Applying the Simulated Annealing to the Problem of Positioning Rotational Non Convex Polygons

Martins, T. C
Department of Mechatronics and Mechanical Systems Engineering, EPUSP
thiago.martins@poli.usp.br

Tsuzuki, M. S. G.
Department of Mechatronics and Mechanical Systems Engineering, EPUSP
mtsuzuki@usp.br

This work deals with the problem of minimize the waste of space that occurs on a placement of a set of bi-dimensional polygons inside a bi-dimensional container. This problem is approached with an heuristic based on Simulated Annealing, which is inspired on the physico-chemical process that take place during the recrystalisation of a metal. Traditional "external penalisation" techniques are avoided through the application of the Minkowski sum algorithm, that determinates collision-free areas for the set of polygons. That gives to the proposed process a more universal character, as external penalisations are based on empiric parameters of great influence on the optimisation performance. The proposed process is suited for non-convex polygons and containers, and can be easily adapted for related problems, such as container size minimisation.

1. Introduction to placement problems

The polygonal placement problems arises in the industry whenever one must place multiple objects inside a container so that there is no collision between the objects, while either minimizing the size of the container either maximizing the volume occupied by the objects. For instance, on the shoe industry, a maximum number of shoe parts must be placed over a leather piece. Another instance of the problem occurs in the textile industry, where cloth parts must be placed over the smallest possible sheet of tissue. Those problems are also closely related to robot motion planning, where a trajectory for an object (the robot) that leads it from one point to another must be found while avoiding collisions with pre-placed obstacles. Those placement problems are also known by the names of nesting, containment, layout, packing and cutting stock (Downsland et al., 1995). Surveys of placement problems can be found at (Lodi et al., 2002; Downsland et al., 1992; Downsland et al., 1995). Computational approaches often focus on purely translational versions of the problem, and the major part of them approach problems whose polygons are constrained to rectangular shapes.

2. This work

As seen on the section 1, due to their complexity, irregular placement problems are the least computationally approached, despite their great relevancy for the industry. This is the motivation for this work, which deals with the bidimentional placement on its most unconstrained form, the translational and rotational placement of heterogeneous irregular forms (both nonvex and non convex) on irregular containers (also both convex and non convex).

2.1 Problem statement

The problem can be defined as the problem of, given a container (a polygon, convex or non-convex) and a polygon set, to determinate a subset of polygons and the transformations (translations and rotations) that, when applied to their respective polygons, place them without collisions inside the container while minimising the wasted space (see Figure 1).

Definition 1 *There is a collision between two polygons when their intersection is a non-empty set.*

As mentioned, it can be show that even restricted versions of this problem (for instance, limiting the polygon shape to rectangles only) are NP-Complete, which means it is believed they cannot be algorithmically solved for practical instances (Fowler et al. 1981). This motivates an heuristic approach, that while is not guaranteed to find the optimal solution, can, in a reasonable amount of time, find a good solution. Probabilistic optimisation heuristics follow this pattern: while a stipulated stop criteria is not satisfied, at each step the function to be optimised is evaluated at a set of points and a set of rules is applied to determinate the set of points to be evaluated at the next step. The process converges to a solution of the problem.

For our problem, the space of valid solutions is the set of transformations $(\Delta x, \Delta y, \Delta\theta)$ to be applied to the polygons restricted to the condition of no-collisions between them. The space delimited by those restrictions is very complex (and mutable). Usually, when confronted to such complex spaces, probabilistic heuristics "relax" the original constrains of

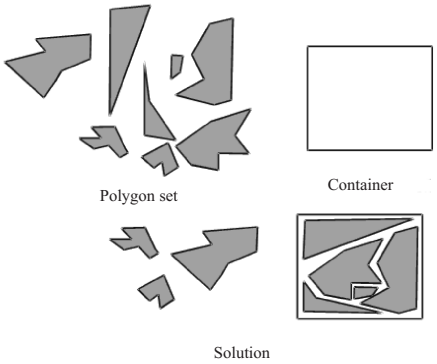


Figure 1. A placement problem and its optimal solution.

the problem, allowing the search to go through points outside the space of valid solutions and applying penalisations to their cost. This technique is known as external penalisation. While at first this technique greatly simplifies the problem, it also introduces and additional problem: How to determinate the exact amount of penalisation to be applied to external points? Another drawback of external penalisation is that it can lead the optimisation process to non-valid solutions. This can be seen on the work of Heckman (1995), where problems very similar to those studied here are approached with simulated annealing. On his work. Heckman points that his optimisation process can produce invalid solutions (solutions with collisions between polygons), requiring a post-processing step of the obtained data.

The approach adopted here avoids the pitfalls of external penalisation by the continuous mapping (meaning it is updated at each step of the process) of the complex space of valid solution into a simplified space. Although this additional mapping step increases the complexity of the process, it confers to the process a more universal character, as there is one less empiric parameter to be defined. Actually, the proposed process does not explores the whole space of possible solutions, focusing instead on a reduced space, that contains at least one optimal solution. This reduces the search through irrelevant points and enhances the performance of the process.

2.2 Simulated annealing

Simulated Annealing (Kirkpatrick et al., 1983) is the probabilistic meta-heuristic adopted on this work. It was chose due to its capacity of "escape" from local minima (which are very frequent on this problem). It is also worth of mention that the process of recrystalisation, the inspiration for simulated annealing, is a natural instance of a placement problem.

2.2.1 Description

Simulated annealing comes from the Metropolis algorithm, a simulation of the recrystalisation of atoms on a metal during its annealing (gradual and controlled cooling). During annealing, atoms migrate naturally to configurations that minimize the system total energy, even if during this migration the system must pass through high-energy configurations. The observation of this behavior suggests the application of the simulation of such process to combinatorial optimisation problems.

Simulated annealing is a hill-climbing local exploration[1] optimisation heuristic, which means it can skip local minima by allowing the exploration of the space in directions that lead to an increase on the cost function. It sequentially applies random modifications on the evaluation point of the cost function. If a modification yields a point of smaller cost, it is automatically kept. Otherwise, the modification also can be kept with a probability obtained form the Boltzman distribution (1).

$$P(\Delta E) = e^{-\frac{\Delta E}{kt}} \tag{1}$$

where $P(\Delta E)$ is the probability of the optimisation process to keep a modification that incurred on an increase ΔE of the cost function, k is a parameter of the process (analogous to the Stefan-Boltzman constant) and t is the instantaneous "temperature" of the process. This temperature is defined by a cooling schedule, and it is the main control parameter of the process. Several cooling schedules were evaluated for our problem. The figure 2 shows the shape of the Boltzman distribution. It can be seen that the probability of a given state decreases with its energy, but as the temperature rises, this decrease (the slope of the curve $P(\Delta E)$) diminishes.

[1]Local exploration heuristics are heuristics that search the space of solutions sequentially, jumping from one solution to a neighbor, possibly more interesting.

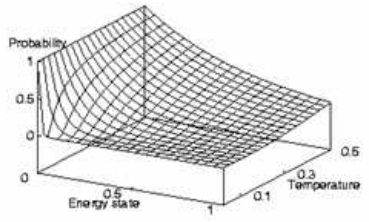


Figure 2. The Boltzman distribution.

2.2.2 Algorithm

Be the problem of minimise the function $F(x)$, where x is a vector. The algorithm starts with a feasible random solution x_0. Next, at each iteration, it applies a transformation to the solution producing a new feasible solution x^* on the neighborhood of x. The cost increase $\Delta E = F(x^*) - F(x)$ is evaluated. If this increase is negative (meaning the new solution has smaller cost), the new solution is automatically kept. If this increase is positive (meaning the new solution has greater cost), a random number r is uniformly generated between 0 e 1. If r is lesser than the probability calculated by (1), x^* replaces x as the new current solution. Otherwise, x^* is discarded.

```
    Optimisation by simulated annealing
1.  x ←<Initial random solution>
2.  i ← 0
3.  WHILE <Global stop condition not satisfied> DO
4.    t ← CoolingSchedule(i)
5.    i ← i + 1
6.    WHILE <Local stop condition not satisfied> DO
7.      x* ←RandomModification(x, t)
8.      ΔE = F(x*) − F(x)
9.      IF ΔE < 0 THEN
10.       x ← x*
11.     ELSE
10.       r ←<random uniform number between 0 and 1>
12.       IF r < e^(−ΔE/t) THEN
13.         x ← x*
14.       END IF
15.     END IF
16.   END WHILE
17. END WHILE
18. <Display solution x>
```

Figure 3. The optimisation by simulated annealing algorithm.

As can be seen on the figure 3, the algorithm executes iterations at a fixed temperature until a specific stop condition is not met. This condition determinates whether the system has attained "thermal equilibrium" at a determinated temperature. Usually, it is defined as a maximum number of accepted modifications and a maximum number of iterations at a temperature. When any of the condition is met, the algorithm proceeds to the next temperature on the cooling schedule. The global stopping condition usually is defined by the cooling schedule itself. When the cooling schedules reaches its end, the algorithm stops. There are, though, condition to stop the algorithm before the cooling schedule ends. A common condition is a maximum number of iterations of the algorithm without significative progress (a situation known as "frozen state").

2.2.3 Cooling schedules

The following cooling schedules (Steinhöfel et al., 1998) were evaluated:

Logarithmic cooling This schedule has proved convergence for the global miminum with a given choice of parameters

i_0 and c. But it is usually too slow for our problem.

$$t_i = \frac{c}{\log(i + i_0)} \tag{2}$$

Linear cooling This schedule has a tendency of being too fast on the final moments of the algorithm, leading it to premature non-optimal frozen states.

$$t_i = t_{i-1} - \Delta t \tag{3}$$

Geometric cooling This schedule usually leads to good results, allowing the system to settle at an optimal configuration as the temperature falls. The parameter α is usually chosen around 0.95, while the determination of an appropriate value remains a difficult problem.

$$t_i = \alpha \cdot t_{i-1} \tag{4}$$

Adaptative cooling This schedule is a variation of the geometric cooling. It tries to solve the problem of determination of α by the evaluation of the "specific heat" of the system using the standard deviation of the cost function σ_i during the iterations at a fixed temperature. This schedule produces the best results.

$$t_i = \alpha_i \cdot t_{i-1} \tag{5}$$

$$\alpha_i = e^{-\frac{\lambda \cdot t_{i-1}}{\sigma_{i-1}}} \tag{6}$$

2.2.4 Initial temperature

The initial temperature must be chosen such that the system does not get stuck on a sub-set of solutions at the beginning of the process. So, it must be selected in order to allow almost any modification of the solution. On the other hand, a too much elevated initial temperature leads to redundant iterations of the process. A proposed (Heckmann et al., 1995) heuristic of initial temperature determination is:

$$T_{initial} = \frac{-3\sigma_E}{\ln(P)} \tag{7}$$

Where σ_E is the standard deviation of the cost function obtained through some iterations of the algorithm, and P is an arbitrary parameter (related to the probability of acceptance of initial solutions) taken between 0.85 and 0.5.

2.3 Application to the placement problem

Hertz and Widmer (2003) proposed general rules for applications of local-search meta-heuristics to combinatorial optimisation problems. The proposed rules are:

1 *It must be easy to produce feasible solutions*

2 *For each feasible solution, there must be a path linking it to an optimal solution.*

3 *The solution explored at an iteration must be at the neighborhood of the solution explored at the preceding iteration.*

4 *The topology of the space induced by the cost function must not be too flat.*

As exposed previously, simulated annealing approaches to placement problems follow the rule 1 by use of external penalisation, a technique that has its shortcomings. This work proposes an alternative to external penalisation while respecting rule 1. Its proposal is to continuously map a subset of the whole space of feasible solutions into a simplified space (a space constrained by very simple rules, so it is trivial to generate feasible solutions inside this space). The subset of feasible solutions considered by the process is the subset of *connected solutions*. In order to define Para *connected solutions* let's define the *connectivity graph* of a solution.

Definition 2 *Two distinct polygons A e B are said to be connected if there are at least one point on the perimeter of A for which there is not an neighborhood that does not contains at least one point on the perimeter of B.*

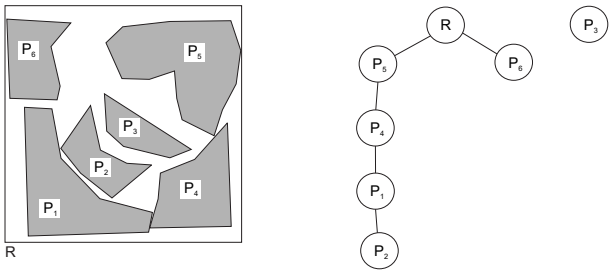
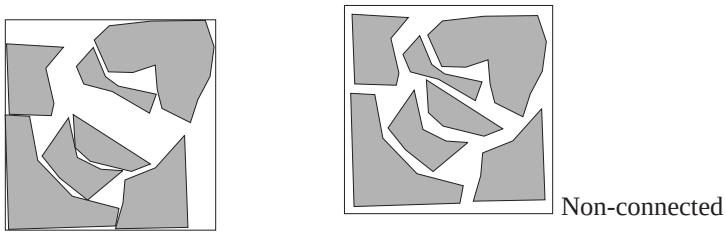


Figure 4. A feasible solution and its connectivity graph.



connected Solution

Non-connected Solution

Figure 5. Example of connected and non-connected solutions.

Definition 3 *The* connectivity graph *related to a solution s of a placement problem is the graph whose vertexes are composed by the container P_0 and by the polygons P_i placed inside it. The edge $E_{i,j}$ exists only if the polygon i is connected to the polygon j.*

Definition 4 *A solution s of a placement problem is said to be* connected *only and if only its connectivity graph is connected.*

In short, a connected solution is a solution without "loose" polygons on the container.

Theorem 1 *It is possible to construct a connected solution for any from any non-connected solution of a placement problem that allows free translation without increase in cost.*

Proof: For each sub–graph of the connectivity graph related to the given solution, it is possible to translate the polygons related to its vertexes until at least one of them become connected to a polygon belonging to another connected sub-graph, generating another solution of same cost where two connected sub–graphs are merged. So, it is always possible to construct from any solution which connectivity graph has $n > 1$ connected sub–graphs a solution which connectivity graph has $n - 1$ connected sub–graphs. By induction, it is possible to build a connected solution.

Corollary 2 *Every placement problem that allows free translations has at least one optimal connected solution.*

From the corollary 2 we see that restraining the exploration to the space of connected solutions does not violate rule 2. This restrained exploration is made by the sequential placement of the polygons inside the container. For each polygon to be placed, its collision-free area is calculated (the area where it can be placed without collisions with already placed polygons). The polygon is then placed on the perimeter of this area, ensuring thus that it will always be connected to at least one polygon. This placement is controlled by parameters produced by the Simulated Annealing. Those parameters are the order in which the polygons are placed, the rotation applied to each polygon and the relative position of each polygon inside its collision–free area. As the shape and number of collision-free areas changes with the position of the polygons already placed inside the container, the points on their perimeter (where the new polygon will be placed) must be mapped on normalized variables. The points inside each collision-free area are mapeds into an uniform variable $t \in [0, 1[$. This is done by picking a reference point on the perimeter (on this work this point is the leftmost point on the area). This reference point is the equivalent of the placement point for $t = 0$. From this point, the perimeter followed counterclockwise and its points are mapped uniformly on the interval $]0, 1[$. There remains the matter of deciding on which of the many possible free areas to place the polygon. For that, the free areas are sorted after some criteria (on this work they are sorted after the x coordinate of their leftmost vertex). Based on that, at each area is assigned one uniform division of the interval $[0, 1[$. The parameter f of a polygon dictates on which area the polygon shall be placed.

For instance, let´s suppose that for a particular polygon at a given moment of the process there are four free areas, A_1, A_2, A_3 and A_4. Let´s suppose their sorting produced the sequence (A_2, A_1, A_3, A_4). Then, the polygon shall be placed in the area:

A_2 if $f \in [0, 1/4[$,
A_1 if $f \in [1/4, 1/2[$,
A_3 if $f \in [1/2, 3/4[$,
A_4 if $f \in [3/4, 1[$

So, to each polygon are assigned parameters (θ, t, f), produced by the simulated annealing algorithm (as the placement order). The placement process of a single polygon occurs in the following sequence (see figure 6):

- The rotation θ is applied to the polygon.
- The collision–free areas inside the container are calculated.
- The parameter f is used to determinate on which free area the polygon shall be placed.
- Using the parameter t, the polygon is placed on the perimeter of the chose area.

When there are no more free areas for a polygon, it is not placed on the container[2].

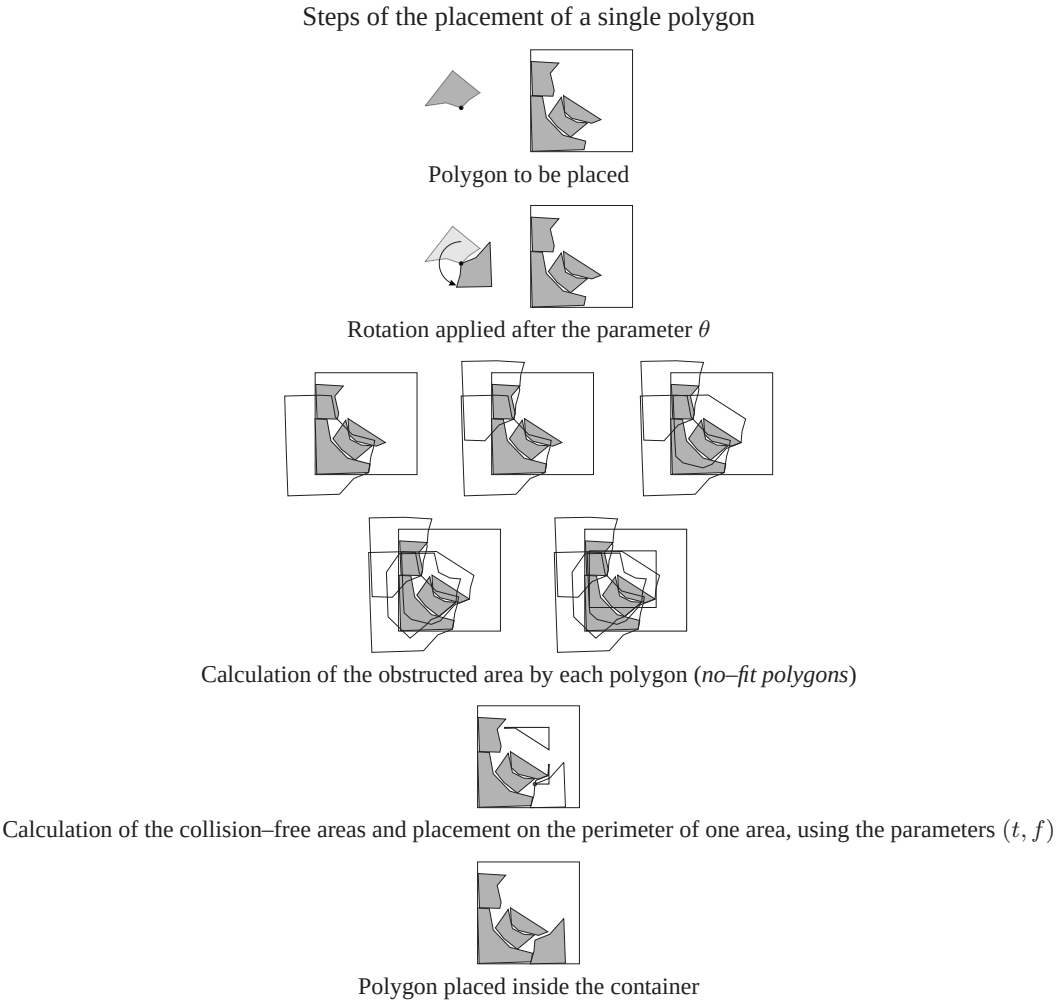


Figure 6. Example of placement of a single polygon during the optimisation process.

2.4 Calculating collision–free areas through Minkowski sum

The collision–free areas for a given polygon are obtained by the boolean subtraction from the container area the *no–fit polygons* induced by all polygons already placed.

[2] But see the section 2.5.1

Definition 5 *The non–fit polygon induced by a polygon B to a polygon A noted $B \ominus A$ as the set of translation vectors applied to A that leads it to a* collision [3] *collision with B, thus $B \ominus A = \{O + \vec{v} | A + \vec{v} \in B\}$.*

The non–fit polygons can be obtained by the *Minkowski sum* algorithm.

Definition 6 *The Minkowski sum of two polygons A and B, noted $A \oplus B$, is defined as the set of points $\{O + \vec{v_a} + \vec{v_b} | O + \vec{v_a} \in A, O + \vec{v_b} \in B\}$.*

Definition 7 *The polygon opposed to a given polygon A noted $-A$ is defined as the set of points $-A = \{O - \vec{v_a} | O + \vec{v_a} \in A\}$.*

The opposed polygon is obtained inverting the signal of all coordinates of the original polygon. From the above definitions, one can see that $B \ominus A = B \oplus (-A)$, meaning that the no–fit polygon is produced by the Minkowsky sum of the obstacle with the polygon opposed to the one to be placed. Minkowski sums can be calculated very efficiently for convex polygons. The result of a Minkowski sum of two convex polygons is a convex polygon build from the edges from the original polygons taken in counterclowsise order. Non–convex polygons can be decomposed on a pre–processing step, as the transformations applied (rotations and translations) do not affect such decomposition.

2.5 Evaluation of the cost function from the placement parameters

As seen, the polygon placement is conducted by the following parameters:

- Placement order for the polygons
- For each polygon, parameters (θ, f, t)

The cost function to be minimised is the wasted space on the container. Its evaluation takes place as explained of figure 7.

Algorithm for cost evaluation of the placement parameters

```
1.  FOR <each polygon to be placed> DO
2.    <Apply the rotation θ to the polygon>
3.    <Calculate the no–fit polygons>
4.    <Calculate the collision-free areas>
5.    IF <there are free areas> THEN
6.      <sort the areas>
7.      <With the parameter f of the polygon, select a free area>
8.      <With the parameter t, select a point on the area perimeter>
9.      <Add the polygon to the container>
10.   FIM SE
11.  FIM PARA
12.  <Calculate the wasted space>
```

Figure 7. Evaluation of the cost function for the placement.

2.5.1 Handling solutions with identical costs

As mentioned on rule 4, the existence of contiguous solutions with identical costs is harmful to the algorithm performance. To sort out those cost collisions, the cost of a given solution can be modified in order to estimate how close this solution is to have a non-placed polygon fitted on the container. So, for each non–placed polygon, a limited-deph binary search can be performed to find a scale factor (between 0 and 1) that, when applied to the polygon, would allow it to be fitted on the container.

2.6 Generation of placement solutions

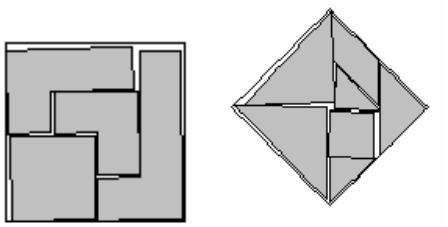
The initial solution is generated at random. At each step, the preceeding solution is modified in order to produce a new exploration point. As mentioned at rule 3, it is intersting that this new solution is taken at the neighborhood of the preceeding solution. Therefore, a single polygon is randomly selected and only one of its parameters (θ, t, f) (also

[3] see definition 1.

chosen at random) is randomly modified. As all the parameters (θ, t, f) are defined in $[0, 1[$, the modification consists in adding a random number (not necessarily uniform — see below) Δ generated in -1/2 e 1/2 and take the result modulus 1. Rejected solutions do not contributed to the progress of the optimisation process. So, the distribution of Δ are adapted in order to increase the number of accepted solutions. When at a given iteration the modification applied to a parameter leads to a rejected solution, the Δ number distribution for that specific parameter is modified in order to have its standard deviation reduced (resulting on a lesser modification amplitude). When the modification leads to an accepted solution, the distribution of Δ for that parameter is modified so its standard deviation grows (resulting in a bigger modification amplitude).

3. Results

The optimisation method was implemented using a modified version of the *PolyBoolean* library (Leonov, 1998). All problem instances studied here have a solution where all polygons can be fitted on the container. That allowed the adoption of artifical stop conditions to simplify the study. On all problems, the container aera is 10% larger than the total polygons area. Four non-convex non-congruent polygons (Figure 8.(a)). Seven convex non–congruent polygons (Figure 8.(b)).



(a) (b)

Figure 8. (a) Placement of four non–convex non–congruent polygons. (b) Placement of seven convex non–congruent polygons.

4. Conclusions

The proposed process has the merit, when compared to traditional approaches, of not using external penalisation, with ehances its performance and makes it more independend of the nature of the problem.

5. Acknowledgements

This research was partially supported by CNPq.

6. References

Downsland, K. A. and Downsland, W. B.; 1992. “Packing Problems”, *European Journal of Operational Research*, 56:2-14.

Downsland, K. A. and Downsland, W. B.; 1995. “Solution approaches to irregular nesting problems”, *European Journal of Operational Research*, 84:506-521.

Fowler, R. J.; Paterson, M.; Tanimoto, S. L.; 1981. “Optimal Packing and Covering in the Plane are NP-Complete.”, *Inf. Process. Lett.*, 12(3):133-137.

Heckmann, R. and Lengauer, T.; 1995. “A simulated annealing approach to the nesting problem in the textile manufacturing industry”, *Annals of Operations Research*, 57:103-133.

Herz, A. and Widmer, M.; 2003. “Guidelines for use of meta-heuristics in combinatorial optimization”, *European Journal of Operational Research*, 151:247-252.

Kirkpatrick, S.; Gellat, C. D.; Vecchi, M. P.; 1983. “Optimization by Simulated Annelaing”, *Science*, 220:671-680.

Leonov, M. V. Leonov; 1998. *Implementation of Boolean operations on sets of polygons in the plane*, Novosibirsk State University, BS Thesis.

Lodi, A.; Martello, S.; Monaci, M.; 2002. “Two-dimensional packing problems: A survey”, *European Journal of Operational Research*, 141:241-525.

Steinhöfel, K.; Albrecht, A. A.; Wong, C. K.; 1998. “On Various Cooling Schedules for Simulated Annealing Applied to the Job Shop Problem”, *RANDOM*, 260-279.