Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 159 (2004) 430–448
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Discrete Optimization

A simulated annealing approach for the circular cutting problem

Mhand Hifi [a], Vangelis Th. Paschos [b,*], Vassilis Zissimopoulos [c]

[a] *LaRIA, Université d'Amiens, CERMSEM, Université Paris 1, France*
[b] *LAMSADE, Université Paris-Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France*
[c] *Department of Informatics, University of Athens, Athens, Greece*

Received 22 November 1999; accepted 5 May 2003
Available online 28 September 2003

Abstract

We propose a heuristic for the constrained and the unconstrained circular cutting problem based upon simulated annealing. We define an energy function, the small values of which provide a good concentration of the circular pieces on the left bottom corner of the initial rectangle. Such values of the energy correspond to configurations where pieces are placed in the rectangle without overlapping. Appropriate software has been devised and computational results and comparisons with some other algorithms are also provided and discussed.

Keywords: Cutting; Heuristics; Optimization; Simulated annealing

1. Introduction

The cutting problem consists of cutting large plates into smaller pieces in such a way as to optimize a given objective. Different approaches and constraints for the problem have been considered in the literature over the last 30 years (see [7,8,17]). In this work we treat the problem of cutting a rectangular plate R of dimensions (L, H) into as many circular pieces of n different types of values v_ℓ and radii r_ℓ, $r_\ell \leqslant \min\{L/2, H/2\}$, $\ell = 1, \ldots, n$.

We say that the n-dimensional vector $(w_1, \ldots, w_n)$ of integer and nonnegative numbers corresponds to a feasible cutting pattern, if it is possible to produce w_ℓ pieces of type ℓ, $\ell = 1, \ldots, n$, in the initial rectangular plate without overlapping. The circular cutting problem (shortly CC) consists of determining the cutting pattern with the maximum value, i.e.,

* Corresponding author. Tel.: +33 -1-44-05-45-82; fax: +33-1-44-05-40-91.
E-mail addresses: hifi@univ-paris1.fr (M. Hifi), paschos@lamsade.dauphine.fr (V.Th. Paschos), vassilis@di.uoa.gr (V. Zissimopoulos).

0377-2217/$ - see front matter
doi:10.1016/S0377-2217(03)00417-X

$$\mathrm{CC} = \begin{cases} \max & \sum_{\ell=1}^{n} v_\ell w_\ell \\ \text{subject to} & (w_1, \ldots, w_\ell) \text{ corresponds to a feasible cutting pattern.} \end{cases} \quad (1)$$

In our study, the quantity v_ℓ appearing in expression (1) represents the surface of a piece of type ℓ. So, maximization of the objective function becomes maximization of the rectangular plate's surface covered by the $\sum_{\ell=1}^{n} w_\ell$ pieces, or equivalently, minimization of the plate's surface not covered by them.

The formulation of expression (1) introduces what in the sequel is called *unconstrained* CC. When a restriction on the maximum number of pieces to be produced has to be satisfied, we then have another version of CC, the *constrained* one. Let $b = (b_1, \ldots, b_n)$ be a vector where b_ℓ, $\ell = 1, \ldots, n$, represents the maximum number of times that piece of type ℓ can appear in a feasible cutting pattern. Then, constrained CC is expressed as above with the additional constraint $(w_1, \ldots, w_n) \leqslant (b_1, \ldots, b_n)$. In our study, we consider that value v_ℓ represents the surface of the ℓth type of pieces. Note that the unconstrained version can be seen as a particular case of the constrained one considering $b_\ell = \lfloor (L/(2r_\ell))(H/(2r_\ell)) \rfloor$.

The (un)constrained CC problem is NP-complete since it is a generalization of the rectangular cutting stock problem, which in turn is a generalization of the famous one-dimensional knapsack problem. To our knowledge, very few works deal with CC problems. Most of them deal with packing circles of the same size into rectangles and the proposed algorithms strongly depend on the "single-size" constraint [6,9]. We also quote the work of [5] (subsequent to the first version of our paper [14]), pointed out very recently by one of the anonymous referees, where a simulated annealing (SA) method is developed for cylinder packing. The resolution of this problem in [5] consists in determining a circular cutting pattern involving circular pieces of the same size, in such a way that the number of pieces used is maximized. As far as we know, the only work dealing with the problem of packing circular pieces of different sizes is the one developed in [10], where the heuristics proposed are based upon several solution building rules simulating the packing operation; the best ones are a quasi-random algorithm and a genetic algorithm. A very interesting work has been presented by Stoyan and Yaskov [16] for a kind of "dual" of CC problem, the strip-packing problem, where we are given a set of different circular pieces and we try to put them in a rectangle, without overlapping, in such a way that some criterion, either its dimension L on axis $\vec{0}_x$, or its dimension H on axis $\vec{0}_y$, or both of them, are minimized. In [16], H is fixed and L is minimized by constructing a mathematical model and an exact method of searching a local extremum. Finally, another version of circular strip-packing problem has been theoretically studied by Graham and Lubachevsky [12]. They consider a fixed number of disks of the same diameter and search *for the side length of the smallest square that contains the disk centers*.

As we show in Section 3, very interesting heuristics, developed for solving other types of cutting problems (for example, rectangular or irregular cutting/packing, [2,3,13]), can be satisfactorily adapted to deal with CC. In general, these approaches construct rectangular blocks of pieces with respect to b_ℓ, $\ell = 1, \ldots, n$; next, they select some blocks that do not exceed a fixed threshold of waste; finally, a further selection of the already pre-selected blocks is performed by dynamic programming or artificial intelligence procedures.

In this paper we propose a SA algorithm for CC and compare it with several approaches of the literature. The rest of the paper is organized as follows. We first present the problem by means of an energy function E to be minimized. Next, by applying the SA scheme, we reach low values of E corresponding, in general, to solutions of good quality. Finally, after having specified the overall CC algorithm, we perform some experimental and comparative studies. These studies concern both the feasibility of the method and its ability in computing satisfactory solutions.

As we have already mentioned, we consider that the contributed profit of each piece is its surface. Therefore, the quality of a solution is measured by the surface covered by it (in other words, the greater the rectangle's surface covered, the better the solution). Our method can be outlined as follows. We start from introducing a large number of pieces into the rectangle (this is our initial solution). This number is sufficiently large to cover rectangle's surface but the solution so obtained is infeasible in the sense that some of

the pieces introduced either intersect mutually, or intersect rectangle's edges. Then, we try to eliminate overlapping by dropping some of the pieces out of the solution. Let us note that the method could generate a small percentage of infeasible cutting patterns in the sense that the corresponding pieces of the pattern intersect mutually. In this case, we apply a post-processing (discussed in Section 2.5) of the infeasible CC-solutions in order that the solutions finally computed are all feasible.

2. A simulated annealing algorithm for the circular cutting

The main principle of the SA algorithm was first described in [15] and has been successfully used for solving optimization problems. An extensive bibliography about this use can be found in [1,4]. The SA algorithm can be outlined as follows:

1. fix initial temperature (T_0) and the number of neighboring solutions M_0 generated at constant temperature (usually called length of the Markovian chain);
2. set $k = 0$ and determine the first initial solution ($p = p_{\text{start}}$) and its energy function value ($E_{p_{\text{start}}}$);
3. `repeat` M_k times:
   $q = \text{neighbor}(p)$;
   `if` $\Delta_{qp} = E_q - E_p \leqslant 0$ `then` set $p = q$; `else if` $\exp(-\Delta_{qp}/T_k) > \text{random}\ [0,1]$ `then` set $p = q$;
4. increment(k), compute M_k and T_k (reduced temperature);
5. `repeat` steps 3–5, `until` exit criterion.

In this algorithm several decisions have to be taken. Some of them are specific to the problem one deals with: how to generate an initial configuration; how to generate the neighbor to a configuration; how to calculate the difference Δ_{qp}? Other decisions are internal to the annealing process: how to fix the initial temperature; how to determine the length M_k and the new temperature T_k; what is the exit criterion? These decisions are usually empirically taken.

In what follows, we shall see how one can use the principle of the SA algorithm to approximately solve CC.

2.1. Some definitions and notations

We can consider that the initial rectangle lies in the plane and its four vertices can be represented by the coordinates: $(0,0)$, $(L,0)$, $(0,H)$ and (L,H) (L is on the axis $\vec{0}_x$ and H on the axis $\vec{0}_y$), respectively. Note that we consider $L \geqslant H$ (by performing a rotation if necessary). Finally, recall that for each $\ell \in \{1,\ldots,n\}$ one can consider more than one (at most b_ℓ) pieces of type ℓ.

Each piece i is represented by the coordinates (x_i, y_i) of its center; moreover we use the following definitions:

- a piece i feasibly lies to the interior of R if it lies in the interior of R, or if it feasibly touches its borders, i.e., if $(r_i, r_i) \leqslant (x_i, y_i) \leqslant (L - r_i, H - r_i)$;
- a piece i lies to the exterior of R if one of the following eight conditions holds:
  (i) $0 \leqslant x_i \leqslant L$ and $y_i < -r_i$,
  (ii) $0 \leqslant x_i \leqslant L$ and $y_i > H + r_i$,
  (iii) $x_i < -r_i$ and $0 \leqslant y_i \leqslant H$,
  (iv) $x_i > L + r_i$ and $0 \leqslant y_i \leqslant H$,
  (v) $x_i > L$ and $y_i \leqslant 0$ and $(x_i - L)^2 + y_i^2 > r_i^2$,
  (vi) $x_i \geqslant L$ and $y_i \geqslant H$ and $(x_i - L)^2 + (y_i - H)^2 > r_i^2$,

(vii) $x_i \leqslant 0$ and $y_i \geqslant H$ and $x_i^2 + (y_i - H)^2 > r_i^2$,
(viii) $x_i \leqslant 0$ and $y_i \leqslant 0$ and $x_i^2 + y_i^2 > r_i^2$;

- a piece i intersects R if one of the following eight conditions holds:
  (i) $0 \leqslant x_i \leqslant L$ and $|y_i| \leqslant r_i$,
  (ii) $0 \leqslant x_i \leqslant L$ and $H - r_i \leqslant y_i \leqslant H + r_i$,
  (iii) $0 \leqslant x_i \leqslant r_i$ and $r_i \leqslant y_i \leqslant H - r_i$,
  (iv) $L - r_i \leqslant x_i \leqslant L$ and $r_i \leqslant y_i \leqslant H - r_i$,
  (v) $x_i \leqslant 0$ and $y_i \leqslant 0$ and $x_i^2 + y_i^2 \leqslant r_i^2$,
  (vi) $x_i \geqslant L$ and $y_i \leqslant 0$ and $(x_i - L)^2 + y_i^2 \leqslant r_i^2$,
  (vii) $x_i \leqslant 0$ and $y_i \geqslant H$ and $x_i^2 + (y_i - H)^2 \leqslant r_i^2$,
  (viii) $x_i \geqslant L$ and $y_i \geqslant H$ and $(x_i - L)^2 + (y_i - H)^2 \leqslant r_i^2$.

Moreover, we denote by S_0 the surface of R, by S_r the set $\{(x, y) : x \geqslant L \text{ and } 0 \leqslant y \leqslant H\}$, and by $\bar{S}$ the set $\mathbb{R}^2 \setminus (S_0 \cup S_r)$; finally, $S_i = v_i$ represents the surface of piece i.

2.2. An energy function associated with the circular cutting

A common way for solving optimization problems is to devise methods (exact or heuristics) improving a feasible initial solution. As we have already mentioned in the outline of our method, our approach is somewhat different. We first produce a configuration generally corresponding to an infeasible solution, the objective value of which is "better" (greater) than the optimal value for CC. We then try to render the configuration produced feasible. In other words, we first favor, as we have already mentioned, the introduction of a maximum number of pieces into R (even allowing some mutual intersections of the pieces), and next we rearrange the pieces introduced (even eliminating some of them) to omit mutual intersections. The feasible configuration so obtained is our final solution. The energy function we will propose in the sequel reflects the thought process just described, and informally, is based upon the following requirements:

(i) situations where:
  (a) two pieces intersect mutually, or
  (b) a piece intersects R, or
  (c) a piece lies in the exterior of R (intersects $\mathbb{R}^2 \setminus S_0$), have to be avoided in a feasible pattern;
(ii) the greater the number of pieces feasibly placed in R, the higher the usage (covering) of the the surface of R.

Our objective is to build an energy function E, the optima (global or local) of which correspond to satisfactory CC-solutions. In what follows, let $m = \sum_{\ell=1}^{n} b_\ell$ be an upper bound on the number of pieces that can be cut. We consider that $i \in \{1, \ldots, m\}$ represents the index of the ith piece and $i = 0$ represents the index associated with $\bar{S}$ or S_r. We also denote by $A(S)$ the area of S.

In order to introduce a maximum number of pieces in the rectangle, we "strain" them to be concentrated to the bottom-left corner of R. A measure of how close to the bottom-left corner of R a piece i lies is the value of the expression $C_i = (r_i(|y_i - r_i| + |x_i - r_i|))$, where if $(x_i, y_i) = (r_i, r_i)$, then $C_i = 0$, while if $(x_i, y_i) \neq (r_i, r_i)$, the non-zero value of $(|y_i - r_i| + |x_i - r_i|)$ is amplified by its multiplication by r_i. In other words, informally, the fact that a piece i is far from the bottom-left corner of R induces a large value for C_i, and consequently, for the energy function.

Introduction of a large number of pieces in R fulfills requirement (ii), but is contradictory with requirement (i) since it may produce intersections between pieces in R. Here, for any such illegal intersection, the measure of "how much requirement (i) is violated" is the area of the intersection. The areas induced by the problematic configurations described in requirements (ia), (ib) and (ic) are the following:

1. if a piece j, $j = 1, \ldots, m$, intersects i and if: both i and j lie in the interior of R (case 1.1), or both i and j intersect R (case 1.2), or only j intersects R (case 1.3), then the corresponding areas are

$$\begin{cases} A(S_i \cap S_j) & i \neq 0,\ j > i \quad \text{for cases (1.1) and (1.2)}, \\ A(\bar{S} \cap S_j) \quad \text{and} \quad A(S_r \cap S_j) & i = 0,\ j \neq 0 \quad \text{for case (1.3)}; \end{cases}$$

2. if a piece j, $j = 1, \ldots, m$, lies in the exterior of R ($S_j \cap S_0 = \emptyset$), then we have $i = 0$ and the corresponding areas are

$$\begin{cases} A(S_j) & S_j \cap S_r = S_j, \\ A(S_j) & S_j \cap \bar{S} = S_j, \\ A(\bar{S} \cap S_j) \quad \text{and} \quad A(S_r \cap S_j) & S_j \cap (\bar{S} \cup S_r) = S_j. \end{cases}$$

Since requirements (i) and (ii) are contradictory, a trade-off between them is performed by using penalty parameters. We use four such parameters: p_1 penalizing non-empty intersections between pieces, p_2 penalizing intersections of the pieces with $\bar{S}$, p_3 penalizing intersections of the pieces with S_r, and p_4 penalizing the distance from the axes $\vec{0}_x$ and $\vec{0}_y$ (bottom-left corner of R). Let us now show how we use them in order to produce the mathematical expression of the energy function proposed.

We first consider requirement (i) and the different areas described in items 1 and 2 just above. Let $E_{ij}^{(1)}$ be the basic component of the term of E dealing with requirement (i). Then:

- dealing with 2, we set

$$E_{ij}^{(1)} = \begin{cases} p_1 A(S_i \cap S_j) & i \neq 0,\ j > i \quad \text{for cases (1.1) and (1.2)}, \\ p_2 A(\bar{S} \cap S_j) + p_3 A(S_r \cap S_j) & i = 0,\ j \neq 0 \quad \text{for case (1.3)}; \end{cases} \tag{2}$$

- while, dealing with 2, we set

$$E_{0j}^{(1)} = \begin{cases} p_3 A(S_j) & S_j \cap S_r = S_j, \\ p_2 A(S_j) & S_j \cap \bar{S} = S_j, \\ p_2 A(\bar{S} \cap S_j) + p_3 A(S_r \cap S_j) & S_j \cap (\bar{S} \cup S_r) = S_j. \end{cases} \tag{3}$$

In summary, the term of E dealing with requirement (i) is $E^{(1)} = \sum_{i=0}^{m-1} \sum_{j>i} E_{ij}^{(1)}$. Let now $E_i^{(2)}$ be the basic component of the term of E dealing with requirement (ii). Then,

$$E_i^{(2)} = p_4(r_i(|y_i - r_i| + |x_i - r_i|)) \tag{4}$$

and the term of E dealing with this requirement is $E^{(2)} = \sum_{i=1}^{m} E_i^{(2)}$. Finally,

$$E = E^{(1)} + E^{(2)} = \sum_{i=0}^{m-1} \sum_{j>i} E_{ij}^{(1)} + \sum_{i=1}^{m} E_i^{(2)}. \tag{5}$$

We now show how to adjust the parameters involved in expressions (2)–(4) in order to obtain configurations of the form "a maximum number of pieces in the strip defined by $S_0 \cup S_r$ without overlapping" characterized by low values of E. We concentrate a maximum of pieces in the strip $S_0 \cup S_r$ (in view of their introduction in R) by allowing some overlapping. For this, we have to strain the pieces to avoid placing them in $\bar{S}$. We therefore consider a large value for p_2. In this way, since E (expression (5)) is to be minimized, the terms with factor p_2 must be the least possible in number or/and to have values close to 0. Since we have decided to allow some overlapping, we consider $p_1 < p_2$.

In order to avoid overlapping inside R, we enable a certain shifting to the right (even if this shifting will produce the introduction of some pieces from S_0 to S_r); $p_3 < p_1$ is a possibility for this. Thus, finally, $p_3 < p_1 < p_2$, and the term $\sum_{i=0}^{m-1} \sum_{j>i} E_{ij}^{(1)}$ encourages the introduction of pieces in R, avoiding both overlapping between pieces and overflow of pieces with respect to the edges of R.

We now complete the adjustment by trying to further strain the pieces to be placed in the strip $S_0 \cup S_r$ in order to ensure that a maximum number of pieces will finally be introduced in the strip without overlapping. This can be obtained using parameter p_4. At the initial step of the algorithm, we consider a value for p_4 as large as the one for p_2 (by allowing high initial-energy values). In other words, we attribute the same importance to both the overlapping elimination and the concentration of the pieces to the bottom-left corner of R. Progressively, the value of p_4 will be reduced. This means that we accept the shifting of some pieces from S_0 to S_r in order to avoid overlapping into S_0. So, if p_4 becomes close to 0 and if no overlapping is produced, then the value of the energy becomes small.

2.3. Initial solution

We have already mentioned that, initially, all the $m = \sum_{\ell=1}^{n} b_\ell$ pieces are randomly placed on the plane. In order to be more efficient in time, we try to reduce the "randomness" of the initial placement by solving the following subset sum problem:

$$\mathrm{KP} = \begin{cases} \max & \sum_{\ell=1}^{n} A(S_\ell) z_\ell \\ & \sum_{\ell=1}^{n} A(S_\ell) z_\ell \leqslant LH, \\ & z_\ell \leqslant b_\ell, \ \ell = 1, \ldots, n, \\ & z_\ell \in \mathbb{N}^+, \end{cases}$$

the solution of which gives an upper bound for the value of the optimal solution of CC. We recall that the unconstrained case can be reduced to the constrained one by considering bounds $b_\ell = \lfloor (L/(2r_\ell))(H/(2r_\ell)) \rfloor$.

Let z_ℓ^*, $\ell = 1, \ldots, n$, be the solution of KP. We start from the hypothesis that at most z_ℓ^* pieces of type ℓ, $\ell = 1, \ldots, n$, will be placed in R without overlapping. For each ℓ, we consider a partition of the set of pieces of type ℓ into two subsets B_ℓ and $\overline{B}_\ell$, of cardinalities z_ℓ^* and $b_\ell - z_\ell^*$, respectively.

We place randomly the $b_\ell - z_\ell^*$, pieces of $\overline{B}_\ell$, $\ell = 1, \ldots, n$, in S_r by allowing overlapping. We can observe that since we allow overlapping and $A(S_r)$ is infinite, we can place all these pieces here. Procedure 1 just below places the pieces of $\cup_{\ell=1}^{n} \overline{B}_\ell$ in S_r (allowing overlapping).

Procedure 1

Step 1. set $L_1 = L$, $H_1 = H$, path $= 0$ and set $\mathscr{S} = \bigcup_{\ell=1}^{n} \overline{B}_\ell$;
Step 2. let $k \in \mathscr{S}$ be the piece of radius r_k and set $f_k = k$; set $(x_k, y_k) = (L_1 + r_k, H_1 - r_k)$;
Step 3.

```
repeat
1.    let (a randomly taken) k' ∈ S of radius r_k'; set (x_k', y_k') = (L_1 + r_k', y_k - r_k');
2.    set S = S \ {k'}, k = k';
until (S = ∅) or (y_k' ⩽ 0);
```

Step 4. `if` $\mathscr{S} \neq \emptyset$ `then` set path $= f_k$, $L_1 = L_1 + 2x_{f_k}$ and `goto` Step 2.

Next, we place (always randomly and in a greedy way) some pieces belonging to the different B_ℓ's into S_0 without allowing overlapping for them. The rest of the non-placed pieces of B_ℓ are randomly placed in $\overline{S}$. Procedure 2 places the pieces of $\bigcup_{\ell=1}^{n} B_\ell$ in S_0 and $\overline{S}$.

Procedure 2

Step 1. set $L_1 = 0$, $H_1 = 0$ and $\mathscr{S} = \bigcup_{\ell=1}^{n} B_\ell$;
Step 2. let $k \in \mathscr{S}$ be the piece of radius r_k; set $(x_k, y_k) = (L_1 + r_k, H_1 + r_k)$ and $H_{\text{limit}} = 2r_k$;

Step 3. `do`

1. let (a randomly taken) $k' \in \mathscr{S}$ of radius $r_{k'}$ be such that $(x_k + r_k + 2r_{k'}, H_1 + 2r_{k'}) \leqslant (L, H)$;
   `if` k' exists `then` set $(x_{k'}, y_{k'}) = (x_k + r_k + r_{k'}, H_1 + r_{k'})$ and `goto` 3.2; `else goto` Step 4;
2. set $H_{\text{limit}} = \max\{H_{\text{limit}}, H_1 + 2r_{k'}\}$, $\mathscr{S} = \mathscr{S} \setminus \{k'\}$ and $k = k'$;
   `if` $\mathscr{S} = \emptyset$ `then goto` Step 5; `else goto` 3.1;

Step 4.

`if` $\mathscr{S} \neq \emptyset$ `then` set $L_1 = 0$, $H_1 = H_{\text{limit}}$ and $H_{\text{limit}} = 0$;
`if` $\exists k \in \mathscr{S}$ such that $H_1 + 2r_k \leqslant H$ `then`
set $(x_k, y_k) = (L_1 + r_k, H_1 + r_k)$ and `goto` Step 3; `else goto` Step 5;

Step 5. `if` $\mathscr{S} \neq \emptyset$ `then` place all pieces of $\mathscr{S}$ on $\overline{S}$ around the three edges of the initial plate, i.e., $(0,0) \leqslant (x,y) \leqslant (0,H)$, $(0,0) \leqslant (x,y) \leqslant (L,0)$ and $(0,H) \leqslant (x,y) \leqslant (L,H)$.

Procedures 1 and 2 give an initial placement of all the pieces. Later, if a local solution with good characteristics is obtained, then we try to use an intensification on its neighborhood in order to improve the quality of this solution. On the other hand, if the solution obtained cannot be improved after a certain number of iterations, then we try to use a diversified solution in order to "globally" perturbate the system. The intensification and diversification strategies are performed by considering the transformations (T1) to (T5) discussed in the following section.

2.4. Generation of a neighboring solution

Consecutive configurations of system's states are obtained by means of a number of transformations on positions of the pieces. In what follows, we describe these transformations.

For a piece i, we define its neighborhood as the set of pieces, the center of which lies into a fictive cycle of center (x_i, y_i) and radius $2r_i$.

2.4.1. Transformation (T1)

This transformation is performed either horizontally or vertically (in both cases parallel either to axis $\vec{0}_x$, or to axis $\vec{0}_y$). We randomly choose a piece i; for a piece j neighbor of i, we define $\delta_j^x = |x_i - x_j| - (r_i + r_j)$, its horizontal distance from i, and $\delta_j^y = |y_i - y_j| - (r_i + r_j)$, its vertical distance from i (Fig. 1(a) and (b)). Transformation (T1) consists of bringing piece $k \in \arg\min_{1 \leqslant j \leqslant m}\{\delta_j^x, \delta_j^y\}$, k being a neighbor of i closer to i.

2.4.2. Transformation (T2)

This transformation is performed diagonally with respect to axes $\vec{0}_x$ and $\vec{0}_y$ (Fig. 2(a) and (b)). Given two pieces i and j, we define their diagonal distance $d = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2}$. As previously, we try to move

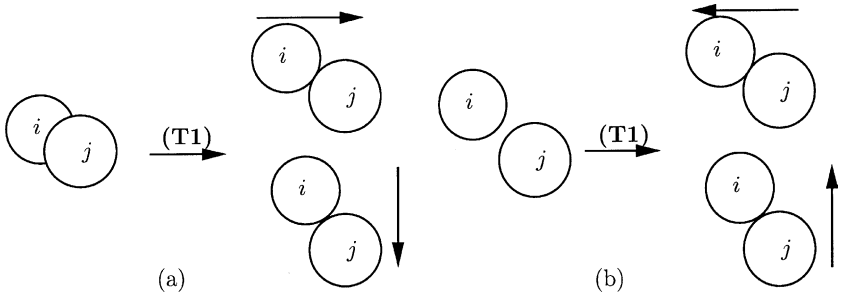

Fig. 1. (a) Eliminating overlapping, and (b) bringing j closer to i via transformation (T1).

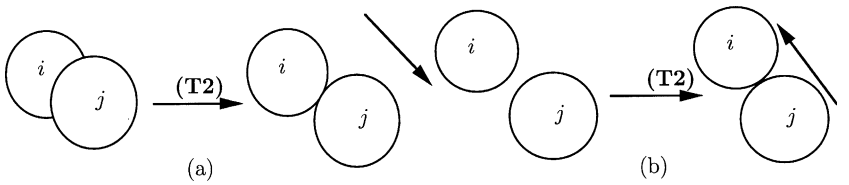

Fig. 2. (a) Eliminating overlapping, and (b) bringing j closer to i via transformation (T2).

piece j, lying on the neighborhood of i, in such a way that the two pieces touch each other and, moreover, the center of j always lies on the straight line defined by the points (x_i, y_i) and (x_j, y_j). This movement can be seen as two simultaneous movements, one parallel to axis $\vec{0}_x$ by a distance δx, and the other parallel to axis $\vec{0}_y$ by a distance δy. These two distances are defined by

$$(\delta x, \delta y) = \begin{cases} \left(\Delta x\left(1 - \frac{r_i + r_j}{d}\right), \Delta y\left(1 - \frac{r_i + r_j}{d}\right)\right), & d \geqslant r_i + r_j, \\ \left(\Delta x\left(\frac{r_i + r_j}{d} - 1\right), \Delta y\left(\frac{r_i + r_j}{d} - 1\right)\right), & \text{otherwise,} \end{cases}$$

where $\Delta x = |x_i - x_j|$ and $\Delta y = |y_i - y_j|$.

2.4.3. Transformation (T3)

Let us consider two circular pieces i and j touching each other, and suppose that i lies on the left of j. Let us denote by (x_t, y_t) the coordinates of the point where a tangent of j, vertical to the axis $\vec{0}_x$, touches j. Transformation (T3) consists of performing a symmetric displacement of j with respect to the axis represented by the straight line vertical to $\vec{0}_x$ containing point (x_i, y_i), and then a right shifting of the two pieces by an horizontal distance $|x_t - x_i - r_i|$ (Fig. 3). The meaning of this transformation is that if i has important overlapping surfaces on its left, and moreover, $r_j < r_i$, by (T3) we can reduce overlapping. On the other hand, if $r_j > r_i$, and there exist some "holes" (waste) on the left of i, then (T3) will reduce their surface.

2.4.4. Transformation (T4)

Let us consider a piece j into S_0 and tangent (i) either to straight line $y = H$, (ii) or to straight line $x = L$. Let us also consider a piece i tangent to j and lying out of S_0. Then, transformation (T4) consists of a symmetric displacement of i and j with respect to the straight line $y = H$ in case (i) (Fig. 4), or $x = L$ in case (ii). This transformation allows new pieces to be introduced into S_0.

2.4.5. Transformation (T5)

Consider a piece i lying on S_0 and a piece j out of S_0, the radii of which verify $0 < |r_i - r_j| / \max\{r_i, r_j\} \leqslant \beta$ for a positive constant β. Then transformation (T5) consists of an interchanging between i and j. The

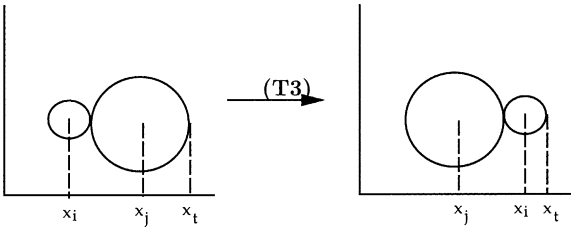

Fig. 3. Transformation (T3).

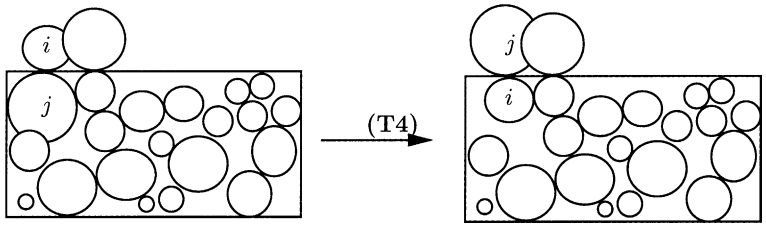

Fig. 4. Transformation (T4).

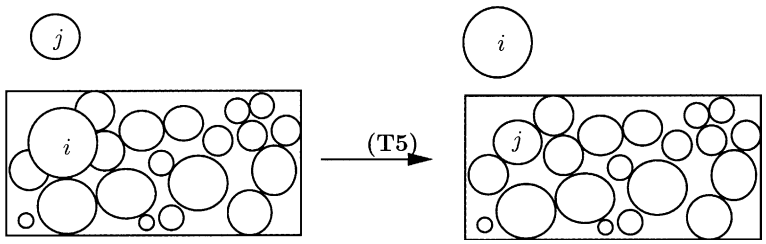

Fig. 5. Transformation (T5).

meaning of (T5) is that if $r_i > r_j$, then the performed interchanging will reduce the overlapping into S_0, while, in the opposite case, the waste will be reduced. Fig. 5 gives an idea of such a transformation for $\beta = 2$ and for the case $r_i > r_j$; for the case $r_i < r_j$, it suffices to read the figure from the right to the left.

Transformations (T1) to (T3) represent a kind of "local re-arrangement" in the interior of S_0 in order to obtain feasible solutions corresponding to local minima of E. These transformations are applied on the pieces lying to the neighborhood associated to a fixed piece on S_0. On the other hand, transformations (T4) and (T5) cause a kind of "perturbation" leading, eventually, smaller local minima, corresponding to better quality solutions. Finally, let us note that at least one among transformations (T1), (T2), or (T3) is always applicable; for example, if (T1) does not work, then there exists a piece j tangent to i, and consequently, transformation (T3) can be applied.

2.5. Outline of the simulated annealing algorithm

The SA heuristic developed for CC is outlined by Algorithm 1. The default settings for it are:

M_i: the neighborhood of a fixed piece i, represented by the pieces lying to the fictive cycle of center (x_i, y_i) and radius $2r_i$;
δm: the length of the Markovian chain at a certain temperature;
global_counter: the counter (limited by δm) associated with the length of the Markovian chain for a given temperature;
local_counter: the counter computing, for a fixed piece i, the number of configurations considered in the neighborhood M_i;
local_find: limited by the number v means that the Markovian chain is cut if the energy function does not change after v consecutive configurations;
best_sol: the configuration corresponding to the best current solution.

Algorithm 1. The SA heuristic for CC.

1. fix the initial temperature T_0, set counter $k = 0$ and let $C_p = C_{p_{\text{start}}}$ be the initial solution obtained by solving KP and using Procedures 1 and 2; evaluate energy $E(p)$ (corresponding to C_p); set global_counter = 0 and local_find = 0;
2. execute the following steps:
   (a) let i be the index of a piece lying in R or intersecting R, and set local_counter = 0;
   `while` local_counter $\leqslant |M_i|$ `do`
   i. let C_q =neighbor(C_p)
   /* C_q is obtained by applying arbitrarily one of (T1), (T2) and (T3) */;
   ii. `if` $\Delta_{qp} \leqslant 0$ or $\min\{1, \exp(-\Delta_{qp}/T_k)\} >$ random $[0,1]$ `then`
   set $C_p = C_q$; update the best configuration best_sol (if necessary); set local_find = 0;
   `else` increment(local_find);
   iii. increment(local_counter);
   `if` local_find $\geqslant v$ or global_counter + local_counter $\geqslant \delta m$ `then` goto 3;
   (b) let C_q =neighbor(C_p)
   /* C_q is obtained by applying randomly one of (T4) and (T5) */
   (c) `if` $\Delta_{qp} \leqslant 0$ or $\min\{1, \exp(-\Delta_{qp}/T_k)\} >$ random $[0,1]$ `then`
   set $C_p = C_q$; update the best configuration best_sol (if necessary); set local_find = 0;
   `else` increment(local_find);
3. set global_counter = global_counter + local_counter;
   `if` global_counter $< \delta m$ and local_find $< v$ `then` goto 2;
4. increment(k); reduce the temperature T_k; set global_counter = 0 and local_find = 0;
5. `repeat` steps 3–5 `until` one of the exit criteria i or ii (below) is satisfied.

We denote by $E(p)$ the energy associated with configuration C_p and by $\Delta_{qp} = E(q) - E(p)$ the difference between the energies of the two consecutive configurations C_q and C_p. We now describe how we compute this quantity. Revisit expression (5) (the energy function E) and denote, for a configuration C_p, by $E^{(1)}(p)$ the term $\sum_{i=0}^{m-1} \sum_{j>i} E_{ij}^{(1)}$ and by $E^{(2)}(p)$ the term $\sum_{i=1}^{m} E_i^{(2)}$. We decompose Δ_{qp} into two terms, the former accounting for $E^{(1)}$ and the latter for $E^{(2)}$. So, $\Delta_{qp} = \Delta_{qp}^{(1)} + \Delta_{qp}^{(2)}$, with $\Delta_{qp}^{(1)} = E^{(1)}(q) - E^{(1)}(p)$, and $\Delta_{qp}^{(2)} = E^{(2)}(q) - E^{(2)}(p)$.

If the transformation considered uses two different pieces s and t ($s \neq t$, $s > t$), then $\Delta_{qp}^{(1)}$ yields:

$$\Delta_{qp}^{(1)} = \left[\sum_{\substack{i=0 \\ i\neq s \\ i\neq t}}^{m} t\left(E_{is}^{(1)}(q) - E_{is}^{(1)}(p)\right) + \left(E_{it}^{(1)}(q) - E_{it}^{(1)}(p)\right) \right] + \left(E_{st}^{(1)}(q) - E_{st}^{(1)}(p)\right)$$

$$= \Delta_{qp(st)}^{(1)} + \sum_{\substack{i=0 \\ i\neq s \\ i\neq t}}^{m} \left(\Delta_{qp(is)}^{(1)} + \Delta_{qp(it)}^{(1)}\right)$$

where the last equality is obtained by setting $\Delta_{qp(ij)}^{(1)} = E_{ij}^{(1)}(q) - E_{ij}^{(1)}(p)$.

The second term is computed in a similar way resulting: $\Delta_{qp}^{(2)} = \Delta_{qp(s)}^{(2)} + \Delta_{qp(t)}^{(2)}$, where $\Delta_{qp(i)}^{(2)} = E_i^{(2)}(q) - E_i^{(2)}(p)$, and finally,

$$\Delta_{qp} = \Delta^{(1)}_{qp(st)} + \left[\sum_{\substack{i=0 \\ i\neq s \\ i\neq t}}^{m} (\Delta^{(1)}_{qp(is)}. + \Delta^{(1)}_{qp(it)}) \right] + \Delta^{(2)}_{qp(s)} + \Delta^{(2)}_{qp(t)}.$$

The law of temperature's decrement is geometric and is defined by $T_k = aT_{k-1}$, $k \geqslant 1$ and $a < 1$. The two exit criteria used in Algorithm 1 are:

(i) a threshold for the temperature;
(ii) a threshold for the quantity $|E(\alpha) - (\sum_{r=1}^{\alpha} E(r)/\alpha)|$, where $E(\alpha)$ is the energy of the configuration best_sol and $E(r)$ is the energy of the last configuration produced for a temperature value T_r.

If $|E(\alpha) - (\sum_{r=1}^{\alpha} E(r)/\alpha)| < \epsilon$ for a positive constant ϵ, or if the threshold temperature is attained, then the algorithm stops and the solution corresponding to best_sol represents the final solution for the CC problem. The interpretation of criterion (i) is taken from the (original) conception of SA in physics. It means that if temperature attains some low limit, then no significant change can be expected for material's state. In other words, the material has attained a stable state. This means that no significant improvement of the configuration best_sol is to be expected. Consequently, the execution of Algorithm 1 can stop. Analogous is the interpretation of exit criterion (ii). It implies that, if for a number of new configurations, no significant improvement is obtained with respect to the solution corresponding to best_sol, then one can consider that this solution is quite satisfactory and, consequently, one can stop the algorithm.

In Table 1 the values chosen for the parameters used in Algorithm 1 are given. As we have already mentioned in Section 2.2, the coefficient p_4 was progressively reduced. For each temperature value T_k, $k = 0, 1, 2, \ldots$, $p_4 = 10^{-1}T_k$.

Solutions considered by Algorithm 1 are represented by a simple exploration of the domain corresponding to some feasible and infeasible solutions. Our computational results show that generally the final solutions obtained are feasible, but sometimes, Algorithm 1 may produce an unfeasible solution. In order that our method becomes completely operational, we have applied a simple and polynomial post-processing transforming an unfeasible configuration into a feasible one. We construct a graph whose vertices represent all of pieces lying on R (and not intersecting the edges of R); two such vertices are linked by an edge only if the corresponding pieces overlap each other; then, a feasible solution of CC corresponds to a maximal independent set of the so-constructed (conflict) graph. On this graph (let us denote it by $G = (V, E)$), we apply the following greedy procedure.

Table 1
The values for the parameters used in Algorithm 1

a[a]	T_0[b]	Temperature's threshold[c]	ϵ[d]	δm[e]	v
0.95	10	10^{-3}	10^{-3}	$3m$	$3m/2$

[a] The parameter in the law of the cooling schedule.
[b] The initial temperature.
[c] Exit criterion (ii).
[d] Exit criterion (ii).
[e] The parameter representing the length of the Markovian chains for each temperature.

Procedure 3
1. set $V' = \emptyset$;
2. `repeat`
   (a) sort the vertices of V in increasing degree order;
   (b) let v_0 be the minimum degree vertex of V; put v_0 in V'; set $V = V \setminus (\{v_0\} \cup \{x : v_0x \in E\})$ and $E = E \setminus \{v_0x : v_0x \in E\}$;
   `until` $V = \emptyset$;
3. `output` V'.

We are now well-prepared to give an overall specification of the algorithm proposed for CC, denoted by FSA in what follows.

Algorithm 2. Algorithm FSA.

1. `call` Algorithm 1 on the CC instance;
   let S be the solution obtained;
2. `if` S infeasible for CC `then` construct the conflict graph G;
   `else` set $S = F$ `and goto` step 4;
3. `call` Procedure 3 on G;
   let F be the solution obtained;
4. `output` F.

Procedure 3 is simple and very easy to implement. There exist other post-processing leading to better final solutions than Procedure 3. For example, one can consider any piece together with its surface. Then, she/he constructs a weighted conflict graph G, where any vertex is weighted by the surface of the corresponding circular piece. In this case, one can use an approximation algorithm for weighted independent set instead of Procedure 3. Another way, always under the modeling of the solution obtained by Algorithm 1 as a conflict graph, would be to solve the corresponding independent set problem by exact methods, weightening so the execution time of the overall method (Algorithm 2).

Here our purpose is to study the capacity of a proper SA algorithm to solve CC. For this reason, no particular care has been taken in using more elaborated post-processing. As we shall see in Section 3, the solutions produced are fairly satisfactory.

3. Computational studies

We have conducted two series of experiments. The former compares the SA-method developed with the algorithms of [2,3,13]. Experiments here deal with both the feasibility of Algorithm 1 and with comparisons with other efficient cutting-algorithms originally devised for other than circular shapes. These algorithms have been implemented by us; we have properly adapted them in order to run for circular shapes. Both our method and the implemented algorithms run on randomly generated CC-instances. The latter series of experiments compares our method with the one of [16] and with the (theoretical) packings presented in [12]. Data have been sent us by professor Y.G. Stoyan, for the comparisons with [16], or had been extracted by us, as described in Section 3.2.2, for the comparisons with [12]. All the experiments have been performed on a Data General 486.

3.1. First series of experiments

As already mentioned, CC algorithms have been presented in [6,9,10]; the ones of [6,9] deal with cycles of the same size. Unfortunately, we have not been able to be provided with numerical data for them, and what we have alluded in [6,9,10] does not allow us to implement the heuristics proposed without altering their basic spirit.

We so have adapted and implemented three of the best known cutting algorithms originally devised to place other than circular shapes (irregular or rectangular ones), and compared the results obtained with the ones provided by Algorithm 2 (algorithm FSA). For the constrained CC-case, we have considered the algorithms of Beasley [3], and of Albano and Sapuppo [2], while for the unconstrained case the algorithm of Haims and Freeman [13].

Beasley's algorithm [3] is originally an exact algorithm for non-guillotinable rectangular cutting problems. Following the same process, we have modified it in order to solve CC approximately. Here is an outline of this modified version that we have devised and implemented:

1. apply the first step of the algorithm of [13] in order to solve a rectangular cutting problem (to construct rectangular blocs) without violating the upper bounds b_ℓ, $\ell = 1, \ldots, n$;
2. for each of the obtained rectangles R', solve CC in R' by applying Lagrangian relaxation as a heuristic for obtaining an initial feasible CC-solution (used as a lower bound) and a global upper bound for CC (obtained by considering specified Lagrangian multipliers);
3. improve the best current feasible solution by running a depth-first branch-and-bound procedure (stop if 10^6 nodes are created).

The modified version of Albano–Sapuppo's algorithm (originally devised in [2]) is as the previous one, except for step 2, where Lagrangian relaxation is matched with a best-first-search heuristic introducing displacements and rotations of the circular pieces.

Haims and Freeman's algorithm [13] performs a grouping of pieces (irregular shapes) into small rectangles. Next, it uses dynamic programming techniques (that can be seen as a modified version of Gilmore and Gomory's algorithm [11]) to place the small rectangles into bigger ones. We shortly outline its modification:

1. circular pieces are embedded (singly or by two or three) into minimum area rectangles (in order to minimize waste) called modules;
2. using dynamic programming, these modules are then further packed and enclosed into new (minimum area) rectangles to yield new modules and so on; the so-produced new modules are retained if they improve the solution;
3. the procedure just described ends when a module coincides with R.

We use this last algorithm for the unconstrained case, because it seems better adapted than the ones of [2,3] when the bounds b_i are quite large.

3.1.1. Problem generation

Our computational results were conducted on three groups of instances G1, G2 and G3. Each group contains 80 random instances generated as follows. The number of types of pieces for each group is randomly taken in the integer intervals $[5, 30]$, $[10, 70]$ and $[30, 100]$, respectively. The dimensions L and H of the initial rectangle are uniformly taken from $[50, 100]$ for the instances of G1, $[100, 150]$ for the ones of G2, and $[150, 250]$ for the ones of G3. The radius of each type ℓ of pieces is defined by $r_\ell = (1/2)\lceil 4\pi_\ell/\pi \rceil$, where $\pi_\ell = \gamma_\ell \min\{L, H\}$ with γ_ℓ randomly taken in the interval $]0, 1/3]$, $\ell = 1, \ldots, n$, and the value of π was setting

equal to 3.14. Finally, the number of each type of piece to be cut b_ℓ is randomly taken from $[1, \min\{10, \lfloor (L/(2r_\ell))(H/(2r_\ell)) \rfloor\}]$, $\ell = 1, \ldots, n$.

We have considered both constrained and unconstrained cases of the CC problem. The groups G1, G2 and G3 represent the instances with the bounds b_ℓ, $\ell = 1, \ldots, n$. For the unconstrained case, bounds b_ℓ are represented by the corresponding quantities $\lfloor (L/(2r_\ell))(H/(2r_\ell)) \rfloor$. Unconstrained and constrained versions of CC were tested separately.

3.1.2. The experimental results

When using meta-heuristics to solve optimization problems, it is well-known that different parameter settings for the method (e.g., slower cooling schedule, increasing of the neighborhood, etc.) lead to results of variable quality. Here also, a different adjustment of method's parameters would lead to a higher percentage of feasible solutions. But this "better" adjustment would lead to heavier execution time requirements. The set of values chosen in our experiment represents a satisfactory trade-off between solution quality and execution time.

3.1.2.1. On the feasibility of Algorithm 1. The percentage of feasible solutions produced by the SA algorithm (Algorithm 1) is shown in Table 2. As one can see, the larger the size of the instances, the greater the percentage of infeasible solutions. This is expected since, for the instances of large size, there is a big number of types of pieces that implies a great total number m of pieces. The energy of the system has more local minima that means that the SA method can generate a large number of infeasible solutions. This remark applies also to the unconstrained case where, in addition, each type contains more pieces than in the constrained one since the bound associated with each type of pieces is represented by the natural bound $\lfloor (L/2r_\ell)(H/2r_\ell) \rfloor$. This last fact draws an explanation for the slight superiority of the constrained case to the unconstrained one.

3.1.2.2. Comparisons with the methods of Beasley, Albano–Sapuppo, and Haims–Freeman. We have already mentioned that we treat a problem that consists of maximizing the surface covered by a cutting pattern. The pattern is represented by a vector $(x_1, \ldots, x_n)$, where x_ℓ is the number of times that type ℓ appears in this solution. Table 3 shows the percentages of the surface covered by the different methods (i.e., the surface covered, supposing that the surface of R is of 100 units); the symbol * means that the corresponding

Table 2
Percentage of the feasible solutions (output of Algorithm 1)

The CC problem	G1	G2	G3	Average over groups
Constrained version	83.75	76.25	75	79.33
Unconstrained version	70.63	67.5	61.50	66.54
Average over versions	77.19	71.875	68.25	72.935

Table 3
Percentage of the surface covered by the different algorithms (the output considered is the one of Algorithm 2)

Group	Constrained case			Unconstrained case	
	FSA	Algorithm of [3]	Algorithm of [2]	FSA	Algorithm of [13]
G1	69.32	63.25	63.78	72.07	68.17
G2	73.49	65.19	65.99	80.51	74.43
G3	77.37	*	67.68	83.32	*
Average	73.39	64.22	65.82	78.63	71.30

Table 4
Execution times (in minutes) of the tested methods (SA is Algorithm 1)

Group	Constrained case			Unconstrained case	
	SA	Algorithm of [3]	Algorithm of [2]	SA	Algorithm of [13]
G1	55.04	65.07	62.31	84.14	10.47
G2	170.38	208.56	192.23	292.12	21.58
G3	357.39	*	401.52	566.58	*

algorithm is not able to solve this type of instances (on our machine) because memory requirements are too large.

As one can see, Algorithm 2 is superior to the other methods giving, in average, a covering greater than 73% for both constrained and unconstrained cases and for all groups, while the surface usage of the other algorithms is less than 66% for the constrained case and less than 71.50% for the unconstrained one. Moreover, note that the method seems to be more efficient in unconstrained case than in the constrained one. This is due to the fact that since for the latter case the number of pieces available is greater than for the former one, more pieces can be placed in R, covering so a larger part of its surface.

We also note that, for all the tested algorithms, the surface covering increases with the size of instances. Concerning Algorithm 2, this phenomenon is due to the fact that the transformations (T1) to (T5) described in Section 2.4 make the introduction of small pieces around the big ones easier and this reduces the residual waste.

Table 4 shows the execution times (in minutes) of the different algorithms on the several types of instances. The average execution time for the Algorithm 1 is about 70 min for G1, about 3.85 h for G2, and about 7.7 h for G3. For the constrained case, Algorithm 1 is very efficient in time and faster than the algorithms of [3] and [2]. We think that an important factor explaining this superiority is the existence of the upper bounds b_i, $i = 1, \ldots, n$, that reduce duplications of the pieces. On the other hand, in the unconstrained case, Algorithm 1 is considerably time consuming, and in any case, much slower than the algorithm of [13]. This is due to the lack of bounds that allows consideration of a more important number of duplicate pieces. We also observe some superiority of the algorithm of [2] to the one of [3]. Very likely, this is due to the fact that the displacements and rotations performed produce in step 2 an initial feasible solution better than the solution produced by the latter. Achieving of such solution shortens the size of the tree constructed in step 3.

3.2. Second series of experiments

We now further confirm the efficiency of our method by testing it on instances for which optimal, or near-optimal solutions are known. We note here that *for all the experiments that will be described in the sequence, the method used is* Algorithm 1. More precisely, our SA-method has always computed feasible solutions in the interior of R; the cycles intersecting its edges have simply been removed.

3.2.1. The method of Stoyan–Yaskov

As we have already mentioned, [16] deals with the circular strip-packing problem. In this paper, a set of different circular pieces are considered, dimension H is fixed, authors try to minimize L.

For the tests of this section, the inputs of Algorithm 1 are the diameters of the pieces of [16] and the dimensions of the optimal plate: H (fixed in [16]) and L (the output of [16]). Our objective is to feasibly place cycles of [16] in such a way that the surface of the plate of dimensions (L, H) covered by them is as large as possible.

Professor Y.G. Stoyan has sent us two sets of tests. The first one includes two tests with the following characteristics:

1. in the first test, 100 circular pieces of diameters varying from 1.018 to 4.3694 and a dimension $H = 19$ are given; the least length L, such that all the pieces given are feasibly placed in a plate $R = L \times H$, is $L = 38.647179$; the usage of the plate in [16] is 82.24%; our inputs here are the diameters of the pieces and the dimensions $H = 19$ and $L = 38.647179$;
2. in the second test, 100 circular pieces of diameters varying from 1.066 to 4.3728 are to be placed to a plate whose height is fixed to $H = 15$; the least length L such that all the pieces given are feasibly placed in a plate $R = L \times H$ is $L = 38.0465$; the usage of the plate is 82.22%; as previously, our inputs are the diameters of the pieces and the dimensions of the final plate $H = 15$ and $L = 38.0465$.

A summary of the results obtained for the first set of tests is presented in Table 5. The approximation ratio of the third line is the ratio of the usage (%) of Algorithm 1 to the usage (%) of the algorithm of [16].

The second set of tests includes four tests with the following characteristics:

1. *inputs*: number of circular pieces: 20, $H = 8.5$ (the fixed dimension);
   outputs: $L = 14.895$, coverage: 81.597%;
2. *inputs*: number of circular pieces: 25, $H = 9$ (the fixed dimension);
   outputs: $L = 14.93$, coverage: 81.898%;
3. *inputs*: number of circular pieces: 30, $H = 9.5$ (the fixed dimension);
   outputs: $L = 17.491$, coverage: 83.144%;
4. *inputs*: number of circular pieces: 35, $H = 11$ (the fixed dimension);
   outputs: $L = 24.355$, coverage: 81.697%.

A summary of the results obtained by Algorithm 1 for the second set of tests is given in Table 6. Here also the approximation ratio of the third line is the ratio of the usage (%) of Algorithm 1 to the usage (%) of the algorithm of [16].

For more details about the experiments performed as, for example, the placement of the pieces in the solution computed by Algorithm 1, expressed by means of the coordinates (x_i, y_i) of their centers, or the

Table 5
The first set of tests with the method of [16]; by SA we denote Algorithm 1

	$(L,H) = (38.647179, 19)$		$(L,H) = (38.0465, 15)$	
	SA	Algorithm of [16]	SA	Algorithm of [16]
Number of pieces placed	91	100	92	100
Usage (%)	79.453	82.24	80.208	82.22
Approximation ratio	0.966	1	0.9755	1
CPU time (s)	174		159	

Table 6
The second set of tests with the method of [16]; as in Table 5, by SA we denote Algorithm 1

(L,H)	(14.895,8.5)		(14.93,9)		(17.491,9.5)		(24.355,11)	
	SA	[16]	SA	[16]	SA	[16]	SA	[16]
Pieces placed	19	20	23	25	28	30	33	35
Usage (%)	69.908	81.597	65.385	81.898	74.357	83.144	71.796	81.697
Ratio	0.857	1	0.798	1	0.894	1	0.879	1
CPU time (s)	36		49		42		121	

pieces that have not been placed (with respect to the corresponding solution of [16]), the interested reader can contact the authors of this paper.

3.2.2. SA and the strip-packings of Graham–Lubachevsky

In [12], authors consider disks of the same diameter. They fix a number of cycles and a diameter, and search for the *side length of the smallest square that contains the disk centers*. Here, inputs are the diameter of the cycles and their number; the output of each packing is the parameter m, i.e., *the ratio of the disk diameter to the side length of the smallest square (called Graham–Lubachevsky square in what follows) that contains the disk centers*.

In order to follow a protocol similar to the one of section, we had to determine the dimension L of the Graham–Lubachevsky square R containing these *disks*. Next, we used L and the diameters of the cycles as inputs, and we computed the number of cycles feasibly packed in R. All cycles having the same radius, there exists a direct relationship between their number and the surface covered.

Since we have no numerical data for the results of [12], we have created them based upon some of the Figures 2.1–2.9 (pp. 5–13) of the paper. This was somewhat awkward, and for reasons of accuracy of our input-data, we have chosen some figures where the packing of the cycles was "convenient". Any of the packings given in [12] can be seen as a matrix whose inputs are the cycles packed. We have chosen packings where the centers of the cycles of the first and last column and row of this matrix lied on the four edges of the Graham–Lubachevsky square. For any of these packings, we have proceeded as follows:

- we have fixed the radius r of cycles to be equal to 1;
- we have determined the Graham–Lubachevsky square of side length $2r/m$ and have set $L = H = (2r/m) + 2 = (2/m) + 2$ (L and H are the dimensions of the plate considered);
- we have run Algorithm 1 with inputs identical circular pieces with $r = 1$ and with L as determined in the item just above.

We have performed five tests refereed, in what follows, by the number of cycles packed, the value of m, the figure number and the page number in [16]:

Test 1: 25 disks, $m = 0.25$ (first sub-figure of Figure 2.2, p. 6);
Test 2: 51 disks, $m = 0.16561837431260$ (second sub-figure of Figure 2.7, p. 11);
Test 3: 52 disks, $m = 0.16538623796964$ (third sub-figure of Figure 2.7, p. 11);
Test 4: 56 disks, $m = 0.15615650046215$ (sixth sub-figure of Figure 2.7, p. 11);
Test 5: 72 disks, $m = 0.13541666666667$ (fifth sub-figure of Figure 2.8, p. 12).

The results for the five tests just described, as well as the sides of the squares that we have computed, are shown in Table 7. More details about the placement of the pieces in the solution computed by Algorithm 1 (coordinates (x_i, y_i) of their centers in the plate) are available; the interested reader can contact the authors of this paper.

Table 7
Number of disks placed by Algorithm 1 and the corresponding patterns of [12]; the disks are supposed to be of radius 1

	L	Algorithm 1	Packing of [12]
Test 1	10	25	25
Test 2	14.076	52	51
Test 3	14.093	52	52
Test 4	14.807	52	56
Test 5	16.769	75	72

As one can see, the results of Algorithm 1 coincide with the results of [12] for tests 1 (25 disks) and 3 (52 disks). For test 4 (56 disks in [12]), the result of Algorithm 1 is inferior to the corresponding result of [12], while for tests 2 and 5 (51 and72 disks in [12], respectively) the inverse situation is produced. Dealing with test 5, it is mentioned in [12] that their pattern does not represent the best possible packing; hence, our method brings to the fore a better one. From the coordinates of the centers of the disks in the square, we have a strong intuition that the packing computed is (if not optimal) very near to the optimal one. Test 2 is even more interesting and intriguing, since the corresponding pattern is conjectured to be optimal in [12]. We presently have no sufficient elements to privilege the one or the other result. It is likely that the decoding of the corresponding figure of [12] (second sub-figure of Figure 2.7, p. 11) has introduced some errors to the computation of L. We have considered that the centers of the cycles of the two extremal cycle-columns lie on the two horizontal edges of the Graham–Lubachevsky square. But it is impossible to visually detect from the respective figure whether disk centers lie exactly on the edges of the Graham–Lubachevsky square or whether they were slightly moved to the right or to the left. In such case, this displacement together with the rounding of the quantity $2/m$ could introduce some error in the computation of the dimensions of the square, and therefore, an error on the computation on the number of disks feasibly placed in the real Graham–Lubachevsky square. So, further comparisons of our method with the results of [12] have to be done. They are actually in progress.

4. Conclusion

We have presented a SA method for solving both unconstrained and constrained circular cutting problems. The approach used is principally based upon an energy function that, when it becomes minimum, provides solutions composed by a set of pieces concentrated at the bottom-left corner of the initial rectangle. Computational results show that the approach presented is able to produce good solutions when compared with different approaches of the literature. Experiences with examples taken from [16,12] further confirm its efficiency. Its behavior with respect to the patterns presented in [12] merits further depthening and such a work is actually in progress. Finally note that no post-processing has been applied in the results of the second series of tests (taken form [16,12]), except of removal of cycles intersecting the right edge of the plate.

Acknowledgements

The pertinent suggestions and useful remarks of three anonymous referees largely contributing in the improvement of the impact, the scientific quality, the legibility and the presentation of this paper are gratefully acknowledged. Many thanks to Professor Y.G. Stoyan for having provided us with numerical data and results from his nice work [16] on strip-packing.

References

[1] E. Aarts, J. Korst, Simulated Annealing and Boltzmann Machines, a Stochastic Approach to Combinational Optimization and Neural Computing, John Wiley & Sons, Chichester, UK, 1989.
[2] A. Albano, G. Sapuppo, Optimal allocation of two-dimensional irregular shapes using heuristic search methods, IEEE Transactions on Systems, Man, and Cybernetics 10 (5) (1980) 242–248.
[3] J.E. Beasley, An exact two-dimensional non-guillotine cutting tree search procedure, Operations Research 33 (1) (1985) 49–64.
[4] N.E. Collins, R.W. Eglese, B.L. Golden, Simulated annealing, an annotated bibliography, American Journal of Mathematical and Management Sciences 8 (1988) 209–307.

[5] M.H. Correia, J.F. Oliveira, J.S. Ferreira, Cylinder packing by simulated annealing, Pesquisa Operacional 20 (2) (2000) 269–286.
[6] K.A. Dowsland, Palletisation of cylinders in cases, OR Spektrum 13 (1991) 171–172.
[7] K.A. Dowsland, W.B. Dowsland, Solution approaches to irregular nesting problems, European Journal of Operational Research 84 (1995) 506–521.
[8] H. Dyckhoff, A typology of cutting and packing problems, European Journal of Operational Research 44 (1990) 145–159.
[9] H.J. Fraser, J.A. George, Integrated container loading software for pulp and paper industry, European Journal of Operational Research 77 (1994) 466–474.
[10] J.A. George, J.M. George, B.W. Lamar, Packing different-sized circles into a rectangular container, European Journal of Operational Research 84 (1995) 693–712.
[11] P. Gilmore, R. Gomory, The theory and computation of knapsack functions, Operations Research 14 (1966) 1045–1074.
[12] R.L. Graham, B.D. Lubachevsky, Repeated patterns of dense packings of equal disks in a square, The Electronic Journal of Combinatorics 3, 1996 Report #16.
[13] M.J. Haims, H. Freeman, A multistage solution of the template-layout problem, IEEE Transactions on Systems, Man, and Cybernetics 6 (1970) 145–151.
[14] M. Hifi, V.T. Paschos, V. Zissimopoulos, Circular cutting problem: A simulated annealing approach, Technical Report 94.15, CERMSEM, Université Paris I, 1994.
[15] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, Journal of Chemical Physics 21 (6) (1953) 1087–1092.
[16] Y.G. Stoyan, G.N. Yaskov, Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints, International Transactions of Operational Research 5 (1) (1998) 45–57.
[17] P.E. Sweeney, E.R. Paternoster, Cutting and packing problems: A categorized, application-oriented research bibliography, Journal of the Operational Research Society 43 (7) (1992) 691–706.