Chapter 9
Approximate Packing: Integer Programming Models, Valid Inequalities and Nesting

Igor Litvinchev, Luis Infante, and Lucero Ozuna

Abstract Using a regular grid to approximate a container, packing objects is reduced to assigning objects to the nodes of the grid subject to non-overlapping constraints. The packing problem is then stated as a large scale linear 0-1 optimization problem. Different formulations for non-overlapping constraints are presented and compared. Valid inequalities are proposed to strengthening formulations. This approach is applied for packing circular and L-shaped objects. Circular object is considered in a general sense as a set of points that are all the same distance (not necessary Euclidean) from a given point. Different shapes, such as ellipses, rhombuses, rectangles, octagons, etc., are treated similarly by simply changing the definition of the norm used to define the distance. Nesting objects inside one another is also considered. Numerical results are presented to demonstrate the efficiency of the proposed approach.

Keywords Packing problems • Integer programming • Large-scale optimization

9.1 Introduction

Packing problems generally consist of packing a set of items of known dimensions into one or more large objects or containers to minimize a certain objective (e.g. the unused part of the container or waste). Packing problems constitute a family of natural combinatorial optimization problems applied in computer science, industrial engineering, logistics, manufacturing and production processes (see, e.g., [1–4] and the references therein).

I. Litvinchev (✉)
Complex Systems Department, Computing Center, Russian Academy of Sciences, Moscow, Russia
e-mail: igorlitvinchev@gmail.com

L. Infante • L. Ozuna
Faculty of Mechanical and Electrical Engineering, Nuevo Leon Sate University, Monterrey, Mexico
e-mail: luisinfanterivera@gmail.com; luceroozuna@gmail.com

G. Fasano, J.D. Pintér (eds.), *Optimized Packings with Applications*, Springer Optimization and Its Applications 105, DOI 10.1007/978-3-319-18899-7_9

Along with industrial applications one may find packing problems in healthcare issues (e.g., [5, 6]). Wang [6] considered automated radiosurgical treatment planning for treating brain and sinus tumours. Radiosurgery uses the gamma knife to deliver a set of extremely high dose ionizing radiation, called "shots" to the target tumour area. For large target regions multiple shots of different intensity are used to cover different parts of the tumour. However, this procedure may result in large doses due to overlap of the different shots. Optimizing the number, positions and individual sizes of the shots can reduce the dose to normal tissue and achieve the required coverage.

Packing problems for regular shapes (circles and rectangles) of objects and/or containers are well studied (see, e.g., a review by [7] for circle packing). In circle packing problem the aim is to place a certain number of circles, each one with a fixed known radius inside a container. The circles must be totally placed in the container without overlapping. The shape of the container may vary from a circle, a square, a rectangular, etc. For the rectangular container there are two principal types of objectives [8, 9]: (a) regarding the circles (not necessary equal) as being of fixed size and the container as being of variable size and (b) regarding the circles and the container as being of fixed size and minimize "waste". Examples of the first approach include: minimize the perimeter or the area of the rectangle; considering one dimension of the rectangle as fixed, minimize the other dimension (strip packing or open dimension problem). For the second approach various definitions of the waste can be used. The waste can be defined in relation to circles not packed or introducing a value associated with each circle that is packed (e.g., area of the circles packed).

Many variants of packing circular objects have been formulated as nonconvex (continuous) optimization problems with decision variables being coordinates of the centres [7]. Non-overlapping typically is assured by nonconvex constraints representing that the Euclidean distance separating the centres of the circles is greater than a sum of their radii. The nonconvex problems can be tackled by available nonlinear programming (NLP) solvers, however most NLP solvers fail to identify global optima and global optimization techniques have to be used [2, 10]. The nonconvex formulations of circular packing problem give rise to a large variety of algorithms which mix local searches with heuristic procedures in order to widely explore the search space. We will refer the reader to review papers presenting the scope of techniques and applications for regular packing problem (see, e.g., [8, 9, 11–13] and the references therein).

Irregular packing problems involve non-standard shapes of objects and/or containers. Irregular shapes are those that require non-trivial handling of the geometry [14, 31]. One of the most common representations for irregular shape is a polyhedral domain which may by nonconvex or multi-connected. Heuristic and metaheuristic algorithms are the basis for the solution approaches (see [3, 15] and the references therein).

Discrete approximations of objects by tetris-like items [3] and containers by grids [15–20] were recently used to simplify packing problems. This approach allows handling irregular shapes and reduces (approximately) packing problems to discrete optimization problems. To the best of our knowledge, the proposal to use a grid was first applied by Beasley [21] in the context of cutting problems.

This work is a continuation of Litvinchev and Ozuna [17]. Using a regular grid to approximate the container, packing is reduced to assigning the objects to the nodes of the grid subject to non-overlapping constraints. Different formulations for non-overlapping are considered and compared. Valid inequalities are proposed to strengthening formulations. This approach is applied for packing circular and L-shaped objects. Circular object is considered as a set of points that are all the same distance (not necessary Euclidean) from a given point. This way different shapes, such as ellipses, rhombuses, rectangles, octagons, etc. can be treated by simply changing the norm used to define the distance. Nesting objects inside one another is also considered. Numerical results are presented to demonstrate efficiency of the proposed approach.

The rest of the work is organized as follows. In Sect. 9.2 integer programming approximation of the packing problem is presented along with different formulations for non-overlapping. In Sect. 9.3 the proposed approach is applied to packing circular objects. Experimental results for packing different circular shapes are provided to demonstrate usefulness of valid inequalities proposed in Sect. 9.2. L-shaped objects and containers are considered in Sect. 9.4, while Sect. 9.5 presents concluding remarks and directions for the future research.

9.2 Basic Constructions

Suppose we have non-identical objects G_k, $k \in K = \{1, 2, \ldots K\}$ which have to be packed in a container G. In what follows we will use the same notation G_k, G for the domain in $\mathbb{R}^n$ and for its boundary assuming that it is easy to understand from the context what do we mean. It is assumed that no two objects overlap with each other and each packed object lies entirely in the container. Denote by S_k the area of G_k. Let at most M_k objects G_k are available for packing and at least m_k of them have to be packed. Denote by p_i, $i \in I = \{1, 2 \ldots, n\}$ the nodes of a grid covering the container, $p_i \in G$. It is assumed that the position of the object in the container is completely characterized by the position of its reference point. Define binary variables $x_i^k = 1$ if the reference point of the object G_k is assigned to the node i; $x_i^k = 0$ otherwise. In what follows we will say that the object is assigned to the node i if the corresponding reference point is assigned to that node and will denote this as G_k^i. For fixed i, k let

$$N_{ik} = \left\{ j, l : i \neq j \text{ such that } G_k^i \text{ overlaps with } G_l^j \right\}.$$

Let n_{ik} be the cardinality of N_{ik} : $n_{ik} = |N_{ik}|$. Then the problem of maximizing the area covered by the objects can be stated as follows:

$$\max \sum_{i \in I} \sum_{k \in K} S_k x_i^k \tag{9.1}$$

subject to

$$m_k \le \sum_{i \in I} x_i^k \le M_k, \quad k \in K, \tag{9.2}$$

$$\sum_{k \in K} x_i^k \le 1, \quad i \in I, \tag{9.3}$$

$$x_i^k = 0 \text{ for } G_k^i \setminus \left(G \cap G_k^i\right) \ne \varnothing \quad \text{for } i \in I,\ k \in K, \tag{9.4}$$

$$x_i^k + x_j^l \le 1, \text{ for } i \in I,\ k \in K,\ (j,l) \in N_{ik}, \tag{9.5}$$

$$x_i^k \in \{0,1\}, \quad i \in I,\ k \in K. \tag{9.6}$$

Constraints (9.6) ensure that the number of objects packed is between m_k and M_k; constraints (9.3) that at most one object is assigned to any node; constraints (9.4) that G_k cannot be assigned to the node i if G_k^i is not totally placed inside G; pairwise constraints (9.5) guarantee that there is no overlapping between the objects; constraints (9.6) represent the binary nature of variables.

Remark 2.1 Linear non-overlapping constraints (9.5) are equivalent to a single quadratic constraint

$$Q(x) \equiv \sum\nolimits_{i,k} x_i^k \sum\nolimits_{j,l \in N_{ik}} x_j^l = 0 \ \ (\le 0)\,. \tag{9.7}$$

If (9.7) holds, then for $x_i^k = 1$ we have $\sum_{j,l \in N_{ik}} x_j^l = 0$ yielding $x_j^l = 0,\ \ (j,l) \in N_{ik}$, and if $x_j^l = 1$ at least for one pair $(j,l) \in N_{ik}$, then $x_i^k = 0$. Thus (9.5) can be considered as a specific linearization of (9.7). Other linearizations and relaxations of (9.7), e.g. used for the quadratic assignment problem [22] can also be considered.

Below we present different formulations for the non-overlapping constraints (9.5) which remain valid for the general definition of N_{ik}.

By the definition of N_{ik} if $(j,l) \in N_{ik}$, then $(i,k) \in N_{jl}$. Thus a half of the constraints in (9.5) are redundant since we have:

$$x_i^k + x_j^l \le 1, \text{ for } i \in I,\ k \in K,\ (j,l) \in N_{ik},$$

$$x_j^l + x_i^k \le 1, \text{ for } j \in I,\ l \in K,\ (i,k) \in N_{jl}.$$

We may eliminate any (none) of these two constraints to get the *reduced* equivalent formulation. This can be represented by multiplying constraints (9.5) by a fixed $\lambda_j^l \in \{0,1\}$:

$$x_i^k \lambda_j^l + x_j^l \lambda_j^l \le \lambda_j^l, \text{ for } i \in I,\ k \in K,\ (j,l) \in N_{ik}, \tag{9.8}$$

subject to $\lambda_j^l+\lambda_i^k \geq 1$. This way either one of the redundant constraints is eliminated ($\lambda_j^l + \lambda_i^k = 1$) or no-one ($\lambda_j^l + \lambda_i^k = 2$). Since eliminating redundant constraints does not affect the feasible set, the problem (9.1)–(9.6) is equivalent to (9.1)–(9.4), (9.6), (9.8) for any λ fulfilling the normalized condition

$$\lambda \in \Lambda = \left\{\lambda_j^l \in \{0,1\} : \lambda_j^l + \lambda_i^k \geq 1, \ \ (j,l) \in N_{ik}\right\}.$$

Similar to plant location problems [23] we can state non-overlapping conditions in a more *compact* form. Summing up constraints (9.7) over $(j,l) \in N_{ik}$ we get

$$x_i^k \sum_{(j,l)\in N_{ik}} \lambda_j^l + \sum_{(j,l)\in N_{ik}} \lambda_j^l x_j^l \leq \sum_{(j,l)\in N_{ik}} \lambda_j^l, \ \text{for } i \in I, k \in K. \tag{9.9}$$

Proposition 2.1 *For any $\lambda \in \Lambda$ constraints* (9.5), (9.6) *are equivalent to constraints* (9.6), (9.9).

Proof If constraints (9.5) are fulfilled, then obviously constraints (9.9) hold by construction. Now let constraints (9.9) are fulfilled. Define

$$N_{ik}^1 = \left\{(j,l) \in N_{ik} : \lambda_j^l = 1\right\}, \ N_{ik}^0 = \left\{(j,l) \in N_{ik} : \lambda_j^l = 0\right\}, \ N_{ik}^1 \cup N_{ik}^0 = N_{ik},$$
$$\left|N_{ik}^1\right| = n_{ik}^1, \ \left|N_{ik}^0\right| = n_{ik}^0.$$

By (9.9) we have

$$x_i^k n_{ik}^1 + \sum_{(j,l)\in N_{ik}^1} x_j^l \leq n_{ik}^1$$

and hence,

$$\text{if } x_i^k = 1, \ \text{then } x_j^l = 0 \text{ for } \ (j,l) \in N_{ik}^1. \tag{9.10}$$

By the definition, if $(j,l) \in N_{ik}$, then $(i,k) \in N_{jl}$. Thus by (9.9) we have

$$x_j^l \sum_{(i,k)\in N_{jl}} \lambda_i^k + \sum_{(i,k)\in N_{jl}} \lambda_i^k x_i^k \leq \sum_{(i,k)\in N_{jl}} \lambda_i^k \ \text{for } j \in I, \ l \in K. \tag{9.11}$$

In particular, (9.11) is fulfilled for $(j,l) \in N_{ik}^0$. Since $\lambda_j^l + \lambda_i^k \geq 1$, then for $(j,l) \in N_{ik}^0$ all λ_i^k in (9.11) are positive ($\lambda_i^k = 1$). Then by (9.11) we have:

$$\text{if } x_j^l = 1 \text{ for at least one } \ (j,l) \in N_{ik}^0, \ \text{then } x_i^k = 0. \tag{9.12}$$

Note that constraints (9.5) can be interpreted in two ways. First if $x_i^k = 1$, then $x_j^l = 0$ for all $(j,l) \in N_{ik}$. Second, if $x_j^l = 1$ for at least one $(j,l) \in N_{ik}$, then $x_i^k = 0$. Combining (9.10) and (9.12) we may conclude that if constraints (9.9) are fulfilled, then (9.5) hold. □

Remark 2.2 In Galiev and Lisafina [16] the compact formulation

$$x_i n_i + \sum_{j \in N_i} x_j \le n_i \text{ for } i \in I \tag{9.13}$$

was used to represent non-overlapping constraints for the case of packing identical circles. This corresponds to a singleton set K and all multipliers λ equal to 1 in (9.9).

Remark 2.3 Proposition 2.1 remains true for nonnegative (not necessary binary) multipliers λ subject to $\lambda_j^l + \lambda_i^k \neq 0$. The proof is similar.

As follows from Proposition 2.1, the non-overlapping constraints can be stated in different forms (see [20] for an illustrative example). We have a family of formulations equivalent to (9.5) and obtained for different multipliers λ in (9.9). To compare equivalent formulations, let

$$P_1 = \left\{x \ge 0 : x_i^k + x_j^l \le 1, \text{ for } i \in I,\ k \in K, (j,l) \in N_{ik}\right\},$$

$$P_2 = \left\{x \ge 0 : x_i^k \sum_{(j,l) \in N_{ik}} \lambda_j^l + \sum_{(j,l) \in N_{ik}} \lambda_j^l x_j^l \le \sum_{(j,l) \in N_{ik}} \lambda_j^l,\ \ i \in I,\ \ k \in K\right\},$$

where multipliers λ in P_2 fulfil the normalizing condition stated in Proposition 2.1.

Proposition 2.2 $P_1 \subset P_2$.

Proof Since constraints of P_2 are a linear combination of those in P_1 with nonnegative multipliers λ, then $P_1 \subseteq P_2$. To show that $P_1 \subset P_2$ we need to find a point in P_2 that is not in P_1.

This point can be constructed as follows. Choose $(i,k) \in N_{jl}$ and $(j,l) \in N_{ik}$ such that $\sum_{(j,l) \in N_{ik}} \lambda_j^l,\ \sum_{(i,k) \in N_{jl}} \lambda_i^k \ge 2$. Set to zero all the variables except $x_i^k,\ x_j^l$. Obviously all constraints in P_2 corresponding to zero variables are fulfilled. Define $x_i^k,\ x_j^l$ to fulfil the two remaining constraints as equalities:

$$x_i^k \sum_{(j,l) \in N_{ik}} \lambda_j^l + x_j^l = \sum_{(j,l) \in N_{ik}} \lambda_j^l, \quad x_j^l \sum_{(i,k) \in N_{jl}} \lambda_i^k + x_i^k = \sum_{(i,k) \in N_{jl}} \lambda_i^k.$$

Denote $\overline{n}_{ik} = \sum_{(j,l) \in N_{ik}} \lambda_j^l,\ \ \overline{n}_{jl} = \sum_{(i,k) \in N_{jl}} \lambda_i^k$ with $\overline{n}_{ik}, \overline{n}_{jl} \ge 2$. The corresponding solution of the two equations above is

$$x_i^k = \frac{\overline{n}_{jl}\,(\overline{n}_{ik} - 1)}{\overline{n}_{jl}\overline{n}_{ik} - 1} < 1, \quad x_j^l = \frac{\overline{n}_{ik}\,(\overline{n}_{jl} - 1)}{\overline{n}_{jl}\overline{n}_{ik} - 1} < 1$$

with

$$x_i^k + x_j^l = 1 + \frac{1 + \overline{n}_{jl}\overline{n}_{ik} - \overline{n}_{jl} - \overline{n}_{ik}}{\overline{n}_{jl}\overline{n}_{ik} - 1} > 1.$$

This point violates corresponding constraint in P_1 and hence $P_1 \subset P_2$ as desired. □

As follows from Proposition 2.2, the pairwise formulation (9.1)–(9.6) is stronger than the compact one (9.1)–(9.4), (9.6), (9.9) in the sense of Wolsey [23].

In general, checking if the object is not totally placed inside the container is tricky. However, for a convex container and a polygonal object this problem can be simplified as stated below.

Proposition 2.3 *Let G be a convex set and G_k be a (not necessary convex) polygon. Let v_{ki}^t, $t = 1, \ldots, T_k$ be all vertices of G_k^i. Then $G_k^i \subseteq G$ iff $v_{ki}^t \in G$, $t = 1, \ldots, T_k$.*

Proof If $G_k^i \subseteq G$, then obviously all vertices of G_k^i are in G. Let now $v_{ki}^t \in G$, $t = 1, \ldots, T_k$. Consider the convex hull of G_k^i, $conv\left(G_k^i\right) = \left\{y : y = \sum_t \alpha_t v_{ki}^t,\ \sum_t \alpha_t = 1,\ \alpha_t \geq 0\right\}$. Since all vertices of G_k^i are in G, then by convexity of G any convex linear combination of vertices also belongs to G and hence $conv\left(G_k^i\right) \subseteq G$. By the definition of convex hull, $G_k^i \subseteq conv\left(G_k^i\right)$ and hence $G_k^i \subseteq G$ as desired. □

Good upper (dual) bounds are very important to solve integer programming problems. We may expect that the upper bound obtained by the linear programming relaxation of the problem (9.1)–(9.6) provides a poor upper bound for the optimal objective. For example, for packing equal circles in a rectangular container the objective value of the LP-relaxation grows linearly with respect to the number of grid nodes (see [20] for details).

To tightening the LP-relaxation we consider valid inequalities ensuring that no grid node is covered by two objects. To present this family, define matrix $[\alpha_{ij}^k]$ as follows. Let $\alpha_{ij}^k = 1$ if G_k^i covers a node j, $\alpha_{ij}^k = 0$ otherwise. The following constraints ensure that no nodes of the grid can be covered by two objects:

$$\sum_{k \in K} \sum_{j \in I} \alpha_{ij}^k x_j^k \leq 1, \quad i \in I. \tag{9.14}$$

Note that (9.14) is not equivalent to the non-overlapping constraints (9.5).

9.3 Circular Objects

Define a circular object C_k as a set of points that all are at most the distance R_k from a given point called centre, $C_k = \{y : \|y - y_{0k}\| \le R_k\}$. Here the norm used to define the object is not necessary the Euclidean [32]. Let d_{ij} be the distance between node points i, j in the sense of the norm used to define the circular object.

The set N_{ik} in (9.5) is now defined as follows: $N_{ik} = \{j, l : i \ne j, \ \ d_{ij} < R_k + R_l\}$. For matrix $[\alpha_{ij}^k]$ in (9.14) we have $\alpha_{ij}^k = 1$ for $d_{ij} < R_k$, $\alpha_{ij}^k = 0$ otherwise.

Using different norms we can use constructions of the previous section for packing different geometrical objects of the same shape. For example, a circular object in the maximum norm $\|y\|_\infty := \max_r \{|y_r|\}$ is represented geometrically by a square, taxicab norm $\|y\|_1 := \sum_r |y_r|$ yields a rhombus. In a similar way we may handle rectangles, ellipses, etc. Using a superposition of norms, we can consider more complex circular objects. For

$$\|y\| := \max_r \left\{|y_r|, \gamma \sum\nolimits_r |y_r|\right\}$$

and a suitable $0.5 < \gamma < 1$ we get an octagon, an intersection of a square and a rhombus.

A numerical experiment was designed to evaluate the performance of different non-overlapping formulations and to see the impact of the valid inequalities for packing circular objects in a rectangular container.

In the first part of the experiment the test bed set of 9 instances from ([16], Table 3) was used for packing maximal number of circles into a rectangle of width 3 and height 6. A rectangular uniform grid of size Δ along both sides of the container was used. It was assumed that the supply of the objects is unlimited and constraints (9.2) were relaxed. Similar to [16] the nodes located too close (close than a radius) to the boundary were eliminated from consideration and thus constraints (9.4) were omitted. In all experiments optimization problems were solved by the system CPLEX 12.6 [24]. The runs were executed on a desktop computer with CPU AMD FX 8350 8-core processor 4 GHz and 32 GB RAM.

The following four formulations were compared: pairwise formulation (9.1)–(9.6) (Cmpl), reduced formulation (9.1)–(9.6) without redundant constraints (CmplH), compact formulation (9.13) as in Galiev and Lisafina [16] (Cmpct), and compact formulation obtained by summing up constraints in the reduced formulation (9.1)–(9.6) (CmpctH). All these four formulations were combined with valid inequalities (cuts) (9.14), the corresponding formulations are denoted by CmplC, CmplHC, CmpctC, CmpctHC. The results of the numerical experiment are given in Table 9.1. Here the first three columns present instance number, circle radius, and grid size Δ. The last columns give CPU time (in seconds) for different formulations. For all problem instances $mipgap = 0$ was set for running CPLEX. In this table asterisk indicates that the computation was interrupted after

Table 9.1 CPU-time for circles (*gap* 0 %)

#	R	Δ	Cmpl	CmplC	CmplH	CmplHC	Cmpct	CmpctC	CmpctH	CmpctHC
1	0.5	0.125	2	2	1	1	276	4	5	4
2	0.625	0.078125	71	15	41	11	1,040	35	50	12
3	0.5625	0.0625	337	82	186	75	11,666	87	831	72
4	0.375	0.09375	6	9	4	4	2,698	29	169	92
5	0.3125	0.078125	96	163	114	189	*	819	*	1,027
6	0.4375	0.546875	17,437	1,392	17,654	1,379	*	39,347	*	*
7	0.25	0.0625	*	3,531	*	3,178	*	*	*	*
8	0.275	0.06875	132	87	177	87	*	2,523	*	2,860
9	0.1875	0.046875	*	17,437	*	*	*	*	*	*

Table 9.2 LP-relaxations

#	n_Δ	LP	O	LPC	R	LPC	C	LPC	E	LPC
1	697	348.5	**18**	19	**28**	33.43	**18**	19	**34**	36
2	1,403	701.5	**9**	10	**15**	16.87	**10**	10	**21**	25
3	2,449	1, 224.5	**12**	14.0743	**20**	22.25	**13**	14.07	**27**	29.91
4	1,425	712.5	**26**	30.9485	**39**	41.37	**32**	36.33	**59**	68.86
5	2,139	1, 069.5	**41**	53.4043	**76**	94.76	**45**	53.4	**99**	110
6	3,666	1, 833.5	**20**	22.5537	**35**	39.72	**21**	23.86	**43**	49.787
7	3,649	1, 824.5	**72**	90.9767	**127**	157.96	**74**	90.98	**137**	182
8	2,880	1, 440	**50**	59.014	**75**	79.53	**61**	72	**108**	134.56
9	6,897	3, 448.5	**106**	134.342	**167**	182.28	**140**	162	**261**	273.61

the computation time exceeded 12-h CPU. Number of binary variables and optimal packings are presented in Table 9.2 in columns (n_Δ) and (C), correspondingly.

As we can see from Table 9.1, CPU time for complete formulations is lower than for the compact, especially for large instances. Eliminating redundant constraints typically (but not always) reduces CPU time. Although eliminating redundancy does not change corresponding LP-relaxation, it may affect the path selected by branch and bound technique and thus result in increase/decrease of CPU time.

Introducing valid inequalities decreases CPU time for all problem instances and for all problem formulations. Although introducing valid inequalities slightly increases time to solve the LP-relaxation, the effect of improving quality of the LP-bound becomes more important for the convergence of the overall branch and bound scheme. That is why CPU time decreases significantly for hard instances 6, 7, 9, while for "easy" instances the decrease may be relatively modest. Moreover, with valid inequalities CPU time necessary to get provably optimal solution ($mipgap = 0$) is comparable with that reported in Galiev and Lisafina [16] for their heuristic approach.

Table 9.2 presents values of the LP-relaxations with/without valid inequalities for packing equal circles (C), ellipses (E), rhombuses (R) and octagons (O) into the

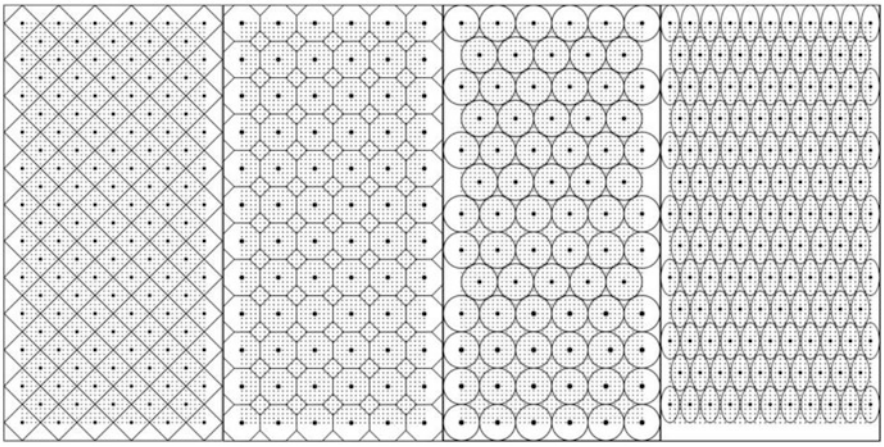

Fig. 9.1 Packing equal circular objects for instance 7

same 3×6 rectangle using the same values Δ for the grid. The standard Euclidean and taxicab norms were used to define circles and rhombuses, while norms

$$\|y\| := \left(2y_1^2 + y_2^2\right)^{1/2} \text{ and } \|y\| := \max\left\{|y_1|,\ |y_2|,\ \left(1/\sqrt{2}\right)(|y_1| + |y_2|)\right\}$$

were used for ellipses and octagons. The same values of radii as in Table 9.1 were used to define circular objects. In Table 9.2 the first three columns present instance number, number of binary variables (n_Δ) and value of the LP-relaxation without valid inequalities (LP). For all circular objects the optimal value of the LP-relaxation was $0.5n_\Delta$ (all variables equal to 0.5). The last eight columns give the value of the optimal integer solution (in bold) and the value LPC of the LP-relaxation improved by the valid inequalities (next to bold). We see that introducing valid inequalities improves significantly the quality of the LP bound for all shapes of the objects. The detailed study of this subject for the case of circles one can find in Litvinchev et al. [20] for the same test bed instances. Packings for the instance 7 are presented in Fig. 9.1.

In many applied problems packing smaller objects inside a larger one is permitted. For example, in tube industry the tubes are produced in a continuous extract machine and cut to the length of the container used for shipping. Before being placed in the container they may be inserted inside other, thicker tubes, so that usage of container space is maximized. Since all the tubes have the same length, maximizing container load is equivalent to maximizing the area filled with circles (rings) in a section of the container. Similar problems arise, e.g. in stacking up different containers to form a tower [25] and in visualization of large hierarchical data by 3D nested cylinders [26]. In tube industry the process is usually named telescoping [27], in optimized packing context the terms nesting [2] or recursive

packing [28] are used. Although the term nesting is also used for packing irregular objects [15], we will use nesting for packing smaller objects inside larger ones assuming that it is easy to understand from the context what do we mean.

To consider nesting circular objects inside one another, we only need to modify the non-overlapping constraints. In order to C_k^i be non-overlapping with other objects being packed (including objects placed inside C_k^i), it is necessary that $x_j^l = 0$ for $j \in I, \quad l \in K$, such the $R_k - R_l < d_{ij} < R_k + R_l$ for $R_k > R_l$. Let

$$\Omega_{ik} = \left\{j, l : i \neq j, \;\; R_k - R_l < d_{ij} < R_k + R_l, \; R_k > R_l\right\}.$$

Then the non-overlapping constraints for packing circular objects with nesting can be stated as

$$x_i^k + x_j^l \leq 1, \;\; \text{for } i \in I; k \in K; (j, l) \in \Omega_{ik}. \tag{9.15}$$

Constraints (9.3) have to be omitted in case of nesting.

If nesting is permitted it may be necessary to take into account the difference between external and internal sizes of the object, i.e. consider the object as a circular ring (a region bounded by two concentric circular objects) having a positive thickness. To consider nesting-subject-to-thickness we need only to redefine the set Ω_{ik}. Let g_k be the thickness of the circle C_k. For Ω_{ik} defined as

$$\Omega_{ik} = \left\{j, l : i \neq j, \;\; R_k - g_k - R_l < d_{ij} < R_k + R_l, \;\; R_k - g_k > R_l\right\}$$

we get non-overlapping constraints similar to (9.15).

The results for packing two different octagons in a square 30×30 container maximizing the total area of the packed objects are presented in Table 9.3. Here the first three columns give instance number, radii, and a number of grid nodes (integer variables). The last columns give the total area without nesting (N−), with nesting (N+) and with nesting and thickness (N+T), number of small (O1) and large (O2) objects packed, as well as corresponding CPU time in sec. The thickness g_k was defined as $0.1R_k$. The packings obtained for the instance 1 are presented in Fig. 9.2.

Table 9.3 Packing 2 different octagons

#	R_1, R_2	n_Δ	N-	O1, O2	CPU	N+	O1, O2	CPU	N+T	O1, O2	CPU
1	0.6, 6.3	441	627.48	85, 4	1	842.21	265, 4	1	804.37	233, 4	1
2	0.6, 6.3	961	699.06	145, 4	6	971.05	373, 4	3	910.209	322, 4	5
3	1, 5.3	441	699.35	41, 6	1	952.82	119, 6	1	922.99	110, 6	1
4	1, 5.3	961	750.09	114, 4	57	1,158.27	181, 6	129	1,019.1	139, 6	49

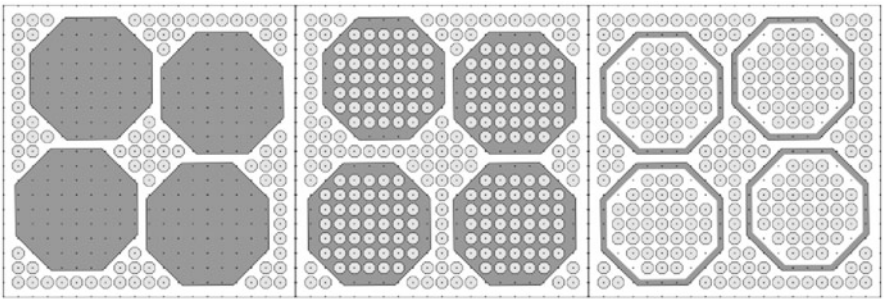

Fig. 9.2 Packing two octagons for instance 1

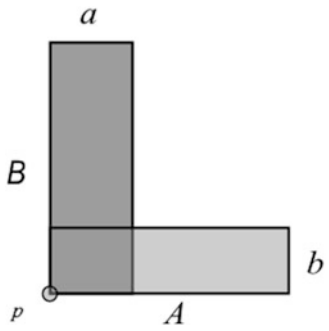

Fig. 9.3 L-object

9.4 L-shaped Objects and Containers

In this section we consider packing L-shaped objects. These shapes appear, e.g. in packing interpretations of scheduling with non-constant operational cycles [3]. Let L-object (see Fig. 9.3) be a superposition of rectangles $(A \times b)$ and $(a \times B)$ with edges parallel to the principal axes, $A > a > 0$, $B > b > 0$ and the principal corner considered as a reference point.

To state the problem (9.1)–(9.6) we need to specify constraints (9.4), (9.5), i.e. present a constructive way to check if the object is totally placed inside the container and if the objects overlap. Suppose we have two L-objects, L_i and L_j, with the reference points located at (y_{1i}, y_{2i}) and (y_{1j}, y_{2j}). Introducing binary variables $z_i, z_j \in \{0, 1\}$ these objects can be represented as follows:

$$L_i = \Big\{ (y_1, y_2, z_i) : z_i \in \{0, 1\}, \;\; 0 \le y_1 - y_{1i} \le A_i + z_i \, (a_i - A_i) \, ,$$

$$0 \le y_2 - y_{2i} \le b_i + z_i \, (B_i - b_i) \Big\},$$

$$L_j = \Big\{ \big(y_1, y_2, z_j\big) : z_j \in \{0,1\},\ \ 0 \le y_1 - y_{1j} \le A_j + z_j \big(a_j - A_j\big),$$

$$0 \le y_2 - y_{2j} \le b_j + z_j \big(B_j - b_j\big) \Big\}.$$

We wonder if $L_i \cap L_j \neq \varnothing$. This holds if the system of inequalities

$$\max\big\{y_{1i}, y_{1j}\big\} \le y_1 \le \min\big\{y_{1i} + A_i + z_i\,(a_i - A_i),\ \ y_{1j} + A_j + z_j\big(a_j - A_j\big)\big\},$$

$$\max\big\{y_{2i}, y_{2j}\big\} \le y_2 \le \min\big\{y_{2i} + b_i + z_i\,(B_i - b_i),\ \ y_{2j} + b_j + z_j\big(B_j - b_j\big)\big\}$$

is consistent at least for one combination of binary z_i, z_j. This can be verified by inspection.

Substituting $z_i = z_j = 0$ yields

$$\max\big\{y_{1i}, y_{1j}\big\} \le \min\big\{y_{1i} + A_i, y_{1j} + A_j\big\},\ \ \max\big\{y_{2i}, y_{2j}\big\} \le \min\big\{y_{2i} + b_i, y_{2j} + b_j\big\}.$$

For $z_i = z_j = 1$ we have

$$\max\big\{y_{1i}, y_{1j}\big\} \le \min\big\{y_{1i} + a_i, y_{1j} + a_j\big\},\ \ \max\big\{y_{2i}, y_{2j}\big\} \le \min\big\{y_{2i} + B_i, y_{2j} + B_j\big\}.$$

Substituting $z_i = 1,\ \ z_j = 0$ yields

$$\max\big\{y_{1i}, y_{1j}\big\} \le \min\big\{y_{1i} + a_i, y_{1j} + A_j\big\},\ \ \max\big\{y_{2i}, y_{2j}\big\} \le \min\big\{y_{2i} + B_i, y_{2j} + b_j\big\}.$$

And finally for $z_i = 0,\ \ z_j = 1$ we get

$$\max\big\{y_{1i}, y_{1j}\big\} \le \min\big\{y_{1i} + A_i, y_{1j} + a_j\big\},\ \ \max\big\{y_{2i}, y_{2j}\big\} \le \min\big\{y_{2i} + b_i, y_{2j} + B_j\big\}.$$

Thus if at least one pair of inequalities above hold, then $L_i \cap L_j \neq \varnothing$. In a similar way we can check overlapping for the other composite objects, e.g., for star-shapes represented as a superposition of a square and a rhombus.

To check if L-object is totally placed inside a convex container we can use Proposition 2.3 since all vertices of the object are easily identified. However, for rectangular and L-shaped containers with all edges parallel to the principal axes we can state constraints (9.4) based on simple geometrical considerations.

Below we present results of a numerical experiment for packing L-objects in rectangular and L-shaped containers. The normalized objective was defined as the total area of the objects divided over the area of the smallest object. For the case of equal objects the normalized objective coincides with the number of objects.

In the first part of the experiment the test bed set of 6 instances was used for packing maximal number of equal L-objects into a rectangular container of width 3 and height 6. Two types of the objects were considered with the shapes corresponding to $A = B = 2R$ and $B = 0.5A = 2R$. The thickness of

Table 9.4 Equal L-objects

#	R	n_Δ	z	LP	LP+C	T	T+C	Z	LP	LP+C	T	T+C
1	0.5	3,321	37	1,163	135.5	16	11	22	1,029	117.2	25	14
2	0.625	2,145	20	680.7	107.7	4	3	10	580.1	79.08	4	4
3	0.5625	2,556	25	851.7	127.5	8	5	14	739.8	92.45	8	8
4	0.375	5,778	75	2,223	271.6	150	150	42	2,036	195.6	610	330
5	0.3125	8,385	116	3,379	384.9	1,867	620	71	3,144	275.4	3,581	1,930
6	0.4375	4,186	48	1,537	201.5	76	76	29	1,383	146.5	73	55

Table 9.5 Packing 2 different L-objects

#	R_1, R_2	n_Δ	z	L1	L2	T	LR	LR+C	L1+	L2+	T+
1	0.6, 6.3	3,969	750	309	4	* (12 %)	78,384	5,980	681	4	* (7.8 %)
2	0.6, 6.3	3,969	447.5	227	2	530	51,404	4,931	400	2	* (6.0 %)
3	1, 5.3	1,369	271.2	215	2	17	8,458	1,352	197	6	* (5.7 %)
4	1, 5.3	1,369	147.2	91	2	16	6,221	1,107	114	3	440

L-object was defined as $a = b = 0.3R$ in both cases. A rectangular uniform grid of size $\Delta = 0.15R$ (a half of the thickness) was used. The results of the numerical experiment are given in Table 9.4. The first three columns present instance number, value of R and a number of binary variables n_Δ. The next five columns present indicators for the case $A = B$: the optimal value of integer solution z; value of the LP-relaxation without and with valid cuts, LP and LP+C; CPU time in sec. to get integer solution without and with valid cuts, T and T+C. The last five columns present similar indicators for the case $B = 0.5A$. For all problem instances $mipgap = 0$ was set for running CPLEX.

As we can see from Table 9.4 introducing valid inequalities improves significantly the LP-bound and reduces CPU-time, especially for hard instances.

In the second part of the experiment two different L-objects were packed in a square 30×30 container maximizing the total normalized area of the packed objects. Four instances were considered according to the shape of the objects. For instances 1 and 3 $A = B = 2R$ and for instances 2 and 3, $B = 0.5A = 2R$. In all cases $a = b = R$. Two values of R were considered and for $R = R_2$ (large object) the minimal number of the objects to be packed was set to two, $m_2 = 2$ in (9.2).

The results are presented in Table 9.5. Here the first four columns give instance number, radii R_1, R_2, number n_Δ of grid nodes (integer variables) and the value z of the optimal solution. Columns 5 and 6 give the number of small (L1) and large (L2) objects in the optimal solution, while column 7 indicates corresponding CPU time in sec. for the case of using the valid inequalities (9.14). Asterisk indicates that the computation was interrupted after the computation time exceeded 1,800 s. CPU time and the value in parenthesis gives the corresponding *mipgap*. Columns 8 and 9 present the value of the LP-relaxation without (LR) and with (LR+C) valid inequalities. The last three columns give the number of objects packed (L1+, L2+)

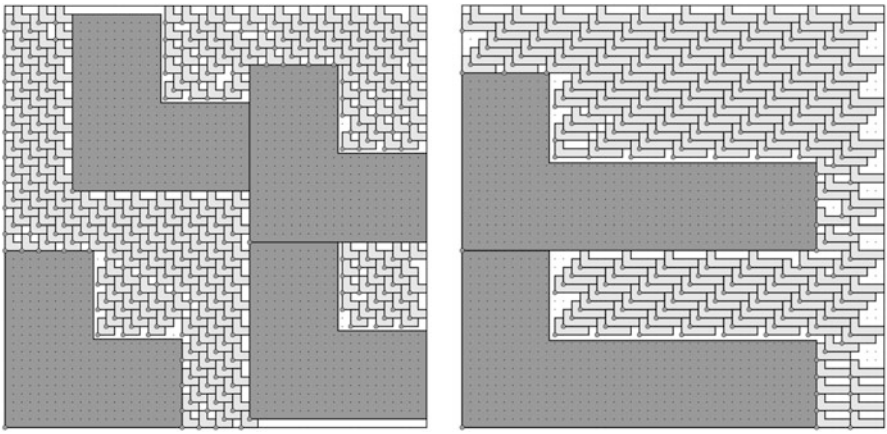

Fig. 9.4 Instances 1, 2

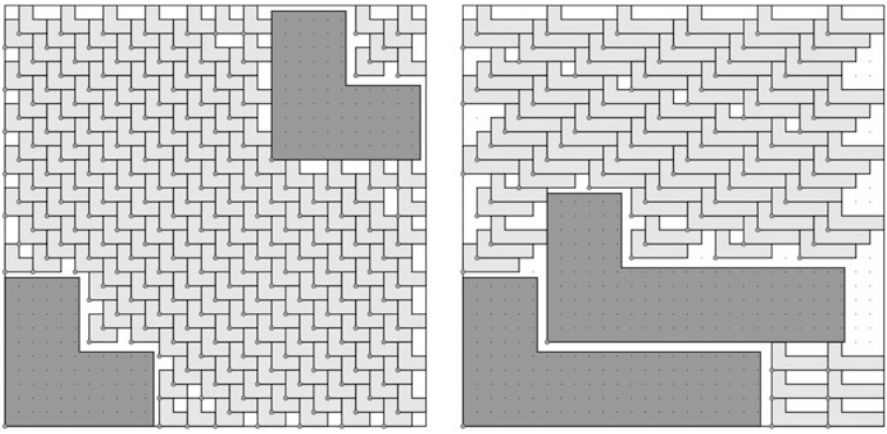

Fig. 9.5 Instances 3, 4

and CPU time (T+) for the case of nesting allowed. Optimal packings for instances 1–4 are presented in Figs. 9.4 and 9.5 for the case without nesting and in Figs. 9.6 and 9.7 for nesting allowed.

In the final part of experimentation L-shaped container was considered for $A = B = 30$, $a = b = 12$. Two instances were considered according to the shape of the two different L-objects. For the first instance $A = B = 2R$ for both objects and for the second $B = 0.5A = 2R$. In all cases $a = b = R$. Two values of R were used, $R_1 = 1$, $R_2 = 5.3$ and for $R = R_2$ we set $m_2 = 2$ in (9.2). A rectangular uniform grid of size $\Delta = \min\{R_1, R_2\} = 1$ was used giving $n_\Delta = 637$ grid nodes in the L-container. The optimal solution was obtained in less than 1 s. CPU time. For the first instance ($A = B = 2R$) the optimal solution gives 108 small and 2 large

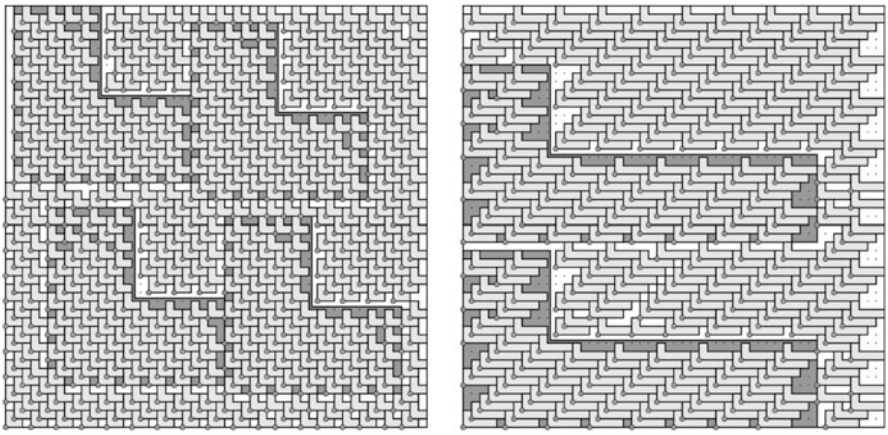

Fig. 9.6 Instances 1, 2 with nesting

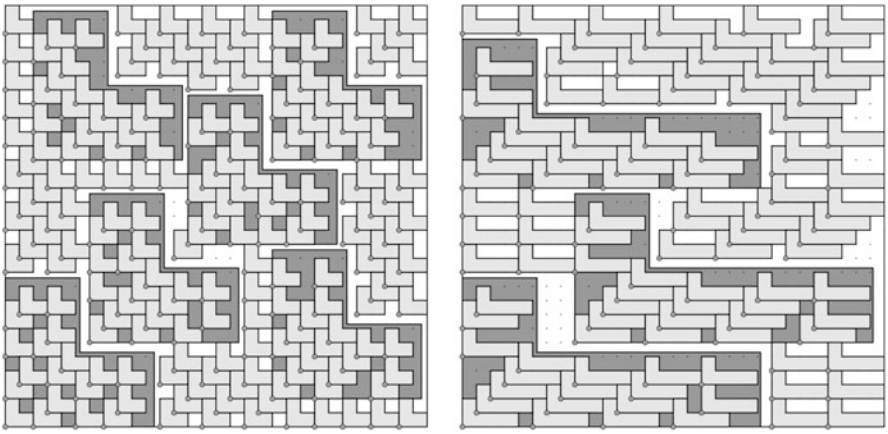

Fig. 9.7 Instances 3, 4 with nesting

L-objects without nesting and (120, 4) for nesting allowed. For the second instance ($B = 0.5A = 2R$) we get (33, 2) and (71, 2) objects, respectively. The optimal packings are presented in Figs. 9.8 and 9.9.

9.5 Conclusions

Integer programming formulations were considered for approximated packing objects in a container. Using a grid approximation of the container packing problems can be transformed into optimal assignment of the objects (reference points) to nodes of the grid subject to non-overlapping constraints. In this work we used

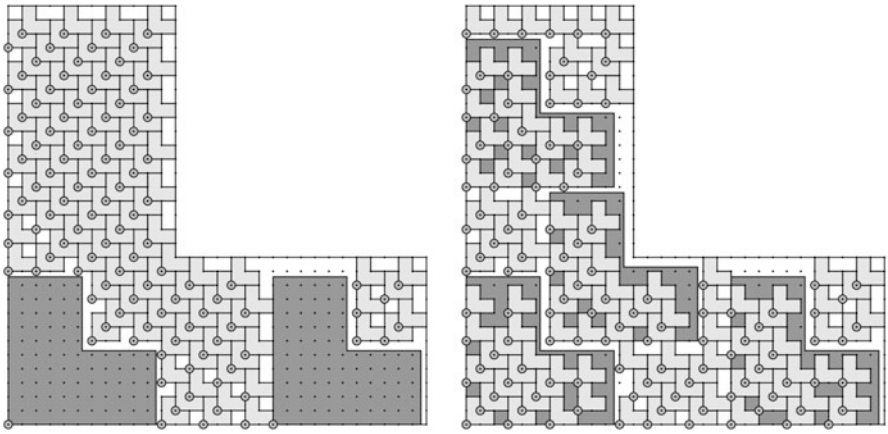

Fig. 9.8 Instance 1

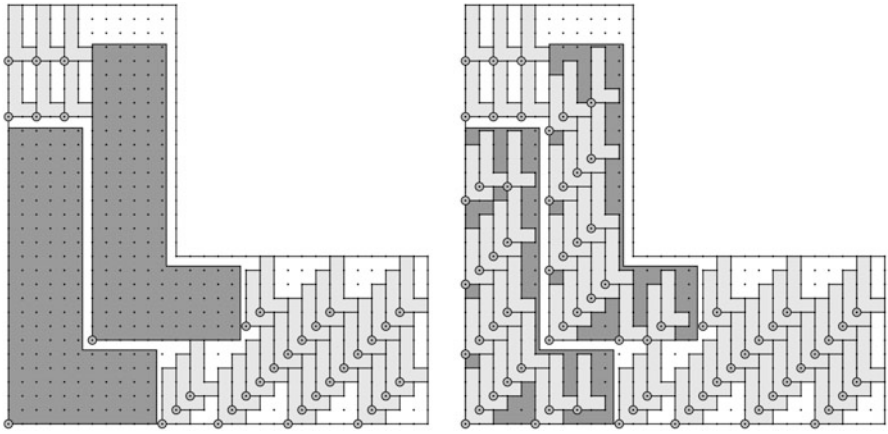

Fig. 9.9 Instance 2

linear non-overlapping constraints. However, as noted in Remark 2.1, the problem (9.1)–(9.6) is closely related to the quadratic assignment problem and corresponding approaches can be also used for packing problems. Some results in this direction are in course.

Valid inequalities (9.14) were proposed to strengthening the formulation and our numerical experiments demonstrate that the value of the LP-relaxation can be tightened significantly by (9.14). Moreover, aggregating valid cuts not only improve the value of the relaxation, but also change the structure of the optimal LP-solution. A simple LP-based heuristic is proposed in Litvinchev et al. [29] for packing circular objects.

Grid approximation of the container results in a large-scale integer optimization problem. Using decomposition and/or aggregation techniques [30] to split the nodes

of the grid into smaller subsets (container decomposition) and/or creating "macro nodes" (nodes aggregation) may be helpful to cope with high dimension. Some results in this direction are in course.

A critical question in grid approximation is how to choose parameters of the grid, e.g. shape and number of nodes, to get a reasonable trade-off between computational burden and proximity to the true optimal packing. The use of non-uniform and/or adaptive grids seems to be interesting direction for the future research.

Acknowledgements This work was partially supported by Grants from RFBR, Russia (12 01 00893 a), and CONACYT, Mexico (167019).

References

1. Baltacioglu, E., Moore, J.T., Hill, R.R.: The distributor's three-dimensional pallet-packing problem: a human-based heuristical approach. Int. J. Oper. Res. **1**, 249–266 (2006)
2. Castillo, I., Kampas, F.J., Pinter, J.D.: Solving circle packing problems by global optimization: numerical results and industrial applications. Eur. J. Oper. Res. **191**, 786–802 (2008)
3. Fasano, G.: Solving Non-standard Packing Problems by Global Optimization and Heuristics. Springer-Verlag, Berlin (2014)
4. Frazer, H.J., George, J.A.: Integrated container loading software for pulp and paper industry. Eur. J. Oper. Res. **77**, 466–474 (1994)
5. Stevenson, D., Searchfield, G., Xu, X.: Spatial design of hearing aids incorporating multiple vents. Trends Hear. **18** (2014). doi:10.1177/2331216514529189
6. Wang, J.: Packing of unequal spheres and automated radiosurgical treatment planning. J. Comb. Optim. **3**, 453–463 (1999)
7. Hifi, M., M'Hallah, R.: A literature review on circle and sphere packing problems: models and methodologies. Adv. Oper. Res. (2009). doi:10.1155/2009/150624
8. Lopez, C.O., Beasley, J.E.: A heuristic for the circle packing problem with a variety of containers. Eur. J. Oper. Res. **214**, 512–525 (2011)
9. Lopez, C.O., Beasley, J.E.: Packing unequal circles using formulation space search. Comput. Oper. Res. **40**, 1276–1288 (2013)
10. Pinter, J.D., Kampas, F.J.: Nonlinear optimization in Mathematica with MathOptimizer Professional. Math. Educ. Res. **10**, 1–18 (2005)
11. Akeb, H., Hifi, M.: Solving the circular open dimension problem using separate beams and look-ahead strategies. Comput. Oper. Res. **40**, 1243–1255 (2013)
12. Birgin, E.G., Gentil, J.M.: New and improved results for packing identical unitary radius circles within triangles, rectangles and strips. Comput. Oper. Res. **37**, 1318–1327 (2010)
13. Stoyan, Y.G., Yaskov, G.N.: Packing congruent spheres into a multi-connected polyhedral domain. Int. Trans. Oper. Res. **20**, 79–99 (2013)
14. Bennel, J.A., Olivera, J.F.: A tutorial in irregular shape packing problems. J. Oper. Res. Soc. **60**, 93–105 (2009)
15. Toledo, F.M.B., Carravilla, M.A., Ribero, C., Oliveira, J.F., Gomes, A.M.: The dotted-board model: a new MIP model for nesting irregular shapes. Int. J. Prod. Econ. **145**, 478–487 (2013)
16. Galiev, S.I., Lisafina, M.S.: Linear models for the approximate solution of the problem of packing equal circles into a given domain. Eur. J. Oper. Res. **230**, 505–514 (2013)
17. Litvinchev, I., Ozuna, L.: Packing circles in a rectangular container. Paper presented at the 1st international congress on logistics and supply chain, Mexican Institute of Transportation, Queretaro, Mexico, 24–25 October 2013

18. Litvinchev, I., Ozuna, L.: Integer programming formulations for approximate packing circles in a rectangular container. Math. Probl. Eng. (2014). Article ID 317697, doi:10.1155/2014/317697
19. Litvinchev, I., Ozuna, L.: Approximate packing circles in a rectangular container: valid inequalities and nesting. J. Appl. Res. Technol. **12**, 716–723 (2014)
20. Litvinchev, I., Infante, L., Ozuna, L.: Approximate circle packing in a rectangular container: integer programming formulations and valid inequalities. Lect. Notes Comput. Sci. **8760**, 47–61 (2014)
21. Beasley, J.E.: An exact two-dimensional non-guillotine cutting tree search procedure. Oper. Res. **33**, 49–64 (1985)
22. Burkard, R., Dell'Amico, M., Martello, S.: Assignment Problems, Revised Reprint. SIAM (2012)
23. Wolsey, L.A.: Integer Programming. Wiley, New York (1999)
24. ILOG CPLEX, Mathematical programming optimizers. Version 12.6 (2013)
25. Bortfeldt, A., Wäscher, G.: Constraints in container loading—a state-of-the-art review. Eur. J. Oper. Res. **229**, 1–20 (2013)
26. Wang, W., Wang, H., Dai, G., Wang, H.: Visualization of large hierarchical data by circle packing. CHI '06 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, April 22–27, Montreal, Canada, pp. 517–520 (2006)
27. George, J.A.: Multiple container packing: a case study of pipe packing. J. Oper. Res. Soc. **47**, 1098–1109 (1996)
28. Pedroso, J.P., Cunha, S., Tavares, J.N.: Recursive circle packing problems. Int. Trans. Oper. Res. (2014). doi:10.1111/itor.12107
29. Litvinchev, I., Infante, L., Ozuna, L.: LP-based heuristic for packing circular-like objects in a rectangular container. Math. Probl. Eng. (to appear)
30. Litvinchev, I., Tsurkov, V.: Aggregation in Large Scale Optimization. Kluwer, Boston (2003)
31. Burke, E.K., Hellier, R.S., Kendall, G., Whitwell, G.: Irregular packing using the line and arc no-fit polygon. Oper. Res. **58**, 948–970 (2010)
32. Litvinchev, I., Infante, L., Ozuna, L.: Packing circular-like objects in a rectangular container. J. Comput. Syst. Sci. Int. **54**, 259–267 (2015)