J Glob Optim (2009) 43:299–328
DOI 10.1007/s10898-007-9274-6

Cutting circles and polygons from area-minimizing rectangles

Josef Kallrath

Received: 30 June 2007 / Accepted: 19 December 2007 / Published online: 17 January 2008

Abstract A set of circles, rectangles, and convex polygons are to be cut from rectangular design plates to be produced, or from a set of stocked rectangles of known geometric dimensions. The objective is to minimize the area of the design rectangles. The design plates are subject to lower and upper bounds of their widths and lengths. The objects are free of any orientation restrictions. If all nested objects fit into one design or stocked plate the problem is formulated and solved as a nonconvex nonlinear programming problem. If the number of objects cannot be cut from a single plate, additional integer variables are needed to represent the allocation problem leading to a nonconvex mixed integer nonlinear optimization problem. This is the first time that circles and arbitrary convex polygons are treated simultaneously in this context. We present exact mathematical programming solutions to both the design and allocation problem. For small number of objects to be cut we compute globally optimal solutions. One key idea in the developed NLP and MINLP models is to use separating hyperplanes to ensure that rectangles and polygons do not overlap with each other or with the circles. Another important idea used when dealing with several resource rectangles is to develop a model formulation which connects the binary variables only to the variables representing the center of the circles or the vertices of the polytopes but not to the non-overlap or shape constraints. We support the solution process by symmetry breaking constraints. In addition we compute lower bounds, which are constructed by a relaxed model in which each polygon is replaced by the largest circle fitting into that polygon. We have successfully applied several solution techniques to solve this problem among them the Branch&Reduce Optimization Navigator (`BARON`) and the `LindoGlobal` solver called from `GAMS`, and, as described in Rebennack et al. [21], a column enumeration approach in which the columns represent the assignments. Good feasible solutions are computed within seconds or minutes

J. Kallrath (✉)
BASF-AG, Scientific Computing, GVC/S-B009, 67056 Ludwigshafen, Germany
e-mail: josef.kallrath@web.de

J. Kallrath
Department of Astronomy, The University of Florida, Bryan Space Science Building, Gainesville, FL 32611, USA
e-mail: kallrath@astro.ufl.edu

usually during preprocessing. In most cases they turn out to be globally optimal. For up to 10 circles, we prove global optimality up to a gap of the order of 10^{-8} in short time. Cases with a modest number of objects, for instance, 6 circles and 3 rectangles, are also solved in short time to global optimality. For test instances involving non-rectangular polygons it is difficult to obtain small gaps. In such cases we are content to obtain gaps of the order of 10%.

Keywords Global Optimization · Mixed integer programming · Cutting stock problem · Packing problem · Shape constraints · Non-overlap constraints · Design problem · Assignment

1 Introduction

A set of circles and convex polygons (often rectangles), hereafter called *objects*, are to be cut from rectangular plates. The objects are free of orientation restrictions. There are two cases to be distinguished: (a) the plates are to be produced, or (b) they are already available on stock. Both the plates to be produced and the plates available on stock are hereafter called *resource plates*. In case (a) the plates are considered as design rectangles whose width and length are to be determined; hereafter called *design* and *production* case. The design plates are subject to lower and upper bounds of their widths and lengths. The number of design rectangles is not known a priori. The objective is to minimize the area of the design rectangles; this is equivalent to minimize trimloss. In case (b) the objects should be cut from a set of up to 50 stocked rectangles of known geometric dimensions, i.e., we have to solve the cutting problem and an additional *assignment problem*. After (a) or (b) have been solved, in a second step, the objects should be arranged in the plates in such a way that the remaining waste area contains a rectangle of maximum size; this remaining rectangle becomes a new stocked plate.

Both problems (a) and (b) fall into the class of two-dimensional cutting or packing problems of regular objects. They come close to the Dyckhoff [8] classification 2/V/D/F; i.e., two-dimensional, V = a kind of assignment: a selection of objects and all items, D = an assortment of large objects: different figures, and F = an assortment of small items: few items of different figures. There is a vast amount of publications in the framework of discrete or computational geometry related to congruent circle packing into squares or densest circle packing; *cf*. Szabo et al. [27]. However, the literature reveals that there exist no publications in which circles of unequal radii *and* orientation-free rectangles or polygons are packed simultaneously into rectangles. If subproblems are treated, i.e., only circles, only rectangles, or only for given resource plates of known size, then no claim of computing globally optimal solutions is made. Regarding our design problem, there exist a few publications in Russian language packing circles into one area-minimizing rectangle; they are referenced in Stoyan and Yaskov [26]. Among them are the papers by Rvachev and Stoyan [23,24] who compute exactly a series of local optima for one area-minimizing rectangle hosting a set of given circles. The most recent publication on packing circles into area-minimizing rectangles (and other geometric forms) is the work by Birgin and Sobral [5]. Stoyan and Yaskov [26] do not claim to compute the global optimum. There are a few more publications on packing *equal* circles into *one* area-minimizing rectangle; cf. Ruda [22] or Lubachevsky and Graham [18]. Although limited to equal circles and not minimizing the area of the rectangle, the work by Lubachevsky and Graham [19] is worth to mention, who computed minimum perimeter rectangles enclosing congruent non-overlapping circles. An interesting approach is that by Yu and Zhang [29] who formulated the problem of packing a given set of different-sized circles into the smallest possible square box as a nonlinear programming (NLP) problem

and established the first order optimality conditions. The augmented Lagrangian method is applied to solve this problem.

We found more publications treating the problem of fitting different-sized circles into rectangles of given size. Fraser and George [10] discuss packing circles of the same size in a container of fixed dimensions. George et al. [11] formulated a mixed integer nonlinear programming (MINLP) problem for packing different-sized pipes into a rectangular container which is equivalent to packing unequal circles into rectangles. They also addressed the problem of how to allocate pipes to various containers in a shipment in order to minimize the number of containers. They developed a number of heuristic procedures and a genetic algorithm for (approximately) solving this problem. Because of the container-shipping background of their problem, they also discussed the stability of packing solutions in their excellent paper. Stoyan and Yaskov [25] discussed and developed exact and approximate algorithms to compute the global optimum of placing either rectangles or circles into a given rectangle but not rectangles and circles simultaneously. Although, they did not consider the case with both circles and rectangles to be placed, this paper is very much recommend to the reader because it contains useful analytical results and also reviews many results obtained by Russian and Ukrainian researchers among them Rvachev. Stoyan and Yaskov [26] extended their approach to strip packing of circles into one rectangle of fixed width and height to be minimized. Huang et al. [13] developed a greedy algorithm for packing unequal circles into given rectangles. The problem of packing circles into given rectangles has been shown to be NP-hard; cf. Lenstra and Rinnooy [15]. Although already 15 years old and mostly on packing into given rectangles, it is still inspiring to read the invited review article by Dowsland and Dowsland [7] on packing problems.

The publications on packing rectangles into rectangles are mostly concerned with axis-parallel or orthogonal packings. A recent very interesting work is that by Birgin et al. [4]. Packing polygons into given rectangles is a field in which only heuristic methods have been used. Jakobs [14] proposed a genetic algorithm for placing polygons into a rectangle in which in a pre-step each polygon is embedded into a rectangle of minimum area. This step is followed by the main step of packing rectangles into one rectangle.

The structure of this article is as follows. In Sect. 2 we develop an NLP model for cutting objects from one design rectangle. The model is extended in Sect. 3 to allow for a modest number of design rectangles and also the assignment/selection problem if the objects need to be assigned to a set of given rectangles (cutting from, or packing into several rectangles). This MINLP problem can be solved to global optimality only for small instances. Numerical experiments and results are discussed in Sect. 4. At all places in this paper where we use the term global optimum, or global optimality, we use it in the sense of small gaps of the order of 10^{-8}, and we are aware that those are also subject to the limits of the numeric solvers dealing with finite number arithmetic subject to round-off errors.

2 Modeling: cutting from one rectangle

If all objects fit into one design or given plate the problem can be described as a nonconvex NLP problem. For circles, the only variables are the coordinates of the center. Rectangles and polygons are described by the coordinates of their vertices. Shape constraints ensure that the variables representing the vertices reproduce the original geometrical objects. The model consists of non-overlap constraints and the constraints ensuring that all objects do not exceed the bounds of the enclosing rectangle. Non-overlap of circles by enforcing that their centers are apart not less than the sum of their radii. Polygons are ensured not to overlap

with other objects by separating lines (in general hyperplanes) exploiting the fact that we are dealing with convex objects. In the model formulation below we put all relations into numbered equations which are either referred to in the text, or which appear in the model and have been coded in GAMS [cf. Brooke et al. [6]). Where possible we try to use a vector notation using the Euclidean norm scalar products avoiding the additional dimension index d. We use lower case symbols for variables, and upper case symbols for input or derived data.

2.1 Indices

used in this model:

$d \in \mathcal{D} := \{1, \ldots, |\mathcal{D}|\}$ dimension of the space.
If we consider only circles, the model is generic for an arbitrary number, $|\mathcal{D}|$, of dimensions. For the current case we work in a two dimensional space with width ($d = 1$) and length ($d = 2$).

$i \in \mathcal{I} := \{i_1, \ldots, i_{|\mathcal{I}|}\}$ circles to be packed.
Circles are characterized by their radii R_i.

$j \in \mathcal{J} := \{r_1, \ldots, r_{|\mathcal{J}|}\}$ rectangles to be packed.
Rectangles are characterized by width W_j and length L_j. Rectangles are treated different from non-rectangular polygons in that different shape constraints are used.

$k \in \mathcal{K} := \{k_1, \ldots, k_{|\mathcal{K}|}\}$ index counting the vertices of the rectangles ($|\mathcal{K}| = 4$) or polygons.
We identify vertex $V_{j,|\mathcal{K}|+1}$ with V_{j1}.

$p \in \mathcal{P} := \{p_1, \ldots, p_{|\mathcal{P}|}\}$ polygons to be packed.
Polygons are characterized by the number of vertices, K_p, and the coordinates of their vertices V_{pk}, $k = 1, \ldots, K_p$. The orientation of placed polygons w.r.t. their reference orientation is described by the angle α_p measuring the anti-clockwise rotation of polygon p. Using only one orientation angle limits us to the two-dimensional case.

$r \in \mathcal{R} := \{r_1, \ldots, r_{|\mathcal{R}|}\}$ resource rectangles to be produced or available on stock.
The number of resource rectangles, $|\mathcal{R}|$ varies between 1 and 50. They are assumed to be ordered in a given sequence. The set $\mathcal{R}_*$ denotes the set of all resource rectangles except the last one.

2.2 Input data

The input data consists of the following geometric data:

A_i [L^2] the areas of all circles i; quantity derived as $A_i = \pi R_i^2$. [L^2] specifies that A_i has the physical dimension *area*.

A_j [L^2] the areas of all rectangles j; quantity derived as $A_j = S_{j1} S_{j2}$.

A_p [L^2] the areas of all polygons p; quantity derived by formula (2.30).

D [L] the maximum possible length of the diagonal of the design rectangle.

K_p [−] the number of vertices of polygon p; [−] specifies that K_p is a *dimensionless* quantity.

L [L] maximum size (upper bound) of the length of the design rectangle; [L] specifies that L has the physical dimension *length*.

L_r^0 [L] the accumulated length of stocked rectangles up to resource rectangle r.

R_i [L] the radius of circle i.

S_{jd} [L] the extension of rectangle j, i.e., width S_{j1} and length S_{j2}.

S_d^+ [L] maximum size (upper bound) of the extension of the design rectangles in dimension d.

S_d^- [L] minimum size (lower bound) of the extension of the design rectangles in dimension d.

W [L] maximum size (upper bound) of the width of the design rectangle.

X_{pkd} [L] the coordinates of vertex V_{pk} of polygon p; in vector notation $\mathbf{X}_{pk}$. The ordering is such that k and $k+1$ refer to adjacent vertices.
The coordinates X_{pkd} define the reference orientation of polygon p.

X_{pd}^0 [L] the center coordinates of polygon p; in vector notation $\mathbf{X}_p^0$.

X_{rd}^{SR} [L] the extension of resource rectangle r in dimension d. In the case of design rectangles this is the maximum extension in the sense of an upper bound; for stocked rectangles it is the given size.

2.3 Variables

used in the different models:

$a \in [S_1^- S_2^-, S_1^+ S_2^+]$ [L^2] the area of the design rectangle. a^- and a^+ are lower and upper bounds on a obtained during the computation.
In more than two dimensions, $S_1^- S_2^-$ and $S_1^+ S_2^+$ are replaced by $\prod_{d=1}^{|\mathcal{D}|} S_d^-$ and $\prod_{d=1}^{|\mathcal{D}|} S_d^+$, respectively. The minimal and maximal extensions, S_1^-, S_2^-, and S_1^+, S_2^+, are machine dependent production constraints.

$\alpha_p \in [0°, 360°]$ [deg] the orientation angle of polygon p.
For rectangles, the range of α_p is reduced to $[0°, 180°]$.

$\Delta_{pp'k} \in [0, D]$ [L] the distances of the vertices to the separating line separating the polygons p and p'.
These variables are bounded by the diagonal, D, of the design rectangle.

$\delta_{ir} \in \{0, 1\}$ $[-]$ the binary variables δ_{ir} decide on the allocation of object i to design rectangle or stocked rectangle r.
These binary variables are only needed in Sect. 3 to model cutting from and object allocation to several rectangles available on stock.

$\mathbf{g}_{pp'} \in [0, S_1^+] \times [0, S_2^+]$ $[L \times L]$ the footing point vector of the separating line between the polygons p and p'.
The vector variable $\mathbf{g}_{pp'}$ leads to the scalar variables $g_{pp'd}$.

$\ell_r^0 \in [0, L_r^0]$ $[L]$ the starting length of resource rectangle or segment r.
These auxiliary variables are only needed in Sect. 3 to model cutting from and object allocation to several rectangles available on stock.

$\ell_{ir}^\delta \in [0, L_r^0]$ $[L]$ the product $\ell_{ir}^\delta := \ell_r^0 \delta_{ir}$.
These auxiliary variables are only needed in Sect. 3 to model cutting from and object allocation to several rectangles available on stock.

$\lambda_{pp'k} \in [-D, D]$ $[L]$ auxiliary variables needed to compute the vectors $\mathbf{p}_{pp'k}$ connecting vertex V_{pk} with the line separating polygon p and polygon p'.
These auxiliary variables are free variables and can take positive and negative values.

$\mathbf{m}_{pp'} \in [-1, +1] \times [-1, +1$ $[-]^2$ the direction vector of the separating line between the polygons p and p'.
The vector variable $\mathbf{m}_{pp'}$ leads to the scalar variables $m_{pp'd}$.

$\mathbf{n}_{pp'} \in [-1, +1] \times [-1, +1]$ $[-]^2$ the normal vector of the separating line between the polygons p and p'.
The vector variable $\mathbf{n}_{pp'}$ leads to the scalar variables $n_{pp'd}$.

$\mathbf{p}_{pp'k} \in [0, D] \times [0, D]$ $[L]$ connection from the separating line $\mathbf{G}_{pp'}$ to vertex V_{pk} of polygon p and p'.
The vector variable $\mathbf{p}_{pp'k}$ leads to the scalar variables $p_{pp'kd}$.

r_p $[L]$ radius of the smallest circle enclosing polygons p.
The variable is only used in the auxiliary model computing the smallest circle enclosing polygons p.

$\sigma_r \in \{0, 1\}$ $[-]$ binary variables indicating the usage of rectangle r.
The binary variables σ_r are only needed in Sect. 3 to model cutting from and object allocation to several rectangles available on stock.

$v_{jkd} \in [-S_{jd}, S_{jd}]$ $[L]$ components of the vector $\mathbf{v}_{jk}$ pointing from vertex k to vertex $k+1$ of rectangle j.
The vectors $\mathbf{v}_{jk}$ support the representation of the rectangular objects by exploiting the fact that a rectangle has two parallel sides with $|\mathbf{v}_{j1}| = S_{j1}$ and $|\mathbf{v}_{j2}| = S_{j2}$, and that adjacent sides are orthogonal to each other, i.e., $\mathbf{v}_{jk}\mathbf{v}_{jk+1} = 0$; here $|\cdot|$ denotes the Euclidean norm, and the orthogonality is expressed by the vanishing scalar product.

$x_{id} \in [0, S_d^+]$ $[L]$ the coordinates of the center vector, $\mathbf{x}_i$, of circle i.
For circles i with radius $2R_i \leq \min_d\{S_d^+\}$ the bounds can be refined to $[R_i, S_d^+ - R_i]$.

$x_{pkd} \in [0, S_d^+]$ $[L]$ the coordinates of the vertex k of polygon p; in vectorial notation $\mathbf{x}_{pk}$.
The vector $\mathbf{x}_{pk}$ is obtained by rotation and translation of the original vertex vector $\mathbf{X}_{pk}$.

$x_{pd} \in [0, S_d^+]$ $[L]$ the center coordinates of polygon p; in vectorial notation $\mathbf{x}_p^0$.
The vector $\mathbf{x}_p^0$ serves as the basis to re-construct polygon p.

x_{rd}^{DR} $[L]$ the extension of design rectangle r.
These variables are only needed in Sect. 3 to model cutting from and object allocation to several rectangles available on stock.

$x_d^{\mathrm{P}} \in [0, S_d^+]$ $[L]$ the extension of the design rectangle in dimension d.
There are pre-given upper bounds on these extensions.

$x_{jkd}^{\mathrm{R}} \in [0, S_d^+]$ $[L]$ the coordinates of the vertex vector $\mathbf{x}_{jk}^{\mathrm{R}}$ of rectangle j.
The variables are only bounded by the size of the design rectangle.

$z \in [0, S_1^+ S_2^+]$ $[L^2]$ the objective function, trimloss or waste, associated to the optimal solution.
This variable is defined as $z = a - \sum_i A_i - \sum_j A_j - \sum_p A_p$.

Note that we provide lower and upper bounds in the model wherever possible and as tight as possible as these bounds help to solve the NLP and MINLP problems to global optimality.

2.4 Model

The objective function to be minimized is the area, a, of the design rectangle

$$\min a, \quad a = \prod_{d=1}^{|\mathcal{D}|} x_d^{\mathrm{P}}, \tag{2.1}$$

where x_d^{P} represents the extension of the design rectangle in dimension d. Equivalent to this is to minimize waste, i.e.,

$$\min z, \quad z = a - \sum_i A_i - \sum_j A_j - \sum_p A_p, \tag{2.2}$$

where A_i, A_j, and A_p denote the known areas of circles, rectangles, and polygons.

The extensions are subject to the bounds

$$S_d^- \leq x_d^{\mathrm{P}} \leq S_d^+, \quad \forall d. \tag{2.3}$$

In the two-dimensional cases considered in this publication x_1^{P} is width w, and x_2^{P} is length ℓ.

2.4.1 Cutting circles

For all objects we have to guarantee that they do not overlap with other objects. For circles the non-overlap constraints simply read

$$(\mathbf{x}_i - \mathbf{x}_{i'})^2 \geq (R_i + R_{i'})^2, \quad \forall\{(i, i') \mid i < i'\}. \tag{2.4}$$

Note that for n circles we have $n(n-1)/2$ inequalities of type (2.4).

Fitting the circles inside the enclosing rectangles requires

$$x_{id} \geq R_i; \quad \forall\{i, d\}. \tag{2.5}$$

and

$$x_{id} + R_i \leq x_d^{\mathrm{P}} \leq S_d^{+}; \quad \forall\{i, d\}. \tag{2.6}$$

2.4.2 Cutting rectangles

The rectangles are described by the vertices x_{jkd}^{R} and the shape constraints (2.8)–(2.11). The rectangle fits into the enclosing rectangles if all corner points are inside the design rectangle, i.e.,

$$0 \leq x_{jkd}^{\mathrm{R}} \leq x_d^{\mathrm{P}} \leq S_d^{+}; \quad \forall\{j, k, d\}. \tag{2.7}$$

To describe rectangle j and establish its proper shape we introduce the vectors $\mathbf{v}_{jk}$ pointing from corner k to corner $k+1$

$$\mathbf{v}_{jk} = x_{jk+1}^{\mathrm{R}} - x_{jk}^{\mathrm{R}}; \quad \forall\{j, k\}, \tag{2.8}$$

where k_5 is identified with k_1; see Fig. 1. Note that (2.8) only connects the corner points. However, it could happen that for instance $\mathbf{v}_{j1}$ and $\mathbf{v}_{j3}$ *cross* each other. In order to avoid such situations we add the following constraints to ensure that we obtain the appropriate shape, i.e., a rectangle with four 90° angles. At first we establish orthogonality between $\mathbf{v}_{j1}$ and $\mathbf{v}_{j2}$, i.e.,

$$\mathbf{v}_{j1}\mathbf{v}_{j2} = \sum_d v_{j1d} v_{j2d} = 0; \quad \forall\{j\}, \tag{2.9}$$

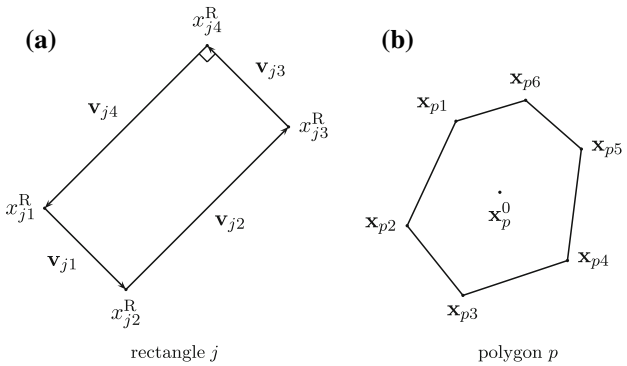

Fig. 1 Representation of rectangles and polygons

and then we require that $\mathbf{v}_{j3}$ and $\mathbf{v}_{j1}$, or $\mathbf{v}_{j4}$ and $\mathbf{v}_{j2}$, resp., are anti-parallel, i.e.,

$$v_{jkd} = -v_{j,k-2,d}; \quad \forall\{jkd|k > 2\}. \tag{2.10}$$

This representation takes care of the orientation automatically. Finally, we ensure that the size of the rectangle is established by

$$\left\|\mathbf{v}_{jk}^2\right\|^2 = \mathbf{v}_{jk}^2 = \sum_d v_{jkd} v_{jkd} = S_{jk}^2; \quad \forall\{j,k|k < 3\}. \tag{2.11}$$

Note that if we to arrange the objects in the design rectangle in such a was to leave a maximum rectangular area free, we just need to replace the known size S_{jk}^2 in (2.11) by variables s_{jd}^{FR} denoting the unknown size of that inner free rectangle; the objective function to be maximized is $s_{j1}^{\mathrm{FR}} s_{j2}^{\mathrm{FR}}$.

2.4.3 Cutting polygons

A polygon p is characterized by its K_p vertices $V_{p1}, \ldots, V_{pK_p}$, i.e., by their coordinates $\mathbf{X}_{pk}, k = 1, \ldots, K_p$, where the index p counts the available polygons, and K_p specifies the number of vertices of polygon p. While our application is two-dimensional, the formulation presented below is in most parts generic to can be extended to higher dimensions. Some care is needed with the neighborhood relationship of vertices, and also with the angular representation (2.14), e.g., in three dimensions we would need two angles.

The polygons will be implicitly described by their centers, the distances of the vertices to the center, and the orientations; see Fig. 1. The center, $\mathbf{X}_p^0$, of the original polygon is defined by

$$\mathbf{X}_p^0 = \frac{1}{K_p} \sum_{k=1}^{K_p} \mathbf{X}_{pk}; \quad \forall\{p\}. \tag{2.12}$$

The polygons can now be placed at a free center represented by the vector $\mathbf{x}_q^0$

$$\mathbf{x}_p^0 = \frac{1}{K_p} \sum_{k=1}^{K_p} \mathbf{x}_{pk}; \quad \forall\{p\}. \tag{2.13}$$

subject to the shape and orientation constraint

$$\mathbf{x}_{pk} = \mathbf{x}_p^0 + \begin{pmatrix} \cos\alpha_p & \sin\alpha_p \\ -\sin\alpha_p & \cos\alpha_p \end{pmatrix} \left(\mathbf{X}_{pk} - \mathbf{X}_p^0\right); \quad \forall\{p,k\}. \tag{2.14}$$

Instead of α as the free variable, we take $-1 \le \cos\alpha \le +1$ and $-1 \le \sin\alpha \le +1$ as the free variables coupled by

$$\sin^2\alpha_p + \cos^2\alpha_p = 1; \quad \forall\{p\}. \tag{2.15}$$

Note that for polygons with a symmetry axis we need to consider only rotation angles in the range of $0°$ to $180°$, i.e., $0 \le \sin\alpha \le 1$. Further symmetry might be exploited in special cases, e.g., regular polygons with n equal sides.

Fitting the polygons inside the enclosing rectangles requires that

$$X_{pkd} \le x_d^{\mathrm{P}} \le S_{\max,d}; \quad \forall\{p,k,d\}. \tag{2.16}$$

2.4.4 Non-overlap constraints for polygons

Non-overlap of polygon p and any other polygon q' is enforced by the condition that all vertices of q and q' are on different sides of, or on a separating line or hyperplane in higher dimensions; see Fig. 2. Let polygon p and p' have K_p and $K_{p'}$ vertices, respectively; note that K_p and $K_{p'}$ may be different. The line, $\mathbf{G}_{pp'}$, separating the vertices of polygon p and p' involves the variables $\mathbf{g}_{pp'}$, $\mathbf{m}_{pp'}$, and $\lambda_{pp'}$ per polygon combination pp' and is given by

$$\mathbf{G}_{pp'} := G_{pp'}(\lambda) = \mathbf{g}_{pp'} + \mathbf{m}_{pp'}\lambda_{pp'}; \quad \forall\{p, p' | p' > p\}, \tag{2.17}$$

where $\lambda \in \mathbb{R}$ parametrizes the line, and the direction vector $\mathbf{m}_{qq'}$ is normalized to unity, i.e.,

$$\mathbf{m}_{pp'}^2 = 1; \quad \forall\{p, p' | p' > p\}. \tag{2.18}$$

The normal direction, $\mathbf{n}_{pp}$, to $\mathbf{G}_{pp'}$ is given by

$$\left(n_{pp'1}, n_{pp'2}\right)^{\mathrm{T}} = \left(m_{pp'2}, -m_{pp'1}\right)^{\mathrm{T}}; \quad \forall\{p, p' | p' > p\}. \tag{2.19}$$

The K_p connection vectors $\mathbf{p}_{pp'k}$ from $\mathbf{G}_{pp'}$ to vertex V_{pk} of polygon p are given by

$$\mathbf{p}_{pp'k} = \mathbf{x}_{pk} - \left(\mathbf{g}_{pp'} + \mathbf{m}_{pp'd}\lambda_{pp'k}\right); \quad \forall\{p, p', k | p' > p \wedge k \leq K_p\}, \tag{2.20}$$

while for the $K_{p'}$ vertices $V_{p'k}$ of polygon p' the connection vectors read

$$\mathbf{p}_{p'pk} = \mathbf{x}_{p'k} - \left(\mathbf{g}_{pp'} + \mathbf{m}_{pp'd}\lambda_{p'pk}\right); \quad \forall\{p, p', k | p' > p \wedge k \leq K_{p'}\}. \tag{2.21}$$

Note that the auxiliary variables $\lambda_{pp'k}$ and $\lambda_{p'pk}$ are needed to compute the connection points $\mathbf{p}_{pp'k}$ and $\mathbf{p}_{p'pk}$. The two polygons are separated by the conditions of parallelism

$$\mathbf{p}_{pp'k} = \Delta_{pp'k}\mathbf{n}_{pp'}; \quad \forall\{p, p', k | p' > p \wedge k \leq K_p\}, \tag{2.22}$$

and anti-parallelism

$$\mathbf{p}_{p'pk} = -\Delta_{p'pk}\mathbf{n}_{pp'}; \quad \forall\{p, p', k | p' > p \wedge k \leq K_{p'}\}, \tag{2.23}$$

where the scalar variables $\Delta_{pp'k}$ and $\Delta_{p'pk}$ measure the distances of the vertices to the separating line.

To enforce that polygons do not overlap with circles, in (2.18)–(2.23) we replace polygon p' by circle i and make the following changes. Variables $\Delta_{p'pk}$ in (2.23) are fixed to radius

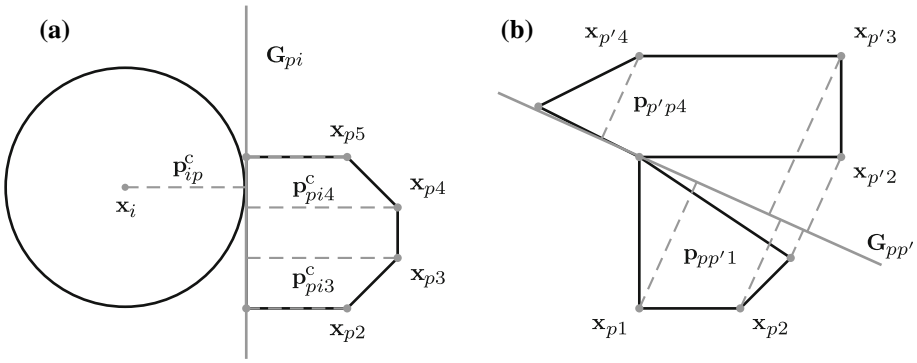

Fig. 2 Non-overlap for polygons and circles

R_i, i.e., the separation line is a tangent to the circle. $\Delta^{\mathrm{c}}_{pik}$ is the replacement of $\Delta_{pp'k}$ in (2.22) which now takes the form

$$\mathbf{p}^{\mathrm{c}}_{pik} = \Delta^{\mathrm{c}}_{pik}\mathbf{n}_{pi}; \quad \forall\{p, i, k | k \leq K_p\}, \tag{2.24}$$

while (2.23) reads

$$\mathbf{p}^{\mathrm{c}}_{ip} = -R(i)\mathbf{n}_{ip}; \quad \forall\{i, p\}. \tag{2.25}$$

Let us conclude this section by two remarks. The idea of the separating lines could be generalized and exploited by a dynamic cutting plane approach, in which the separating lines are added dynamically to ensure that objects do not overlap. The treatment of polygons presented also works for nonconvex polygons. However, in that case it is too restrictive and we may miss the global optimum.

2.5 Symmetry breaking

Symmetry degeneracy for cutting several objects from one design or stocked rectangle can be broken or at least reduced by requesting that the center of *one* of the objects is placed into the first quadrant of the design or stocked rectangle. If we select a specific circle i_* this reads

$$x_{i_*d} \leq \frac{1}{2}x^{\mathrm{P}}_d; \quad \forall\{d\}. \tag{2.26}$$

If we select a polygon, the inequality

$$x^0_{p_*d} \leq \frac{1}{2}x^{\mathrm{P}}_d; \quad \forall\{d\} \tag{2.27}$$

has a great effect.

Symmetry degeneracy due to the presence of congruent objects has been broken by sorting their center points with respect to the *lower left corner* of the design or stocked rectangle. Objects in the same congruence class are given the same congruence value I^{co}. For instance, for circles i and i' in the same congruence class we apply the ordering inequalities

$$x_{i1} + 5x_{i2} \leq x_{i'1} + 5x_{i'2}; \quad \forall\{(i, i') | i < i' \wedge I^{\mathrm{co}}_i = I^{\mathrm{co}}_{i'}\}. \tag{2.28}$$

Rather a matter of degeneracy than that of symmetry we briefly address the problem of free objects. Free objects are objects which can be moved locally without changing the objective function, i.e., the area of the design rectangle at all. Examples are shown in Fig. 4; the circle in the middle of Fig. 4c does not touch any other object. In a cutting problem free objects are not a problem except for degeneracy; in a packing problem this would cause severe problems as they would flow around freely. One avoid free objects by adding a soft penalty term which moves the center coordinate always towards the lower left corner of the design rectangle.

2.6 Structural analysis

The cutting problem has been formulated as an NLP problem with the following nonconvex aspects:

1. In the two-dimensional case their is the bilinear objective function (2.1). If either the length or the width of the design rectangles are fixed, it reduces to a linear objective function.

2. The overlap constraints lead to a geometrical situation with an obviously nonconvex domain. Imagine the rectangle from which to cut the objects and assume that we have one fixed object i_f. The feasible area for a set $\{i_1, i_2, \ldots, i_n\}$ of n objects with respect to i_f is a rectangle without the region covered by i_f. If all objects are circles the critical inequality is (2.4) with quadratic and bilinear terms. The relevant center-coordinate variables are only weakly bounded by the size of the design rectangle itself.
3. Cutting rectangles involves the following nonconvex features: the orthogonality equality (2.9) with bilinear terms, and the normalization equality (2.11) with a sum of quadratic terms. The variables $\mathbf{v}_{jk}$ involved in the critical terms are bounded by $-\max(S_{j1}, S_{j2})$ and $+\max(S_{j1}, S_{j2})$; these bounds grow with with the larger side of the rectangles to be cut.
4. Cutting polygons involves the following nonconvex features: the normalization equality (2.18) with a sum of quadratic terms, and the inequalities (2.20) and (2.21) with bilinear terms. The variables $\mathbf{m}_{pp'}$ involved in the critical terms are bounded by -1 and $+1$. The auxiliary variables λ are only weakly bounded by the size of the diagonal of the design rectangle.

As all nonconvex terms present in the model are bilinear or quadratic terms, algorithms specialized on such nonconvexities might be superior to the general purpose algorithms and software packages we used. The review by Floudas et al. [9] is a good resource for further references and a description of various approaches; we avoid repeating the material here but list a few of the relevant references among them Androulakis et al. [3], Maranas and Floudas [20], Adjiman et al. [2], and Adjiman et al. [1].

Especially, as we will see in Sect. 4.4, for cutting more than one polygon the gap is not closed. Thus, one might want to resort to relaxations to derive better lower bounds. Our approach in Sect. 2.7 is to replace the polygons by simpler objects (in our case) circles. Another approach is to use algebraic reformulations and convex relaxation techniques as described in Liberti [16], and Liberti and Pantelides [17].

2.7 Deriving lower and upper bounds

To reduce the range of variables we compute an upper bound on the area size of the design rectangle by replacing each polygon by its outer circles, i.e., by the smallest circles enclosing the polygons; see Appendix A.1. We then compute the optimal design rectangle and its area, a^+, which gives an upper bound to the optimal area, a, of the original problem. However, it is usually not difficult to find a solution to the original polygon cutting problem during the presolving phase. Therefore, this upper bound, a^+, is only used to reduce the range of the variable a.

However, it is very important to compute a tight lower bound. One might be attempted to compute the lower bound, a^-, by maximizing the radius r_p subject to

$$\left(\mathbf{x}_{pk} - \mathbf{x}_p^0\right)^2 \geq r_p^2; \quad \forall\{p, k | k \leq K_p\}. \tag{2.29}$$

This gives us the largest circle with all vertices *outside of*, or *on the circumference* of the circle. However, as this circle partially exceeds the polygons, solving the auxiliary problem in which the polygons are replaced by those circles, would only lead an estimation of the area, a, but it does not provide a lower bound. Another estimation of a could be derived from

the area, A_p, of the polygon computed by the Gaussian trapezian formula

$$A_p = \frac{1}{2}\left|\sum_{k=1}^{K_p}\left(X_{pk1} + X_{p,k+1,1}\right)\left(X_{pk2} - X_{p,k+1,2}\right)\right|, \tag{2.30}$$

where $X_{p,K_p+1,1}$ is replaced by $X_{p,1,1}$. This would enable us to replace the polygon by its equivalent-area circle with radius $R = \frac{1}{\pi}\sqrt{A_p}$.

A strict way to compute a^- is to derive the maximal inner circle with radius R_p of each polygon p as described in Appendix A.2, replace the polygons by those inner circles, and to compute the area-minimizing plate hosting all inner circles and original circles.

For all polygons replaced with their maximal inner circles plus all original circles, we then compute the optimal design rectangle and its area, a_0^-, which gives a lower bound $a_1^- := a_0^- - \Delta a$ to the optimal area, a, of the original problem; Δa is the absolute gap when solving the auxiliary problem. We can further improve the lower bound if we compute the following auxiliary quantities: the area access ΔA,

$$\Delta A := \sum_p \left(A_p - \pi R_p^2\right) \tag{2.31}$$

and the trimloss, z_0^-, associated with a_0^-

$$z_0^- := a_0^- - \sum_i \pi R_i^2 - \sum_p \pi R_p^2, \tag{2.32}$$

where we have reduced a_0^- by the area of all original circles i and all inner circles corresponding to the polygons. The lower bound, a_1^-, is replaced by

$$a_2^- = a_1^- + \max(0, \Delta A - z_0^-). \tag{2.33}$$

Unfortunately, this bound is only effective if the inner circles are poor approximations to the polygons. If the inner circles are good approximations to the polygons, ΔA is usually smaller than the circular trimloss z_0^-. In that case, however, we would expect that the minimal design rectangular is very similar to that one obtained when replacing the polygons by the circles.

Let us briefly illustrate this approach by example *c1p6a* in Table 5. The original area of the design rectangle is $a = 20.372$, and the gap is $\Delta z = z = 3.849$, i.e., the lower bound on waste or area, 16.523, has not moved at all. The next computational step is to replace all polygons by their maximal inner circles (object relaxation); the area of all circles is 14.428, the area access is $\Delta A = 2.096$. The object-relaxed problem gives a design rectangle of area size $a_0^- = 18.748$; as the circular problem was solved up to a gap of $1.88 \cdot 10^{-8}$ the lower bound on the area is reduced to $a_1^- = 18.748$. The trimloss of the object-relaxed solution is $z_0^- = 4.319$. As $\Delta A < z_0^-$ the lower bound on a_1^- cannot be further reduced; thus $a_2^- = a_1^- = 18.748$. Therefore, we can summarize the results as

	Original			Relaxed	Improved		
Quantity	$(a; z)$	LB	Gap	$(a_1^-; z_1^-)$	$(a_2^-; z_2^-)$	Gap	Relative gap (%)
Area	20.372	16.523	3.849	18.748	18.748	1.624	8.66
Waste	3.849	0.000	3.849	2.225	2.225	1.624	72.99

where LB stands for lower bound obtained for the original problem; the quantities z denote waste and corresponds to the area quantities a. Note that the original absolute gap has been reduced from 3.849 to 1.624, i.e., by about 58% leading to a relative area gap of 8.66%.

3 Modeling: cutting and object allocation to several rectangles

If the objects do not fit into one rectangle, several rectangles need to be designed and produced. However, cutting from several rectangles leads to a nonconvex MINLP problem because in addition to all variables and constraints described in Sect. 2 we need binary variables δ_{ir} to decide on the allocation of object i to design rectangle or stocked rectangle r. Note that in this section we cover both cases: defining new design rectangles subject to specified lower and upper bounds as well as assigning the objects to a set of stocked rectangles with given dimensions. We expect this formulation to work efficiently only for a small number of rectangles r. Problems with a large number of stocked rectangles can be solved by the column enumeration approach described in Rebennack et al. [21].

The assignment constraints are given by the requirement that each objects has to be allocated, i.e.,

$$\sum_r \delta_{ir} = 1; \quad \forall\{i\}. \tag{3.34}$$

Assignment to rectangle r is only possible if this rectangle is used at all. We indicate the usage of rectangle r by the binary variable σ_r which is coupled to the assignment variables by

$$\sum_i \delta_{ir} \geq \sigma_r; \quad \forall\{r\}, \tag{3.35}$$

and

$$\delta_{ir} \leq \sigma_r; \quad \forall\{i, r\}. \tag{3.36}$$

The inequality (3.36) ensures that objects i can only be allocated to resource rectangle r if r is used, while (3.35) enforces that at least one object i is assigned to r if r is used. For design rectangles we add the symmetry breaking constraints

$$\sigma_r \leq \sigma_{r-1}; \quad \forall\{r | r > 1\}. \tag{3.37}$$

To avoid the complicating issues to incorporate δ_{ir} in the non-overlap constraints we use the following equivalent approach. We arrange all resource rectangles (design, or stocked) in a chain of rectangles in which each rectangle r is a segment with width $w_r = x^{\mathrm{DR}}_{r1}$ and length $\ell_r = x^{\mathrm{DR}}_{r2}$ subject to lower and upper bounds 0 and X^{SR}_{rd}. Note that this approach is independent of the sequence. We use this chain of rectangle only to illustrate the ideas and to construct the following constraints. The variables x^{DR}_{rd} are coupled to σ_r by

$$x^{\mathrm{DR}}_{rd} \leq X^{\mathrm{SR}}_{rd}\sigma_r; \quad \forall\{r, d\}. \tag{3.38}$$

If we consider stocked rectangles the variables x^{DR}_{rd} are fixed to the given dimensions X^{SR}_{rd} of resource plate r, if r is used at all, i.e.,

$$x^{\mathrm{DR}}_{rd} = X^{\mathrm{SR}}_{rd}\sigma_r; \quad \forall\{r, d\}. \tag{3.39}$$

In absolute coordinates, rectangle or segment r starts at length ℓ^0_r and occupies the length coordinate up to $\ell^0_r + x^{\mathrm{DR}}_{r2}$. This segment approach guarantees that objects allocated to different rectangles automatically do not overlap. The only constraints we have to modify are those constraints or bounds related to the lower and upper bounds of the center of circles, i.e., (2.5) and (2.6), or vertices of the rectangles (2.7) and polygons (2.16). We illustrate this approach

and the necessary modifications for circles only as the application to the other objects is obvious. Note that our formulation addresses the full assignment problem while George et al. [11] allocated circles to rectangles by inspecting the rectangles *separately*; they were well aware of the limitation of their approach.

The width coordinate, x_{i1}, of the center of circle i, is now restricted by

$$x_{i1} \leq x_{r1}^{\mathrm{DR}} + X_{r1}^{\mathrm{SR}}(1-\delta_{ir}) - R_i\delta_{ir}; \quad \forall\{i,r\}. \tag{3.40}$$

Note that for $\delta_{ir} = 0$ (3.40) becomes redundant, i.e.,

$$x_{i1} \leq x_{r1}^{\mathrm{DR}} + X_{r1}^{\mathrm{SR}}; \quad \forall\{i\}, \tag{3.41}$$

while for $\delta_{ir} = 1$ (3.40) leads to

$$x_{i1} \leq x_{r1}^{\mathrm{DR}} - R_i; \quad \forall\{i\}, \tag{3.42}$$

as wanted. Note that $\delta_{ir} = 1$ is only possible if $x_{r1}^{\mathrm{DR}} \geq 2R_i$ as otherwise the circle does not fit into assignment rectangle r at all.

The length coordinate, x_{i2}, is subject to a lower and upper limit in order to fit into a specific segment. The lower limit is established by

$$x_{i2} \geq \ell_{ir}^{\delta} - L^{+}(1-\delta_{ir}) + R_i; \quad \forall\{i\}, \quad L^{+} = \sum_{r\in\mathcal{R}_*} X_{r2}^{\mathrm{SR}} \tag{3.43}$$

where $\ell_{ir}^{\delta} := \ell_r^0\delta_{ir}$ and L^+ is the sum of the lengths of the design or assignment rectangles (except for the last one) and serves to make (3.43) redundant. An upper limit on x_{i2} is given by

$$x_{i2} \leq \ell_{ir}^{\delta} + x_{r2}^{\mathrm{DR}} - L^{+}(1-\delta_{ir}) - R_i\delta_{ir}; \quad \forall\{i\}, \tag{3.44}$$

which is similar to (3.40) except for the absence of the bilinear term $\ell_r^{\delta} := \ell_r^0\delta_{ir}$ term established by

$$\ell_{ir}^{\delta} \leq \ell_r^0; \quad \forall\{i,r\}, \tag{3.45}$$

$$\ell_{ir}^{\delta} \leq L_r^0\delta_{ir}; \quad \forall\{i,r\}, \tag{3.46}$$

with upper bound $L_r^0 = \sum_{m=1}^{r-1} X_{r2}^{\mathrm{SR}}$ on ℓ_r^0 with $L_1^0 = 0$, and

$$\ell_{ir}^{\delta} \geq \ell_{ir}^{\delta} - L_r^0\delta_{ir}; \quad \forall\{i,r\}. \tag{3.47}$$

Note that $R_i\delta_{ir}$ in (3.44) avoids that object i is assigned to a rectangle r if it does not fit into it.

For design rectangles we add the symmetry breaking constraints

$$x_{r1}^{\mathrm{DR}} \leq x_{r-1,1}^{\mathrm{DR}}; \quad \forall\{r|r>1\}, \tag{3.48}$$

i.e., the design rectangles are constructed according to decreasing width. Another symmetry breaking constraint is the requirement that the width of the design rectangles does not exceed its length, i.e.,

$$x_{r1}^{\mathrm{DR}} \leq x_{r2}^{\mathrm{DR}}; \quad \forall\{r\}. \tag{3.49}$$

Finally, if we consider design rectangles, we assign the object with the largest area to the first design rectangle.

Note that the formulation presented has the following advantage. The binary variables never show up in the non-overlap or shape constraints. They only connect to the variables representing the center of the circles or the vertices of the polygons.

4 Numerical experiments and results

We present a case study in which we apply the solution approach to solve problems of a modest number of objects. We consider up to 10 objects to be nested into one design rectangle. We consider examples with only circles in Sects. 4.1 and 4.2; with circles *and* rectangles in Sect. 4.3; and with circles *and* polygons in Sect. 4.4. We also distinguish experiments with differently shaped objects to be cut and those which have mostly congruent figures. All experiments have been performed using the Branch&Reduce Optimization Navigator (BARON) exploiting global optimization techniques; cf. Ghildyal and Sahinidis [12] or Tawarmalani and Sahinidis [28]. For some of them we also tried LindoGlobal which is part of the GAMS 22.5 distribution. In the tables displayed in the next sections we use the following symbols:

a	Optimal area of the design rectangle
A_{circ}	The area occupied by the circles to be cut
A_{rect}	The area occupied by the rectangles to be cut
CPU	The CPU time in seconds
Δ	The absolute gap; sometimes we display a multiple of the gap
L; ℓ	Upper bound and optimal length of the design rectangle
n	The number of circles
N_{row}	The number of constraints
N_{col}	The number of variables
N_{nz}	The number of non-zero coefficients in the problem matrix
N_{nlin}	GAMS code length providing a measure for the complexity of the nonlinear terms, *e.g.*, e^{xy} is, loosely speaking, more nonlinear than xy
N_{nlz}	The number of nonlinear matrix entries in the model
N_{iter}	The number of BARON iteration
N_{best}	The node at which BARON found the optimal solution
N_{mem}	The maximum number of nodes hold in memory
W; w	Upper bound and optimal width of the design rectangle
z	Minimal waste of the design rectangle

4.1 Sets of mostly congruent circles

In this numerical experiments summarized in Table 1 we used n circles of radius $R = 0.5$, the cases indicated by a, b, and c contain a few larger circles (a: one circle with radius $R = 0.7$, b: one circle with $R = 0.9$, and c: one circle with $R = 0.7$ and another one with $R = 0.9$), e.g., case *c6-b* involves 6 circles of radius $R = 0.5$ and one circle of radius $R = 0.9$. Case *c6a-x* involves 6 circles of radius $R = 0.725$ and one circle of radius $R = 1.2$; it was solved to provide a lower bound on the polygon case *c1p6*.

In this tables and others which contains only circles to be cut, the waste is given by $z = a - \sum_i \pi R_i^2$, and Δ is the absolute gap between the upper and lower bound of the objective function z. If the upper limits, W and L, on the size of the design rectangle are not active, and the x coordinate measures the width, a linear chain of congruent circles with radii $R_i = R$ and center coordinates $(x_i, y_i) = R(1, 2i - 1)$ as displayed in Fig. 3a is globally

Table 1 Congruent circles

n case	3 “c3-1”	3 “c3-2”	4 “c4-1”	4 ”c4-2”	5 “c5-1”	5 “c5-2”	6 “c6-1”	6 “c6-2”	6 “c6-3”	7 “c7-1”	7 “c7-2”
N_{row}	15	15	23	23	39	34	46	55	46	60	60
N_{col}	10	10	12	12	14	14	16	16	16	18	18
N_{nz}	42	42	70	70	130	108	152	190	152	204	204
N_{nlin}	138	138	264	264	582	472	702	892	702	453	453
N_{nlz}	26	26	50	50	106	82	122	162	122	86	86
W; w	4; 1.00	2; 1.87	1; 1.00	2; 2.00	4; 1.00	2; 2.00	4 ; 2.00	2; 2.00	1.9; 1.00	4.0; 1.00	2.0; 2.00
L; ℓ	8; 3.00	2.5; 2.00	4; 4.00	4; 2.00	8; 5.00	4; 2.73	8; 3.00	4; 3.00	8.0; 6.00	8.0; 7.00	6.0; 3.73
a	3.0000	3.7320	4.0000	4.0000	5.0000	5.4641	6.0000	6.0000	6.0000	7.00	7.4641014
z	0.6438	1.3758	0.8584	0.8584	1.0730	1.5371	1.2876	1.2876	1.2876	1.5022084	1.9663143
CPU	1	3	4	40	459	82	477	1107	82	1325	442
N_{iter}	21	455	141	6335	26407	6561	27733	41841	5927	46403	18348
N_{best}	21	424	141	228	774	6338	1488	38224	2830	23952	17754
N_{mem}	4	33	6	105	468	299	563	724	135	887	356
$10^9\Delta$	1.00	1.38	1.00	1.00	1.07	1.54	1.28	1.54	1.28	1.50	1.97

The first line of the column gives the number, n, of circles and the name assigned to that case. For an explanation of the other symbols see the table at the beginning of Sect. 4

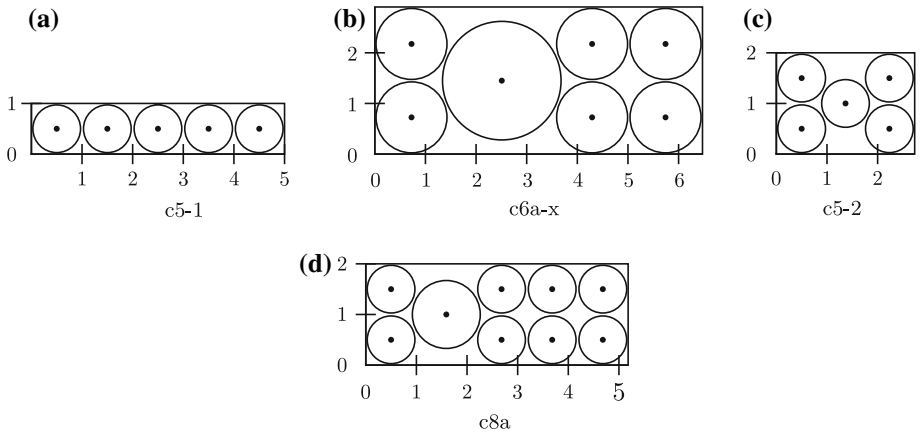

Fig. 3 Mostly congruent circles. The circles in (**a**), and (**c**) have unit diameter. The length of the design rectangle is visible in the horizontal, its width in the vertical extension

optimal. If a linear chain cannot be established because $2nR > L$, solutions with $w = W$ and $\ell < L$ are obtained; see Fig. 3c.

4.2 Several circles of mostly different size

In this test series summarized in Table 2 and displayed in Fig. 4 we consider up to 10 circles of different size for a typical 4×8 design plate, and a larger one with $L = 18$. The cases with more than 5 circles contain one pair of congruent circles of radius 0.6.

These cases can be solved relatively easily by BARON. However, as cases *5a* and *5b* illustrate, the maximum width, W, of the design rectangle has a strong influence on the solution time. The smaller W, the easier to solve a case. Case 10 has been solved exploiting the symmetry breaking constraint (2.26).

4.3 Circles and rectangles

For small cases, the rectangles are modeled as polygons with center $\mathbf{x}_q^0$ rotated by α with $0° \le \alpha \le 180°$. Even for these small cases, it became important and necessary to specify priorities on the branching variables; high priority (5000 as BARON option) for $\mathbf{x}_q^0$ and the variable $\cos\alpha$ is prioritized with the BARON value 1000. Case *c1r1-3* has two rows more than *c1r1-1* and *c1r1-2* because we included the symmetry breaking inequalities (2.26).

The larger cases displayed in Fig. 5 with 6 circles and 1, 2, or 3, resp., rectangles have been solved to optimality during the preprocessing phase using the model formulation presented in Sect. 2.4.2; for rectangles this model is more efficient than the general polygon case. In this case the area, A_{circ}, covered by the circles is 22.84 while the rectangles cover only 1.52, i.e., the rectangles can easily placed in the empty space between the circles (Table 4).

4.4 Circles and polygons

The circle-polygon experiments are summarized in Table 5; some of them are displayed in Fig. 6. In the examples *c1p1-1*, *c1p1-2*, and *c1p1-3* the polygons were rectangles of size 0.1×0.15, 0.5×0.75, and 0.5×0.75, respectively. In the other cases, the coordinates of

Table 2 One or two larger circles added to a set of congruent circles

n case	7 "c6a-x"	7 "c6-a"	7 "c6-b"	7 "c6-c"	7 "c7-a"	8 "c8-a"
N_{row}	56	54	54	65	71	88
N_{col}	18	18	18	20	20	22
N_{nz}	184	180	180	216	240	304
N_{nlin}	453	453	453	600	600	768
N_{nlz}	86	86	86	114	114	146
W; w	4; 2.90	2.9; 2.00	2.9; 2.00	4.0; 3.00	1.9; 1.86602540	2.1; 2.00
L; ℓ	8; 6.47	8.0; 4.18	8.0; 4.62	8.0; 6.87	8.0; 5.19820347	8.0; 5.18174243
a	18.7529	8.363485	9.2306787	20.62142	9.69997972	10.36348470
z	4.32118	2.111715	1.9735997	4.097526	2.6628121754	2.5409189963
CPU	832	2113	3077	44926	112	34324
N_{iter}	27020	70431	107635	959291	1691	640509
N_{best}	26679	59632	67278	925960	1645	639313
N_{mem}	880	2956	2960	30344	170	19800
$10^9\Delta$	4.32	2.11	1.97	2.97	2.66	2.55

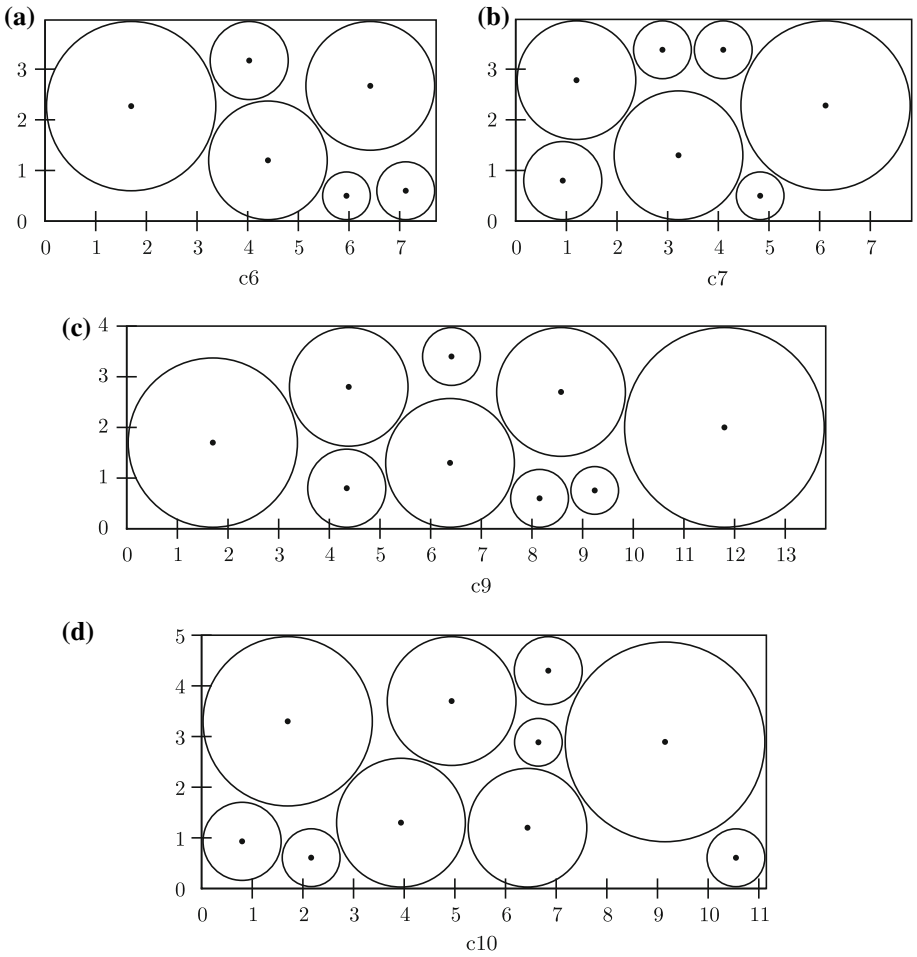

Fig. 4 Several circles of different size. The four examples displayed in this figure are taken from Table 3

the polygon vertices were

Case	Polygons and their vertices
c1p1-1	$1 \times 0.05(0, 0; 0, 2; 3, 2; 3, 0)$
c1p1-2	$1 \times 0.25(0, 0; 0, 2; 3, 2; 3, 0)$
c1p1-3	$1 \times 0.25(0, 0; 0, 2; 3, 2; 3, 0)$
c1p5a	$2 \times \frac{1}{2}(0, 0; 0, 2; 1, 3; 2, 3; 3, 2; 3, 0)$
c1p5b	$5 \times \frac{1}{2}(0, 0, 0, 2, 1, 3, 2, 3, 3, 2; 3, 0)$
c1p6a	$6 \times \frac{1}{2}(0, 0; 0, 2; 1, 3; 2, 3; 3, 2; 3, 0)$
c3p3	$2 \times \frac{1}{2}(0, 0, 0, 2, 1, 3, 2, 3, 3, 3, 0) + 1 \times \frac{1}{2}(0, 0, 0, 2, 3, 2, 3, 0)$
c6p3	$1 \times 0.1(0, 0; 0, 8; 5, 8; 5, 0) + 1 \times (0, 0; 0, 1; 1, 1; 1, 0)$ $+1 \times 0.1(0, 0; 0, 3; 4, 3; 4, 0)$

Let us illustrate the interpretation of this table by case *c1p5a* with 6 vertices (0,0), (0,1), (0.5,1.5), (1,1.5), (1.5,1), and (1.5,0). The small gap of $9.05 \cdot 10^{-6}$ in case *c3p1b* could only

Table 3 Several circles of different size

n	5a	5b	6	7	8	9	10
N_{row}	24	24	45	39	48	58	77
N_{col}	14	14	16	18	20	22	24
N_{nz}	68	68	150	120	152	188	256
N_{nlin}	222	222	642	453	600	768	957
N_{nlz}	42	42	122	86	114	146	182
$10R_i$	12,6,8,17,5	12,6,8,17,5	"5a" + 13	"6" + 6	"6" + 20,13	"8" + 6	"9" + 7
W; w	4; 3.97	5; 4.93	4; 3.97	4; 3.98	4; 4.00	4; 4.00	4; 4.00
L; ℓ	8; 7.72	18; 5.76	8; 7.72	8; 7.82	18; 13.79	18; 13.80	15; 13.83
a	30.6272727	28.41130160	30.6273	31.12348131	55.19629959	55.19629958	55.32088961
z	8.57329222	6.35732118	7.78790	7.15312936	14.48125881	13.35028544	11.93549508
CPU	11	116	41	211	548	5288	1577
N_{iter}	163	11469	688	7221	18455	250286	13305
N_{best}	88	11408	674	7221	13912	87298	13208
N_{mem}	16	308	47	316	897	11611	720
$10^8\Delta$	0.858	0.635	0.778	0.716	1.45	1.34	1.19

For an explanation of the symbols see the table at the beginning of Sect. 4

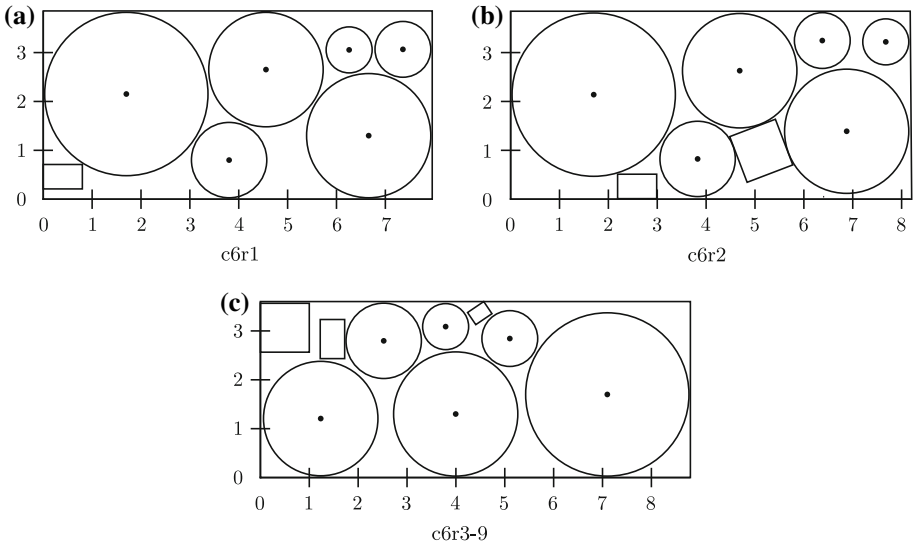

Fig. 5 Cutting circles and rectangles. Note that the configuration displayed in Subfigure (**c**) is not globally optimal. The smallest rectangle could be moved elsewhere giving more space for the small circle on the right of it allowing the largest circle to move to the left, and thus leading to a reduced length of the design plate

be reached when we used the symmetry breaking inequality (2.27). Case *c6p3* contains the same rectangles as *c6r3-9*, but represents them as general polygons. Table 5 shows that only small and moderate cases with less than 200 nonlinear non-zero coefficients can be solved to small gaps. For all other experiments the solutions listed have been solved during preprocessing. However, during the Branch&Reduce phase, no further solutions were found nor the lower bound was increased. In case *c1p6a* we applied the bound improving approach described in Sect. 2.7. The two-polygon problem displayed in Fig. 6c was solved to global optimality using `LindoGlobal` within 2 min.

4.5 Cutting and allocating objects to several rectangles

Here we consider numerical experiments for constructing simultaneously several design rectangles (Fig. 7) and assigning objects to stocked rectangles (Fig. 8). Using the formulation described in Sect. 3 we produced feasible solutions with in 1 or 2 min. But in none of these cases were we able to find solutions with gaps smaller than 10^{-7}.

4.6 Summarizing the experiments

In all circular experiments we found good feasible solutions within seconds or the latest in minutes. Cutting only circles is easiest which is not a surprise as the numbers of variables and constraints are small. Solution with gaps between the upper and lower bound of the objective function of the order of 10^{-8} are within seconds and minutes; these seem to be good approximations to the global optimum. Cutting congruent circles requires to use the symmetry breaking constraints (2.28). Achieving small gaps is difficult when several polygons are involved, especially, when several larger ones need to be cut. Addressing the community of *reliable computing* and *interval arithmetic*, we point out that all our statements about small

Table 4 Circles and rectangles

n	*c1r1-1*	*c1r1-2*	*c1r1-3*	*c6r1*	*c6r2-9*	*c6r3-9*
N_{row}	54	54	56	150	499	388
N_{col}	47	47	47	56	250	136
N_{nz}	168	168	172	500	1,730	1,316
N_{nlin}	218	218	218	3,048	9,531	8,490
N_{nlz}	42	42	42	358	1148	950
$10R_i$	5	5	5	12,6,8,17,13,5	12,6,8,17,13,5	12,6,8,17,13,5
$10S_j$	1,1.5	5,7.5	10,15	8,5	8,5; 10,10	8,5; 10,10; 3,4
A_{circ}	$\pi/4$	$\pi/4$	$\pi/4$	22.84	22.84	22.84
A_{rect}	0.015	0.375	1.50	0.40	1.40	1.52
$W; w$	4; 1.00	4; 1.00	2.5; 1.00	4; 3.85243696	4; 3.85	4; 3.60
$L; \ell$	8; 1.00	8; 1.50	3.0; 2.50	9; 7.95916840	9; 8.18	9; 8.80
a	1.0000	1.5000	2.5000	30.66219452	31.4492	31.6813
z	0.19960183675	0.3396	0.214601528206	7.42271593339	7.20982762724	7.32195479793
CPU	1	42	368	15	60	309
N_{iter}	-1	6,269	6,8109	209	-1	-1
N_{best}	-1	6,234	27,003	-1	-1	-1
N_{mem}	0	70	233	15	0	0
Δ	0	10^{-9}	10^{-9}	10^{-4}	0	0

Table 5 Circles and Polygons; some cases are displayed in Fig. 6

n	*c1p1-1*	*c1p1-2*	*c1p1-3*	*c1p5a*	*c1p5b*	*c1p6a*	*c3p1b*	*c3p3*	*c6p3*
$10R_i$	5	5	5	12	12	12	12,6,8	12,6,8	12,6,8,17,13,5
N_{row}	54	54	56	799	838	1161	141	516	661
N_{col}	47	47	47	767	791	1,110	125	489	628
N_{nz}	168	168	172	2,914	3,076	4,290	484	1,836	2,294
N_{nlin}	218	218	218	6,252	6,492	9,303	907	3,829	4,863
N_{nlz}	42	42	42	1,214	1,262	1,808	178	748	950
W; w	4; 1.00	4; 1.00	2.50; 1.00	4; 2.50	4; 3.90	4; 3.00	4; 2.68	1; 3.40	4; 3.97167407
L; ℓ	8; 1.00	8; 1.50	3; 2.50	8; 6.36	8; 4.76	8; 6.79	8; 4.95	4; 4.93	8; 4.95
a	1.0000	1.5000	2.50	15.90497802	18.55292041	20.37285414	13.29508724	16.75621374	35.10392485
z	0.19960183675	0.3396	0.214601528206	2.88108459665	4.02902699371	3.84896072355	3.62960116801	3.59072766	10.7445462623
CPU	0.1	42	368	2,000	1,800	1,800	53	2,000	33,000
N_{iter}	−1	6,269	68,109	−1	241	14	1	472	11,755
N_{best}	−1	6,234	27,003	−1	−1	−1	1	−1	−1
N_{mem}	0	70	233	0	123	13	1	195	1,415
Δ	0	$1.00 \cdot 10^{-9}$	$1.00 \cdot 10^{-9}$	2.88108459665	4.02902699371	3.84896072355	$9.05 \cdot 10^{-6}$	3.59072766	10.7445462623
CPU						3,774			
a_1^-						18.7477203821			
Δa						$1.88 \cdot 10^{-8}$			
A_{circ}						14.42828163			
z_0^-						4.3194387521			
ΔA						2.096			
a_2^-						18.7477203821			
Δ_2						1.6251337579			

Note that case *c6p3* places the same objects as case *c6r3*. However, numerically it is less efficient to treat rectangles as general polygons. Column *c1p6a* demonstrates the effect of exploiting the lower bound obtained by solving the auxiliary circle problem described in Sect. 2.7; the original gap of 3.85 is reduced to 1.62, which corresponds to a reduction of more than 50% and a relative area gap of 8.7 percent. More details are given at the end of Sect. 2.7. a_1^- is the lower bound on the area of the design rectangle obtained by solving the problem with all polygons replaced by their maximum size inner circle; Δa is the absolute gap obtained for this relaxed problem. A_{circ} denotes the area occupied by the original circles. z_0^- is the trimloss associated with the solution of the relaxed problem, ΔA is the area access of polygons minus their inner circles, a_2^- and Δ_2 are the lower bound and the absolute gap after evaluating the area access and relaxed trimloss

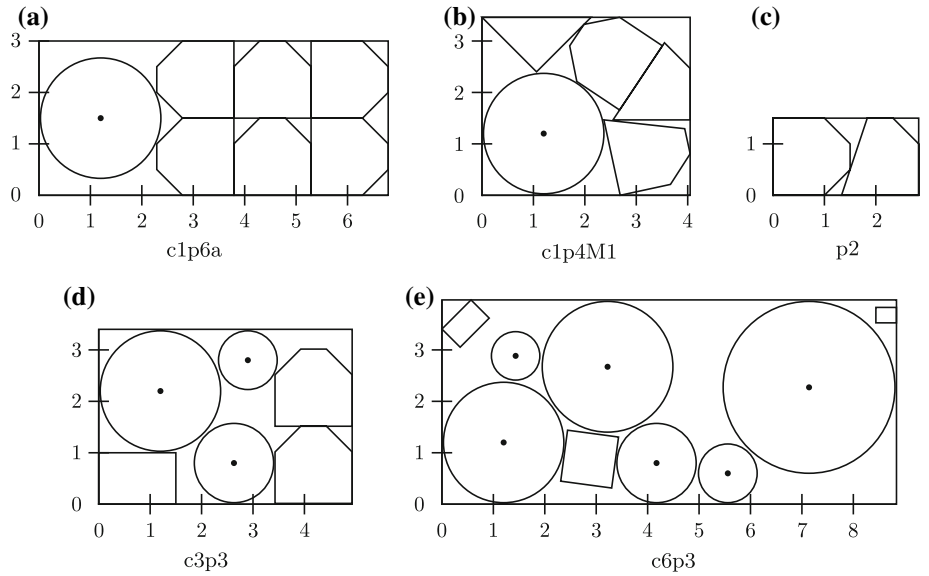

Fig. 6 Circles and convex polygons. Although all solutions have been produced in short time, in most cases we were not able to find solutions with gaps smaller than 10^{-7}. The configurations displayed in Subfigures (**a**) and (**e**) are obviously not the global optimum. In (**a**) one could move the two right polygons left to the circle; in case (**e**) the small circle at the lower left of the big circle could be placed at the position of the small rectangle at the upper left corner of the sheet; the big circle could be moved towards left possibly adjacent to the other two circles, and the small rectangle previously at the upper left corner of the sheet could be placed at the lower right corner of the sheet below the displaced great circle — this would decrease the length of the sheet while keeping the width the same. Only the two-polygon configuration displayed in Subfigure (**c**) was proven to be the global optimum

gaps and global optima are subject to the limits related to the fact that BARON deals with finite number arithmetic subject to round-off errors. Shortly before submitting this paper, LindoGlobal became available in the most recent GAMS 22.5 release. On problems with a small number of circles, it produced quickly solutions with gaps of the order of 10^{-11}. BARON produced smaller for cases with more circles.

In the polygon experiments with more than one polygon we found feasible solutions within seconds or the latest in minutes but the relative gaps were 100% with the lower bound not moving away from zero. An interesting case is the two-polygon problem displayed in Fig. 6c; here LindoGlobal proved global optimality within 40 min, a case on which BARON did not increase the lower bound at all. For larger polygon cases we experienced similar problems as with BARON. Nevertheless, the overall experience with both commercial solvers is encouraging.

5 Conclusions

We have developed NLP and MINLP models describing the problem of cutting circles, rectangles and polygons from rectangular design or stocked plates, and applied several solution techniques to solve this problem among them the Branch&Reduce Optimization Navigator (BARON) called from GAMS, and, for solving the allocation problem, a column enumeration approach [Rebennack [21]] in which the columns represent feasible assignments. It is the first time when circles and arbitrary convex polygons are cut simultaneously. Good,

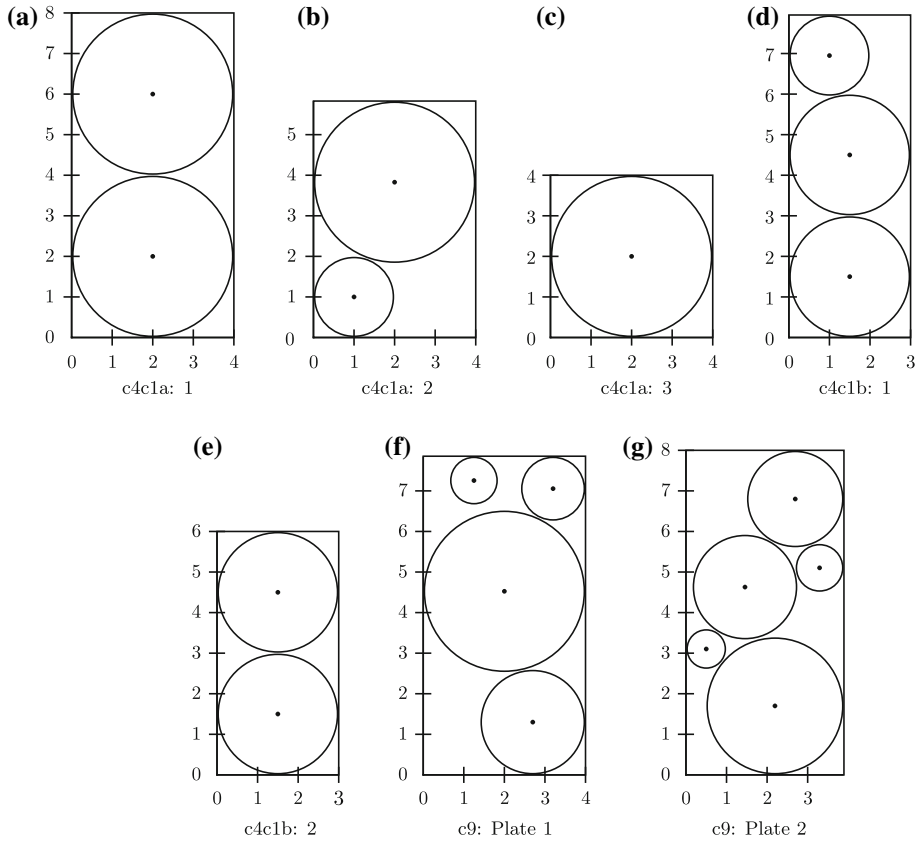

Fig. 7 Subfigures (**a**–**c**) show three design plates to be produced to cut four circles of radius R = 2, and one smaller circle with $R = 1$. Subfigures (**d**, **e**) show a similar case with four circles of radius $R = 1.5$. In the third case (**f**, **g**), two plates are generated to cut 9 circles

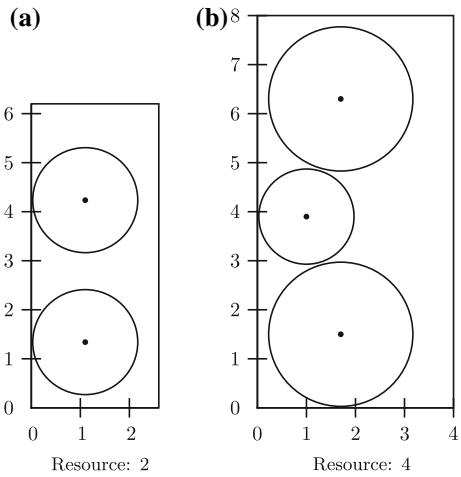

Fig. 8 5 circles are allocated to 2 of 5 stocked plates. Note that the circles are placed within the stocked plates but are not yet arranged optimally. This could be accomplished by solving an additional strip or bin packing problem

often near globally optimal solutions with gaps of the order of 10^{-8}, are computed within seconds or minutes usually during preprocessing. As it is expected from the NP-hard nature of the problem, we can derive such small gaps within seconds or minutes only for small cases. Symmetry degeneration is tackled by appropriate symmetry breaking constraints. Cases with small rectangles and polygons, which fit in the free space between the circles, are relatively easy to solve. Cases with large polygons and significant trimloss are much more difficult, and at best we have derived reasonable lower bounds by exploiting an auxiliary model using the maximum inner circle fitting into a polygon. The upper bound, W, on the width of the design rectangle plays an important role. The smaller W, the faster near globally optimal solutions with gaps smaller than 10^{-7} are reached.

The approach developed here serves real world applications in which one has to cut valuable material. In such cases, solutions proven to be globally optimal can be superior to solutions produced by heuristics. For cases with small number of objects to be cut, the computational effort meets the practical requirements. The reformulation–linearization technique (RLT) approach by Liberti [16], and more recently, Liberti and Pantelides [17] can extend the limit of problem sizes which can be solved with reasonable gaps. If cases with significantly more objects need to be solved, or a certain time limit must not be exceeded in a real world application one might resort to metaheuristics.

Acknowledgements Thanks is directed to Steffen Rebennack (University of Florida) for proof reading and supporting the production of the graphics. Christodoulos A. Floudas (Princeton University) improved this publication by pointing me to a set of publications of E. G. Birgin and co-workers. Comments by and discussions with Leo Liberti (LIX Ecole Polytechnique, F-91128 Palaiseau, France), Tapio Westerlund (Abo Adademi University, Turku, Finland), and Tibor Csendes (University of Szeged, Szeged, Hungary) are greatly acknowledged. Two unknowns referees made constructive and valuable suggestions and thus helped to improve this paper.

Appendix

A Auxiliary models

In this appendix we provide two auxiliary models used to derived upper and lower bounds on the area of the design rectangle. The basic idea is to replace the polygons by there smallest outer and largest inner circle.

A.1 Smallest circles enclosing the polygon

The model to compute the smallest circle enclosing polygon p is to minimize the radius r subject to the constraints that all vertices of the polygon are inside the circle defined by r_p and the center $\mathbf{x}^0$, i.e.,

$$\left(\mathbf{x}_{pk} - \mathbf{x}_p^0\right)^2 \leq r_p^2; \quad \forall\{p, k | k \leq K_p\}. \tag{A.50}$$

The global optimum of this problem is computed within seconds.

A.2 Largest circle fitting in the polygon

For a given polygon p, the radius of the maximal size circle is the maximal smallest height h_{pk} of all onto the edge D_{pk} of the triangles given by the sides d_{pk}, $d_{p,k+1}$ and D_{pk}, where

D_{pk} is the distance between vertex V_{pk} and vertex $V_{p,k+1}$,

$$D_{pk}^2 = \left(\mathbf{X}_{p,k+1} - \mathbf{X}_{pk}\right)^2, \tag{A.51}$$

and d_{pk} and $d_{p,k+1}$ are the distance of the vertices V_{pk} and vertex $V_{p,k+1}$ to the center $\mathbf{x}_p^0$ of the inner circle

$$d_{pk}^2 = \left(\mathbf{X}_{pk} - \mathbf{x}_p^0\right)^2, \quad d_{p,k+1}^2 = \left(\mathbf{X}_{p,k+1} - \mathbf{x}_p^0\right)^2; \quad \forall\{k|k \le K_p\}. \tag{A.52}$$

The objective function of the problem is to maximize h_p subject to

$$h_p \le h_{pk}^{\text{tri}}; \quad \forall\{p, k|k \le K_p\}. \tag{A.53}$$

The heights h_{pk}^{tri} depend on the center, $\mathbf{x}_p^0$, and the distances $d_{pk}, d_{p,k+1}$, and D_{pk}. They follow from Heron's area formula with a, b, and c being the lengths of the sides of the triangle

$$F_{abc} = \sqrt{s(s-a)(s-b)(s-c)}; \quad s = \frac{a+b+c}{2} \tag{A.54}$$

and

$$F_{abc} = \frac{ah_a}{2} = \frac{bh_b}{2} = \frac{ch_c}{2}. \tag{A.55}$$

With $a = d_{pk}$, $b = d_{p,k+1}$, and $c = D_{pk}$ we obtain

$$h_{pk}^{\text{tri}} = 2\frac{F}{D_{pk}} = 2\frac{\sqrt{s(s-d_{pk})(s-d_{p,k+1})(s-D_{pk})}}{D_{pk}}, \quad s_p = \frac{d_{pk} + d_{p,k+1} + D_{pk}}{2}. \tag{A.56}$$

Let h_p^* be the optimal solution to (A.53). The circle placed at $\mathbf{x}_p^0$ with radius $R_p = h_p^*$ is the inner circle of maximal size completely inside the polygon p; see Fig. 9. Under certain assumptions this circle touches all edges but we cannot count on this. For instance, if the polygon is a rectangle with different length and width, the inner circle touches only the two longer edges.

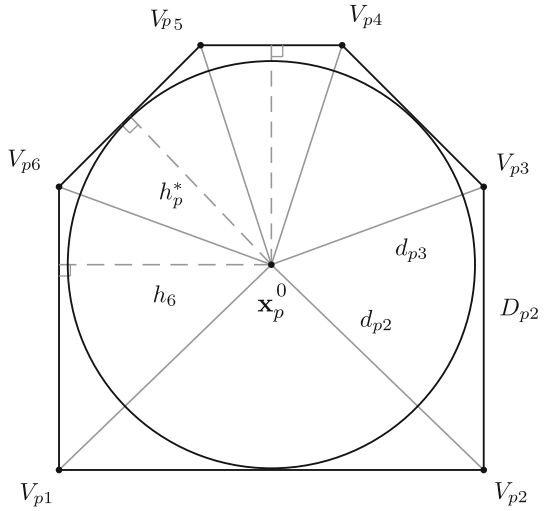

Fig. 9 Polygon p and its maximal inner circle with radius h_p^* and center $\mathbf{x}_p^0$. Note that the circle has three touching points with the polygon

References

1. Adjiman, C.S., Androulakis, I.P., Floudas, C.A.: Global optimization of mixed integer nonlinear problems. AIChE J. **46**, 1769–1797 (2000)
2. Adjiman, C.S., Androulakis, I.P., Maranas, C.D., Floudas, C.A.: A global optimization method aBB for process design. Comput. Chem. Eng. Suppl. **20**, S419–424 (1996)
3. Androulakis, I.P., Maranas, C.D., Floudas, C.A.: aBB: a global optimization method for general constrained nonconvex problems. J. Glob. Optim. **7**, 337–363 (1995)
4. Birgin, E.G., Martínez, J.M., Nishihara, F.H., Ronconi, D.P.: Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization. Comput. Oper. Res. **33**, 3535–3548 (2006)
5. Birgin, E.G., Sobral, F.N.C.: Minimizing the object dimensions in circle and sphere packing problems. Comput. Oper. Res. **34**, online 16. Jan 2007.
6. Brooke, A., Kendrick, D., Meeraus, A.: GAMS – A User's Guide (Release 2.25). Boyd & Fraser Publishing Company, Danvers, Massachusetts (1992)
7. Dowsland, K.A., Dowsland, W.B.: Packing problems. Eur. J. Oper. Res. **56**, 2–14 (1992)
8. Dyckhoff, H.: A typology of cutting and packing problems. Eur. J. Oper. Res. **44**, 145–159 (1990)
9. Floudas, C.A., Akrotiriankis, I.G., Caratzoulas, S., Meyer, C.A., Kallrath, J.: Global optimization in the 21st century: advances and challenges for problems with nonlinear dynamics. Comput. Chem. Eng. **29**, 1185–1202 (2005)
10. Fraser, H.J., George, J.A.: Integrated container loading software for pulp and paper industry. Eur. J. Oper. Res. **77**, 466–474 (1994)
11. George, J.A., George, J.M., Lamar, B.W.: Packing different-sized circles into a rectangular container. Eur. J. Oper. Res. **84**, 693–712 (1995)
12. Ghildyal, V., Sahinidis, N.V.: Solving global optimization problems with BARON. In: Migdalas, A., Pardalos, P., Varbrand, P. (eds.) From Local to Global Optimization. A Workshop on the Occasion of the 70th Birthday of Professor Hoang Tuy, Chap. 10, pp. 205–230. Kluwer Academic Publishers, Boston, MA (2001)
13. Huang, W.Q., Li, Y., Akeb, H., Li, C.M.: Greedy algorithms for packing unequal circles into a rectangular container. J. Oper. Res. Soc. **56**, 539–548 (2005)
14. Jakobs, S.: On genetic algorithms for the packing of polygons. Euro. J. Oper. Res. **88**, 165–181 (1996)
15. Lenstra, J.K., Rinnooy Kan, A.H.G.: Complexity of packing, covering, and partitioning problems. In: Schrijver, A. (ed.) Packing and Covering in Combinatorics, pp. 275–291. Mathematisch Centrum, Amsterdam, The Netherlands (1979)
16. Liberti, L.: Reformulation and Convex Relaxation Techniques for Global Optimization. Ph.D. Thesis, Imperial College London, London, UK (2004)
17. Liberti, L., Pantelides, C.: An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. J. Glob. Optim. **36**, 161–189 (2006)
18. Lubachevsky, B.D., Graham, R.: Dense packings of congruent circles in rectangles with a variable aspect ratio. In: Aronov, B., Basu, S., Pach, J., Sharir, M. (eds.) Discrete and Computational Geometry – The Goodman-Pollack Festschrift, vol. 25 of Algorithms and Combinatorics, pp. 633–650. Springer, Heidelberg (2003)
19. Lubachevsky, B.D., Graham, R.: Minimum perimeter rectangles that enclose congruent non-overlapping circles. ArXiv Mathematics e-prints (2004)
20. Maranas, C.D., Floudas, C.A.: Finding all solutions of nonlinearly constrained systems of equations. J. Glob. Optim. **7**, 143–182 (1995)
21. Rebennack, S., Kallrath, J., Pardalos, P.M.: Column enumeration based decomposition techniques for a class of non-convex MINLP problems. J. Glob. Optim. ∗∗∗ : ∗∗∗_∗∗∗ (2008)
22. Ruda, M.: The packing of circles in rectangles (in Hungarian). Magyar Tud. Akad. Mat. Fiz. Tud. Oszt. Közl. **19**, 73–87 (1970)
23. Rvachev, V.L., Stoyan, Y.G.: At the problem on optimal placement of circles. Cybernetics **4**, 70–75. Kiev, Ukraine (in Russian) (1965)
24. Rvachev, V.L., Stoyan, Y.G.: Solution algorithms of optimal cutting problems by circles when ristances between a pair of circles are given. Cybernetics **3**, 73–83. Kiev, Ukraine (in Russian) (1965)
25. Stoyan, Y.G., Yaskov, G.N.: Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints. Int. Trans. Oper. Res. **5**(1), 45–57 (1998)
26. Stoyan, Y.G., Yaskov, G.N.: A mathematical model and a solution method for the problem of placing various-sized circles into a strip. Euro. Jo. Oper. Res. **156**, 590–600 (2004)
27. Szabó, P.G., Markót, M.C., Csendes, T., Specht, E., Casado, L.G., García, I.: New Approaches to Circle Packing in a Square. Springer, Heidelberg (2007)

28. Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications. Nonconvex Optimization and its Applications Series. Kluwer Academic Publishers, Dordrecht The Netherlands (2002)
29. Yu, H.-X., Zhang, L.-W.: A nonlinear programming model for the packing of unequal circles into a square box. In Proceedings of the 6th World Congress on Intelligent Control and Automation, June 21–23, 2006, Dalian, China, pp. 1044–1047 (2006)