode all attempts to solve a nonlinear problem (e.g. P or P^*) must be subject to a time limit, since otherwise

Algorithm 1 Formulation space search pseudocode.

Initialisation: $\delta \leftarrow 0.05$ $Z_{best} \leftarrow -\infty$ $t \leftarrow 0$

Solve P and update Z_{best} if a feasible solution for P has been found

Iterative process:

while not termination condition **do** Update the
  iteration counter $t \leftarrow t + 1$ Solve P^*
  if a feasible solution for P^* has been found **then**
    Round the $[\alpha_i,\ i = 1, \ldots, n]$ values in the P^* solution and solve the resulting feasibility problem
    Update Z_{best} using the solution to the feasibility problem if a feasible solution for that problem has been found
  end if
  If $\delta \leq 10^{-5}$ or Z_{best} has not improved in the last three itera-tions stop
  Update $\delta \leftarrow \gamma\ \delta$
end while

the computation time consumed could become extremely high. For this reason we always terminate the solution process after a prede-fined time limit, returning the best feasible solution found (if one has been found). In the computational results reported later below this time limit was set to $10n$ seconds.

SCIP is capable of solving our MINLP formulation P to proven global optimality because SCIP restricts the type of nonlinear ex-pression allowed (Bussieck and Vigerske, 2018; Vigerske, 2017; Vigerske and Gleixner, 2018). However with regard to our compu-tational results for all the problem instances considered below this never occurred within the time limit imposed.

Note here that even if solving P or P^* returns a feasible solu-tion we have no guarantee that this is an optimal solution, since a better solution might have been found had we increased the time limit.

The iterative process in the pseudocode is to update the itera-tion counter and solve the continuous FSS relaxation P^*. If a feasi-ble solution for P^* has been found then we round that solution and solve the resulting feasibility problem (again subject to the prede-fined time limit). The best feasible solution found (if any) is up-dated and provided we have not reached the termination condition we reduce δ by a factor γ and repeat.

We terminate when δ is small ($\leq 10^{-5}$) or we have performed a number of consecutive iterations (three iterations) without im-proving the value of the best solution found. We reduce δ by a factor $\gamma = 0.5$ at each iteration and replicate (repeat) our heuristic a number of times (five replications were performed in the compu-tational results reported below). The values for these factors were set based on our previous computational experience with FSS.

4.2. Constraints

There are a number of general constraints that apply whatever the objective adopted. Recall that Z_{best} is the value of the best fea-sible solution encountered during our FSS heuristic. Let the set of rectangles that are packed in this best feasible solution be denoted by F. Then the general constraints that apply are:

$$\alpha_i + \alpha_j \leq 1 \qquad \forall (i, j) \in Q \ \ \min\big(L_i + L_j,\ W_i + W_j\big) > 2R \tag{18}$$

$$\sum_{i=1}^{n} \alpha_i L_i W_i \leq \pi R^2 \tag{19}$$

$$\sum_{i=1}^{n} \alpha_i V_i \geq Z_{best} \tag{20}$$

$$\sum_{i\in F} \alpha_i + \sum_{i\in F} (1 - \alpha_i) \ge 1 \tag{21}$$

Eq. (18) says that if the minimum of the sum of the sides of any two rectangles is greater than the container diameter then we cannot pack both rectangles. Eq. (19) ensures that the total area of the rectangles packed cannot exceed the area of the container. Eq. (20) ensures that the value of any solution found is at least that of the best feasible solution known. Eq. (21) is a feasible so-lution exclusion constraint and ensures that whatever solution is

found must differ from the best known solution (of value Z_{best} with packed rectangles F) by at least one rectangle. The effect of Eqs. (20) and (21) is to seek an improved feasible solution.

Note here that although these constraints may be redundant in the original MINLP they may not be redundant in any relaxation of the problem, in particular here the continuous FSS relaxation when we drop the requirement that the $[\alpha_i, i = 1, \ldots, n]$ are zero-one.

4.3. Summary

We have presented a considerable number of constraints above and so here (for clarity) we specify the constraints that are in-volved with P (the original MINLP) and P^* (the continuous FSS re-laxation) that are used in the statement of our FSS heuristic given above (see Algorithm 1).

- P is optimise Eq. (1) subject to Eqs. (2)–(7),(9)–(11),(18)–(21)
- P^* is optimise Eq. (1) subject to Eqs. (2)–(7),(10),(11),(16)–(21)

When considering just the packing of squares, so as to max-imise the number of squares packed, we add Eqs. (13) and (14) to P and P^* and replace Eq. (11) by Eq. (15). When rectangles can be rotated through ninety degrees we amend the problem in the man-ner discussed above when we considered Eq. (12).

5. Results

The computational results presented below (Windows 2.50GHz pc, Intel i5-2400S processor, 6Gb memory) are for our formula-tion space search heuristic as coded in FORTRAN. We used SCIP (Solving Constraint Integer Programs, version 4.0.1) (Achterberg, 2009; Maher et al., 2018; SCIP, 2018) as the mixed-integer non-linear solver. For a technical explanation as to how SCIP solves MINLPs see Vigerske and Gleixner (2018). To input our formulation into SCIP we made use of the modelling language ZIMPL (Zuse In-stitute Mathematical Programming Language), and to solve contin-uous nonlinear problems we used Ipopt (Interior Point OPTimizer, version 3.12.8), both of which are included within SCIP.

We generated a number of test problems involving n = 10, 20, 30 rectangles/squares, with rectangle/square dimensions be-ing randomly generated (to two decimal places) from [1, 5]. For each test problem we considered three different container radii, where the container radii R were set so that the area of the con-tainer (πR^2) was approximately $\frac{1}{3}$, $\frac{1}{2}$ and $\frac{2}{3}$ of the total area ($\sum_{i=1}^{n} L_i W_i$) of the n rectangles/squares. All of the randomly gen-erated test problems considered in this paper are publicly avail-able from OR-Library (Beasley, 1990), see http://people.brunel.ac.uk/~mastjjb/jeb/orlib/rspackinfo.html.

5.1. Rectangle packing, no rotation

Table 2 shows the results obtained for the rectangle packing test problems considered (where the rectangles have fixed orien-tation, so no rotation is allowed). In that table we show the value of n and the value of the container area fraction. For the two objec-tives considered (maximise the number of rectangles packed, max-imise the total area of the rectangles packed) we give the value of

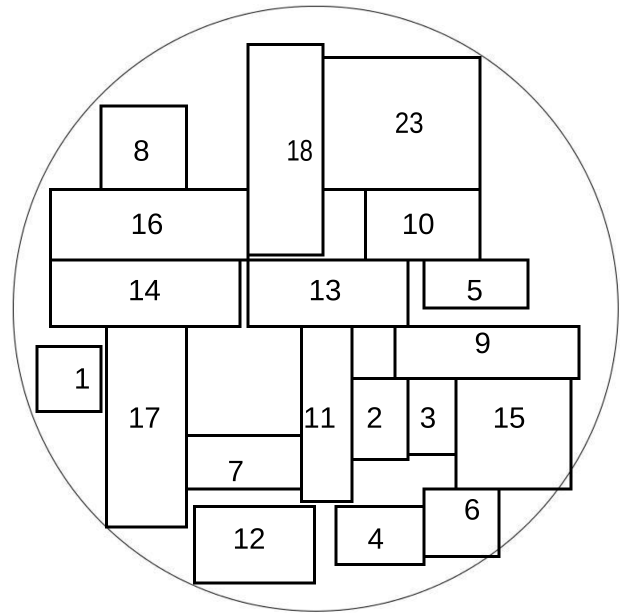

Fig. 5. Maximising number of rectangles packed, no rotation, n = 30, container area fraction $\frac{2}{3}$.

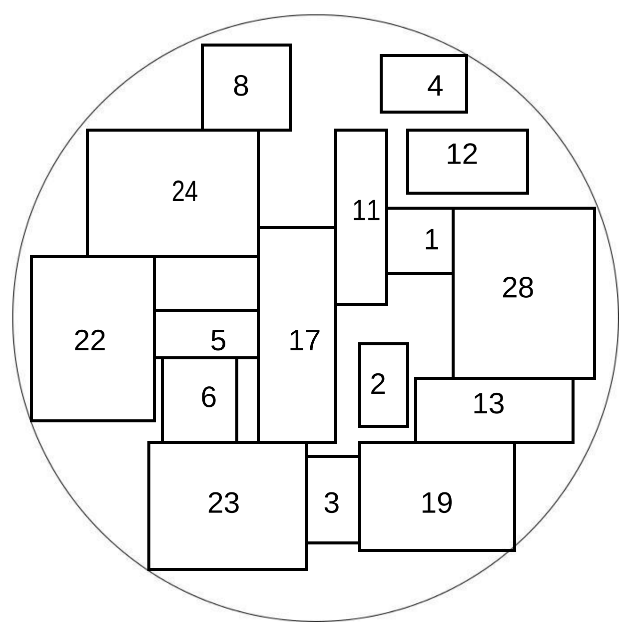

Fig. 6. Maximising total area of rectangles packed, no rotation, n = 30, container area fraction $\frac{2}{3}$.

the best solution achieved. We also show the replication at which we first encountered the best solution shown, as well as the total time (in seconds) over all five replications.

In Table 2 for a fixed n (and so a fixed set of rectangles to be packed) we can see that, as we would expect, as the container area fraction increases (so the container is of larger radius and we can hence pack more of the rectangles) the solution value also in-creases.

As an illustration of the results obtained Figs. 5 and 6 show the solutions in Table 2 for the two problems in that table with n = 30 and the largest container area fraction.

In Fig. 5 we can see that the solution consists of rectangles 1–18, together with rectangle 23. Recalling that rectangles are or-dered in increasing size (area) order this packing is as we would

Table 2
Computational results: rectangle packing, no rotation.

Number of rectangles (n)	Container area fraction	Maximise number			Maximise area		
		Best solution	Replication	Total time (s)	Best solution	Replication	Total time (s)
10	$\frac{1}{3}$	5	2	3058	18.4441	1	3292
	$\frac{1}{2}$	6	1	2862	28.9390	1	2992
	$\frac{2}{3}$	7	1	2966	37.6878	2	4754
20	$\frac{1}{3}$	7	1	6278	43.3885	1	7227
	$\frac{1}{2}$	10	5	4530	63.1643	1	9791
	$\frac{2}{3}$	11	1	7311	84.4446	2	10601
30	$\frac{1}{3}$	13	5	11514	60.3570	4	14011
	$\frac{1}{2}$	16	5	10029	85.2113	5	19786
	$\frac{2}{3}$	19	5	6966	103.4802	5	19470

Table 3
Computational results: square packing.

Number of rectangles (n)	Container area fraction	Maximise number			Maximise area		
		Best solution	Replication	Total time (s)	Best solution	Replication	Total time (s)
10	$\frac{1}{3}$	4	1	1123	22.9485	1	2762
	$\frac{1}{2}$	5	1	2761	36.7126	1	3402
	$\frac{2}{3}$	6	1	2275	51.7583	3	4593
20	$\frac{1}{3}$	11	5	5450	54.1054	5	9412
	$\frac{1}{2}$	12	1	6465	85.2107	4	11304
	$\frac{2}{3}$	14	1	6995	109.8363	5	7636
30	$\frac{1}{3}$	16	2	13552	54.4941	5	16629
	$\frac{1}{2}$	20	2	13457	77.5814	4	14808
	$\frac{2}{3}$	23	5	10427	103.0963	5	15145

expect, in that many of the smaller rectangles are used in a solu-tion that aims to maximise the number of rectangles packed.

In Fig. 6 we can see that the solution consists of a mix of rect-angles. The first six smallest rectangles, rectangles 1–6, together with rectangles 8,11–13,17,19,22–24,28. This figure contains 16 rect-angles in total, compared with the 19 rectangles used in Fig. 5.

5.2. Square packing

Table 3 shows the results obtained for the square packing test problems considered. This table has the same format as Table 2. As an illustration of the results obtained Figs. 7 and 8 show the solutions in Table 3 for the two problems in that table with $n = 30$ and the largest container area fraction.

For Fig. 7, since we are maximising the number of squares packed, the solution must consist of the first K squares, for some K (the squares being ordered in increasing size order). In Fig. 7 we can see that all squares up to and including square $K = 23$ are packed. Visually whether square 24, which must be at least as large as square 23 (and possibly larger), can also be packed into the circular container through judicious rearrangement of all of the currently positioned squares is unclear.

In Fig. 8 we can see that the packing consists of a mix of squares. Squares 1–19, which are the 19 smallest squares, are all packed along with the two of the larger squares, squares 22 and

28. This figure contains 21 squares in total, compared with the 23 squares used in Fig. 7.

5.3. Rectangle packing, rotation allowed

Table 4 shows the results obtained for the rectangle packing test problems considered in Table 2, but where rotation through ninety degrees is allowed. That table has the same format as Table 2.

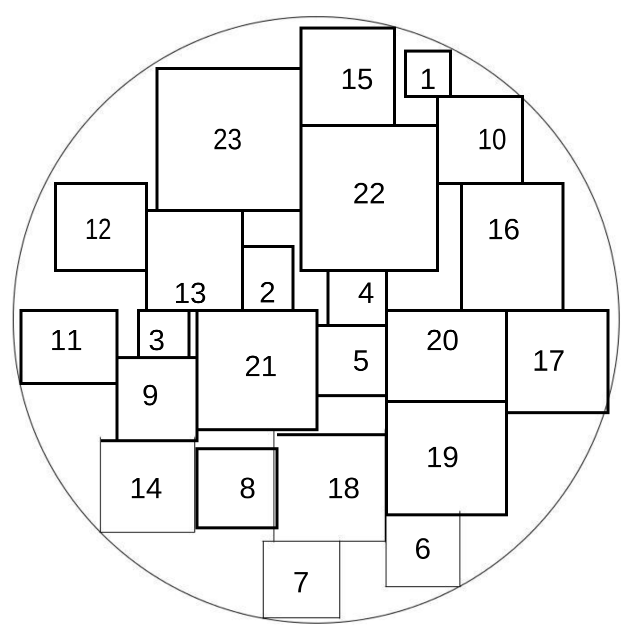

Fig. 7. Maximising number of squares packed, $n = 30$, container area fraction $\frac{2}{3}$.

As an illustration of the results obtained Figs. 9 and 10 show the solutions in Table 4 for the two problems in that table with 30 rectangles and the largest container area fraction. In those figures the letter r after the rectangle number indicates that the rectangle has been rotated through ninety degrees.

Comparing Table 4 with Table 2 we can see that the solution value where rotation is allowed is greater than (or equal to) the

Table 4
Computational results: rectangle packing, rotation allowed.

Number of rectangles	Container area fraction	Maximise number			Maximise area		
		Best solution	Replication	Total time (s)	Best solution	Replication	Total time (s)
10	$\frac{1}{3}$	5	1	9836	19.6702	1	8771
	$\frac{1}{2}$	6	1	10332	29.5041	1	16093
	$\frac{2}{3}$	7	1	12409	37.9687	2	15526
20	$\frac{1}{3}$	8	3	22759	43.6850	2	50558
	$\frac{1}{2}$	10	1	30682	63.5279	1	50013
	$\frac{2}{3}$	12	4	30823	84.7008	3	63350
30	$\frac{1}{3}$	14	1	49724	57.9328	5	69565
	$\frac{1}{2}$	17	1	45857	84.3715	1	82101
	$\frac{2}{3}$	20	1	57427	110.3253	3	39564

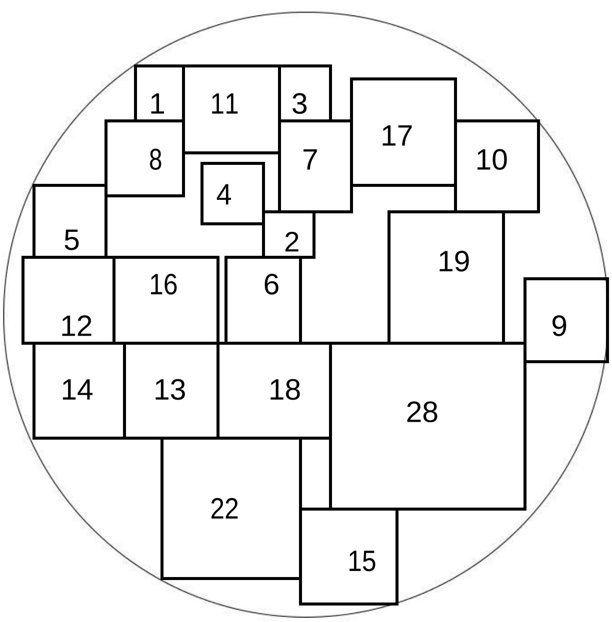

Fig. 8. Maximising total area of squares packed, n = 30, container area fraction $\frac{2}{3}$.

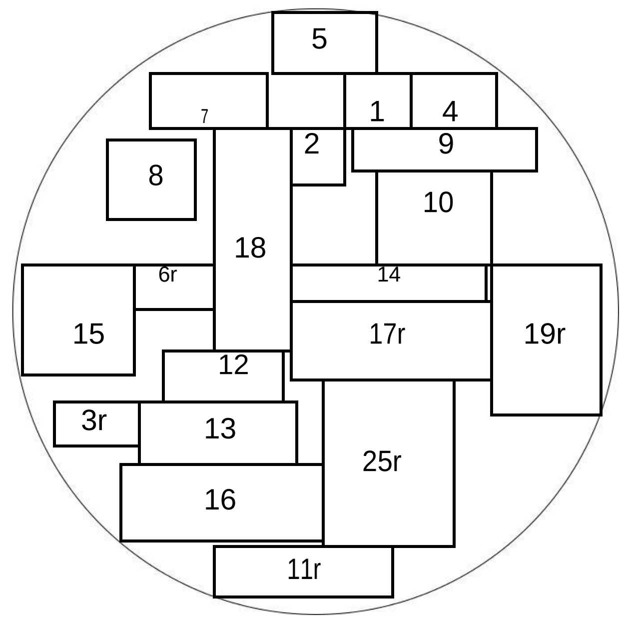

Fig. 10. Maximising total area of rectangles packed, rotation allowed, container area fraction $\frac{2}{3}$.

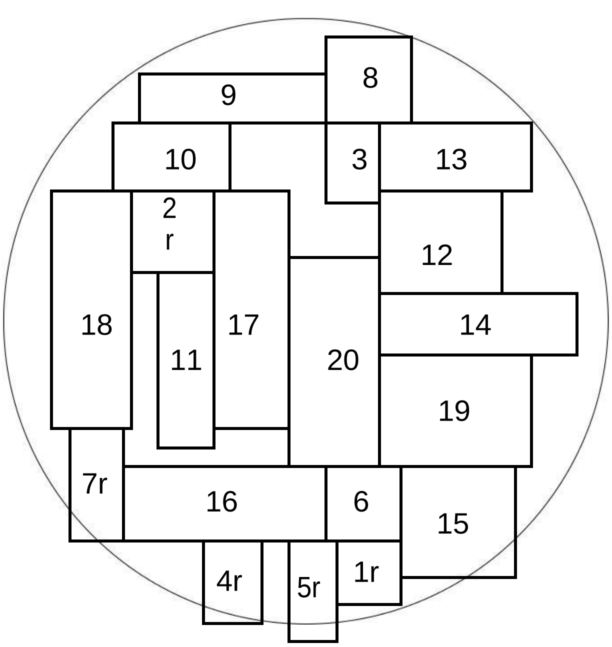

Fig. 9. Maximising number of rectangles packed, rotation allowed, container area fraction $\frac{2}{3}$.

solution value with no rotation for all but two of the 18 test prob-lems considered.

In the discussion above as to how to extend our formulation to deal with rotation through ninety degrees we noted the increase in the consequent size of the formulation, both with respect to the number of linear and nonlinear constraints and with respect to the number of variables. Comparing the computation times in Table 4 with those in Table 2 does indeed indicate that dealing with a problem where rectangles can be rotated is much more challenging computationally than dealing with a problem where the rectangles have fixed orientation.

5.4. Comment

As with many heuristic algorithms presented in the literature it is difficult to draw firm conclusions as to the quality of the results obtained without knowing either the optimal solutions of the test problems solved, or the results obtained by other heuristic algo-rithms by other authors on the same set of test problems.

For the problem considered in this paper we are not aware of any appropriate publicly available test problems which could be used to provide direct insight into the quality of our heuristic. We would stress here however that all of the test problems used in this paper are publicly available for use by future workers to see if

Table 5
Computational results: unit square packing.

Number of unit squares (n)	Best solution (number of unit squares packed)	Replication	Total time (s)
1	1	1	0.0
2	2	1	0.2
3	3	1	0.3
4	4	1	0.7
5	5	1	3.6
7	7	1	140.2
9	9	1	49.2
10	10	1	158.5
11	10	1	2619.5
12	12	1	120.2
14	14	1	460.6
16	16	5	4800.7
18	18	2	3153.6
21	21	5	4961.4
26	23	1	8756.0
30	27	1	18984.5

they can develop approaches that perform better than the formu-lation space search heuristic presented in this paper.

Despite this lack of appropriate test problems it is possible to gain some insight into the quality of our heuristic by taking test problems associated with a slightly different (but similar) problem. This problem is the problem of packing n unit squares within a circle of small (ideally minimal) radius. Here, unlike the problem we consider, all squares must be packed (whereas our heuristic is particularised for the case where one or more squares need not be packed).

We used our heuristic to maximise the number of unit squares packed into a circular container of known radius utilis-ing the test problems given by Friedman (2018). For these prob-lems Friedman (2018) gives the best solution known for the min-imum radius circle within which it is possible to pack all n unit squares. Some of these best known solutions involve arbitrary ro-tation (which our heuristic cannot deal with) and so we only con-sidered problems which did not involve rotation. Note also here that, as far as we aware, the results given in Friedman (2018) were found by varying authors using varying approaches (including, we believe, results based on human intervention). This contrasts with our results produced by a single algorithmic heuristic approach that does not involve any human intervention.

The results are shown in Table 5. In that table we show the number of unit squares (n) and the value of the best solution (maximum number of unit squares packed) as found by our heuris-tic. We also show the replication at which we first encountered the best solution shown, as well as the total time (in seconds) over all five replications. Considering Table 5 we can see that for 13 of the 16 problems considered our heuristic succeeds in finding the best known solution by packing all n unit squares into the given circu-lar container.

6. Conclusions and future work

In this paper we have formulated the problem of packing un-equal rectangles/squares into a fixed size circular container as a mixed-integer nonlinear program. We showed how we can elimi-nate a nonlinear maximisation term that arises in one of the con-straints in our formulation and indicated the amendments that can be made to the formulation when considering packing squares so as to maximise the number of squares packed.

We discussed how to amend our formulation to deal with the case where unequal rectangles can be rotated through ninety de-grees. A formulation space search heuristic was presented and computational results given for test problems involving up to 30 rectangles/squares, with these test problems being made publicly available for future workers.

In terms of future work we plan to investigate changes to our formulation, for example by making use of McCormick cuts to re-place products of variables.

Acknowledgments

The first author has a grant support from the programme UNAM-DGAPA-PAPIIT-IA106916

References

Achterberg, T., 2009. SCIP: Solving constraint integer programs. Math. Programm. Comput. 1 (1), 1–41.
Amirgaliyeva, Z., Mladenovic,´ N., Todosijevic,´ R., Urosevic,´ D., 2017. Solving the max-imum min-sum dispersion by alternating formulations of two different prob-lems. Eur. J. Oper. Res. 260 (2), 444–459.
Andrade, R., Birgin, E., 2013. Symmetry-breaking constraints for packing identical rectangles within polyhedra. Optim. Lett. 7 (2), 375–405.
Beasley, J.E., 1990. OR-library: distributing test problems by electronic mail. J. Oper. Res. Soc. 41 (11), 1069–1072.
Birgin, E.G., 2016. Applications of nonlinear programming to packing problems. In: Anderssen, R.S., Broadbridge, P., Fukumoto, Y., Kajiwara, K., Takagi, T., Verbit-skiy, E., Wakayama, M. (Eds.), Applications + Practical Conceptualization + Math-ematics = fruitful Innovation. Mathematics for Industry, vol 11. Springer, Tokyo, pp. 31–39.
Birgin, E.G., Lobato, R.D., 2010. Orthogonal packing of identical rectangles within isotropic convex regions. Comput. Industr. Eng. 59 (4), 595–602.
Birgin, E.G., Lobato, R.D., Morabito, R., 2010. An effective recursive partitioning ap-proach for the packing of identical rectangles in a rectangle. J. Oper. Res. Soc. 61 (2), 306–320.
Birgin, E.G., Martínez, J.M., Mascarenhas, W.F., Ronconi, D.P., 2006. Method of sen-tinels for packing items within arbitrary convex regions. J. Oper. Res. Soc. 57 (6), 735–756.
Birgin, E.G., Martínez, J.M., Nishihara, F.H., Ronconi, D.P., 2006. Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization. Comput. Oper. Res. 33 (12), 3535–3548.
Bortfeldt, A., 2013. A reduction approach for solving the rectangle packing area min-imization problem. Eur. J. Oper. Res. 224 (3), 486–496.
Brimberg, J., Drezner, Z., Mladenovic, N., Salhi, S., 2014. A new local search for con-tinuous location problems. Eur. J. Oper. Res. 232 (2), 256–265.
Bussieck, M.R., Vigerske, S., 2018. MINLP solver software. Wiley Encyclopaedia of Operations Research and Management Science. Wiley, New York. 2011. Updated version available from http://www2.mathematik.hu-berlin.de/∼stefan/ minlpsoft.pdf. Last accessed February 15th 2018.
Butenko, S., Yezerska, O., Balasundaram, B., 2013. Variable objective search. J. Heurist. 19 (4), 697–709.
Caprara, A., Lodi, A., Martello, S., Monaci, M., 2006. Packing into the smallest square: worst-case analysis of lower bounds. Discrete Optim. 3 (4), 317–326.
Cassioli, A., Locatelli, M., 2011. A heuristic approach for packing identical rectangles in convex regions. Comput. Oper. Res. 38 (9), 1342–1350.
Christofides, N., 1974. Optimal cutting of two-dimensional rectangular plates. In: In CAD 74, Proceedings of the International Conference on Computers in Engineer-ing and Building Design. Imperial College, London, UK, pp. 1–10. 25–27 Septem-ber 1974
Delorme, M., Iori, M., Martello, S., 2017. Logic based Benders' decomposition for or-thogonal stock cutting problems. Comput. Oper. Res. 78, 290–298.
Dowsland, K.A., Dowsland, W.B., 1992. Packing problems. Eur. J. Oper. Res. 56 (1), 2–14.
Duarte, A., Pantrigo, J.J., Pardo, E.G., Sánchez-Oro, J., 2016. Parallel variable neigh-bourhood search strategies for the cutwidth minimization problem. IMA J. Man-age. Math. 27 (1), 55–73.
Friedman, E., 2018. Squares in circles. http://www2.stetson.edu/∼efriedma/squincir/ Last accessed February 15th 2018.
Hansen, P., Mladenovic,´ N., Brimberg, J., Perez, J.A.M., 2010. Variable neighborhood search. In: Gendreau, M., Potvin, J.Y. (Eds.), Handbook of Metaheuristics, Vol. 146. Springer, pp. 61–86. International Series in Operations Research & Manage-ment Science.
Hansen, P., Mladenovic,´ N., Todosijevic,´ T., Hanafi, S., 2017. Variable neighborhood search: basics and variants. EURO J. Comput. Optim. 5 (3), 423–454.
Hertz, A., Plumettaz, M., Zufferey, N., 2008. Variable space search for graph coloring. Discrete Appl. Math. 156 (13), 2551–2560.
Hertz, A., Plumettaz, M., Zufferey, N., 2009. Corrigendum to "variable space search for graph coloring". Discrete Appl. Math. 157 (7), 1335–1336.
Hinostroza, I., Pradenas, L., Parada, V., 2013. Board cutting from logs: optimal and heuristic approaches for the problem of packing rectangles in a circle. Int. J. Prod. Econ. 145 (2), 541–546.
Huang, E., Korf, K.E., 2012. Optimal rectangle packing: an absolute placement ap-proach. J. Artif. Intell. Res. 46, 47–87.
Huang, W.Q., Chen, D.B., Xu, R.C., 2007. A new heuristic algorithm for rectangle packing. Comput. Oper. Res. 34 (11), 3270–3280.

Kochetov, Y., Kononova, P., Paschenko, M., 2008. Formulation space search approach for the teacher/class timetabling problem. Yugoslav J. Oper. Res. 18 (1), 1–11.

Korf, K.E., Moffitt, M.D., Pollack, M.E., 2010. Optimal rectangle packing. Ann. Oper. Res. 179, 261–295.

Leung, J.Y.T., Tam, T.W., Wong, C.S., Young, G.H., Chin, F.Y.L., 1990. Packing squares into a square. J. Parallel Distrib. Comput. 10 (3), 271–275.

Li, K., Cheng, K.H., 1989. Complexity of resource allocation and job scheduling prob-lems in partitionable mesh connected systems. In: Proceedings of the First An-nual IEEE Symposium of Parallel and Distributed Processing. IEEE Computer So-ciety, Silver Spring, MD, pp. 358–365.

Li, Z.Q., Wang, X.F., Tan, J.Y., Wang, Y.S., 2014. A quasiphysical and dynamic adjust-ment approach for packing the orthogonal unequal rectangles in a circle with a mass balance: satellite payload packing. Math. Probl. Eng. 2014 (657170).

Liu, D.Q., Teng, H.F., 1999. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. Eur. J. Oper. Res. 112 (2), 413–420.

Lodi, A., Martello, S., Monaci, M., 2002. Two-dimensional packing problems: a sur-vey. Eur. J. Oper. Res. 141 (2), 241–252.

López, C.O., Beasley, J.E., 2011. A heuristic for the circle packing problem with a variety of containers. Eur. J. Oper. Res. 214 (3), 512–525.

López, C.O., Beasley, J.E., 2013. Packing unequal circles using formulation space search. Comput. Oper. Res. 40 (5), 1276–1288.

López, C.O., Beasley, J.E., 2014. A note on solving MINLP's using formulation space search. Optim. Lett. 8 (3), 1167–1182.

López, C.O., Beasley, J.E., 2016. A formulation space search heuristic for packing un-equal circles in a fixed size circular container. Eur. J. Oper. Res. 251 (1), 65–73.

Maag, V., Berger, M., Winterfeld, A., Kufer, K.H., 2010. A novel non-linear approach to minimal area rectangular packing. Ann. Oper. Res. 179 (1), 243–260.

Maher, S. J., Fischer, T., Gally, T., Gamrath, G., Gleixner, A., Gottwald, R. L., Hendel, G., Koch, T., Lübbecke, M. E., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Weninger, D., Witt, J. T., Witzig, J., 2018. The SCIP optimization suite 4.0. ZIB Report 17-12 (March 2017, Revised September 2017). Available from https://opus4.kobv.de/opus4-zib/ files/6217/scipoptsuite-401.pdf. Last accessed February 15th 2018.

Martello, S., Monaci, M., 2015. Models and algorithms for packing rectangles into the smallest square. Comput. Oper. Res. 63, 161–171.

Mladenovic,´ N., Plastria, F., Uroševic,´ D., 2005. Reformulation descent applied to cir-cle packing problems. Comput. Oper. Res. 32 (9), 2419–2434.

Mladenovic,´ N., Plastria, F., Uroševic,´ D., 2007. Formulation space search for circle packing problems. In "Engineering stochastic local search algorithms. designing, implementing and analyzing effective heuristics". In: Proceedings of the Inter-national Workshop. SLS 2007, Brussels, Belgium, pp. 212–216. September 6–8, 2007. Lecture Notes in Computer Science volume 4638

Murata, H., Fujiyoshi, K., Nakatake, S., Kajitani, Y., 1996. VLSI module placement based on rectangle-packing by the sequence-pair. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 15 (12), 1518–1524.

Pardo, E.G., Mladenovic,´ N., Pantrigo, J.J., Duarte, A., 2013. Variable formulation search for the cutwidth minimization problem. Appl. Soft Comput. 13 (5), 2242–2252.

Picouleau, C., 1996. Worst-case analysis of fast heuristics for packing squares into a square. Theor. Comput. Sci. 164 (1–2), 59–72.

SCIP, 2018. Solving constraint integer programs. Available from http://scip.zib.de/ Last accessed February 15th 2018.

Sweeney, P.E., Paternoster, E.R., 1992. Cutting and packing problems: a categorized, application-orientated research bibliography. J. Oper. Res. Soc. 43 (7), 691–706.

Vigerske, S., 2017. Private communication. April.

Vigerske, S., Gleixner, A., 2018. SCIP: global optimization of mixed-integer non-linear programs in a branch-and-cut framework. ZIB Report 16–24 (May 2016). Available from https://opus4.kobv.de/opus4-zib/frontdoor/index/index/ docId/5937. Last accessed February 15th 2018.

Wascher, G., Haußner, H., Schumann, H., 2002. An improved typology of cutting and packing problems. Eur. J. Oper. Res. 183 (3), 1109–1130.

Wu, Y.L., Huang, W.Q., Lau, S.C., Wong, C.K., Young, G.H., 2007. An effective quasi-hu-man based heuristic for solving the rectangle packing problem. Eur. J. Oper. Res. 141 (2), 341–358.