Rotational Placement of Irregular Polygons over Containers with Fixed Dimensions using Simulated Annealing and No-Fit Polygons

Article *in* Journal of the Brazilian Society of Mechanical Sciences and Engineering · July 2008

DOI: 10.1590/S1678-58782008000300005

CITATIONS
8

READS
225

3 authors, including:

Thiago C Martins
University of São Paulo
55 PUBLICATIONS **274** CITATIONS

SEE PROFILE

Marcos Tsuzuki
University of São Paulo
142 PUBLICATIONS **576** CITATIONS

SEE PROFILE

Thiago de C. Martins
thiago@usp.br

Marcos S. G. Tsuzuki
Senior Member, ABCM
mtsuzuki@usp.br
University of Sao Paulo - USP
Escola Politécnica
Dept. of Mechatronics and Mech. Syst. Eng
05508-900 São Paulo, SP, Brazil

Rotational Placement of Irregular Polygons over Containers with Fixed Dimensions using Simulated Annealing and No-Fit Polygons

This work deals with the problem of minimizing the waste of space that occurs on a rotational placement of a set of irregular bi-dimensional small items inside a bi-dimensional large object. This problem is approached with an heuristic based on simulated annealing. Traditional "external penalization" techniques are avoided through the application of the no-fit polygon, that determinates the collision-free region for each small item before its placement. The simulated annealing controls: the rotation applied and the placement of the small item. For each non-placed small item, a limited depth binary search is performed to find a scale factor that when applied to the small item, would allow it to be fitted in the large object. Three possibilities to define the sequence on which the small items are placed are studied: larger-first, random permutation and weight sorted. The proposed algorithm is suited for non-convex small items and large objects.
***Keywords**: knapsack problem, cutting and packing, optimization*

Introduction

The knapsack problem arises in the industry whenever one must place multiple small items inside a large object such that there is no collision between the small items, while either minimizing the size of the large object or maximizing the volume occupied by the small items. High material utilization is of particular interest to mass production industries since small improvements of the layout can result in large savings of material and considerably reduce production cost.

The knapsack problem belongs to the more general class of combinatorial problems known as cutting and packing problems. According to Dyckhoff (1990), the cutting and packing problems are mainly characterized by the number of relevant dimensions, the regularity and irregularity of the shapes of the small items and large objects and the problem assignment. Considering assignment, it is possible to identify two situations: output maximization and input minimization. In the input minimization, the set of large objects is sufficient to accommodate all small items, and there is no selection regarding the small items. In the case of output maximization, the set of large objects is not sufficient to accommodate all the small items and a set of small items has to be assigned to a given set of large objects. The survey made by Wascher et al. (2005) improved the typology of cutting and packing problems proposed by Dyckhoff (1990). According to Wascher et al. (2005), the problem studied in this work can be defined as: "... the knapsack problem represents a problem category which is characterized by a strongly heterogeneous assortment of small items which have to be allocated to a given set of large objects. Again, the availability of the large objects is limited such that not all small items can be accommodated. The value of the accommodated small items is to be maximized". In this survey, Wascher et al. (2005) identified 294 papers containing material relevant to cutting and packing. Only 5 papers thereof were classified as dealing with two dimensional irregular single knapsack problems. This fact shows that the literature related to this kind of problem is scarce.

It can be shown that even restricted versions of this problem (for instance, limiting the small item shape to rectangles only) are NP-Complete, which means that all algorithms currently known for finding optimal solutions require a number of computational steps that grows exponentially with the problem size rather than according to a polynomial function (Fowler et al., 1981). It is not worthwhile to search for an exact (optimal) algorithm, since it does not appear that any efficient optimal solution is possible. Alternative approaches that are not guaranteed to find an optimal solution are considered instead. Thus, by giving up solution quality, computational efficiency can be gained. Probabilistic optimization heuristics follow this pattern: while a stipulated stop criteria is not satisfied, at each step the function to be optimized 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 algorithm stops when a satisfactory solution of the problem is reached.

Jakobs (1996) studied orthogonal packing where the small items and large objects are rectangular. He used a bottom-left strategy to position the small items; the placement sequence is determined using a genetic algorithm. He extended the algorithm to process irregular small items. The main idea is the determination of embedding rectangles with minimum area for all small items. The rotation of the small items is determined in this local search. The large object is big enough for all small items to be easily placed. The algorithm considers all small items as rectangles and ensures that no overlap exists among them. This bounding rectangle algorithm is not a good approach as the bounding rectangle usually contains wasted material.

Hifi and Hallah (2003) proposed a hybrid algorithm to solve the irregular problem. The hybrid algorithm searches for an optimal ordering of the small items using a genetic algorithm and identifies the best packing using a constructive approach, which consists of sequentially positioning a set of ordered small items. Each small item is tested for a set of potential positions defined with respect to already positioned small items. They studied the exclusively translational problem where the large object is big enough such that all small items can be easily placed. The algorithm considers all small items as rectangles for positioning, then a translation is applied to pack the configuration and, simultaneously, ensuring that no overlap exists among them.

Recently, researchers used the no-fit polygon concept to ensure feasible layouts; i.e. layouts where the small items do not overlap and fit inside the large object. This concept was first introduced by Art (1966). Given two small items, *A* and *B*, the no-fit polygon can be found by tracing one shape around the boundary of the other. One of the small items remains fixed in space and the other slides in contact with the fixed small item's boundary whilst ensuring that the small items always touch but never intersect.

Paper accepted May, 2008. Technical Editor: Glauco A. de P. Caurin.

Dowsland et al. (2002) used the no-fit polygon and a bottom-left strategy as a deterministic heuristic. A small item is placed adjacent to the boundary of the no-fit polygon according to the bottom-left strategy. The positioning sequence is algorithmically defined by a sorting criteria: decreasing area, decreasing length, decreasing width and others. They studied an exclusively translational problem and the large object is big enough such that all small items can be easily placed without overlap.

Gomes and Oliveira (2006) used the no-fit polygon concept to eliminate the overlap in the definition of the initial solution. Simulated annealing is used to define which small items exchange position and with which orientation. After exchanging position of two small items in the layout, overlap usually occurs, which is removed by applying a separation model, i.e. a set of translations is applied to the overlapping small items. Afterwards, the layout is compacted by a set of translations applied to the placed small items, achieving layouts that are local minima. However, it is possible that the separation model fails in achieving a feasible layout. In this case, the proposed algorithm ignores the failed swap operation and attempts exchanging two different small items.

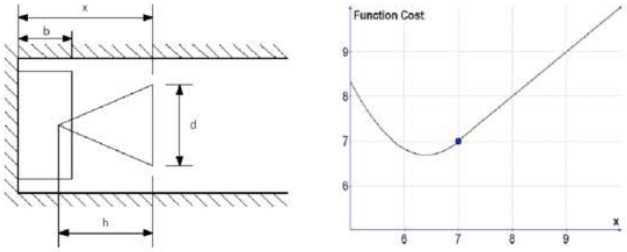
Probabilistic heuristics explore the domain space in a effective way, however the space delimited by the non-overlap restriction is very complex. Usually, when confronted with such complex spaces, probabilistic heuristics "relax" the original constraints of the problem, allowing the search to go through points outside the space of valid solutions and applying penalization to their cost. This technique is known as external penalization. Several authors suggest that such an approach which allows but penalizes configurations with overlapping small items in the solution space is more efficient (Heckmann and Lengauer, 1995; Bennel and Dowsland, 2001). However, depending on the severity of the penalty this relaxation results in a tendency to converge toward infeasible solutions. A further problem is that feasible solutions may be forced to be computed at the expense of overall quality.

This work shows the application of a probabilistic heuristic, the so-called simulated annealing, to define the placement of small items without the use of external penalization. The knapsack problem has an additional difficulty: the large object is bounded. In this kind of problem, usually the cost function is the value of the unfilled area. A limitation of this approach is its inability to differentiate between two different packing arrangements of the same set of small items. In this paper, this difficulty is solved by scaling unplaced small items and based thereon defining a measure of compactness of the remaining free space in a packing arrangement.

External Penalization

The usual approach to simulated annealing applied to the kind of complex spaces discussed above is external penalization. While at first this technique greatly simplifies the algorithm, it also introduces the additional problem of determining the adequate amount of penalization to be applied to external points. This problem turns out to be surprisingly difficult for the knapsack problem. The most adopted penalization heuristic for external solutions of the knapsack problem (that is, solutions with overlapping small items) is to apply a penalization based on the overlapping area of colliding small items. While this heuristic leads to very computationally efficient iterations of the optimization process, it has some significant shortcomings discussed in the following.

For illustration purposes, a very simple instance of a packing problem is considered: the task consists of packing two small items, a rectangle and a triangle inside a variable-length bin. The only considered variable is the position x of the triangle (see Fig. 1.(a)), the cost function $f(x)$ is the length of the smallest bin containing the two small items plus an amount proportional to the overlapping area:



(a) Two small items with overlap. (b) Graphic of the cost function.

Figure 1. (a) a relaxed packing problem with non-valid optimal solution. The overlapped area is given by $(h+b-x)^2h/2d$, the cost function is given by $f(x)=x+(h+b-x)^2h/2d$ when there is overlap, and by $f(x)=x$ when no overlap exists. (b) shows the graphic of cost function ($h=5$, $b=2$, $d=3$), it is possible to observe that the minimum happens when some overlap is present.

$$f(x)=\begin{cases} x & \Rightarrow x > b+h \\ x+(h+b-x)^2\dfrac{h}{2d} & \Rightarrow x \le b+h \end{cases} \quad (1)$$

For this problem, a penalization based on the overlapping area can lead to an invalid optimal solution as shown in Fig. 1.(b) where the minimum corresponds to a situation with overlap.

External penalization based on overlapping length (informally defined as the length of the shortest translation that, when applied to an overlapped polygon, eliminates its overlap) would in theory eliminate this problem (Heckman and Lengauer, 1995). However, the calculation of the overlapping length turns out to be computationally expensive for non-convex polygons. Another drawback is that those heuristics make the implicit assumption that the only class of modification that may be applied to a small item is a translation (the overlapping length being the estimative for the length of the translation that removes the overlap). This makes such heuristics to perform poorly on rotational problems where, in several overlapping instances, a large translation may be necessary to remove an overlap that may be eliminated with a small rotation.

A good discussion of external penalization techniques can be found in the work of Heckman and Lengauer (1995), where problems very similar to those studied here are approached with simulated annealing. One characteristic of the presented solution is that the optimization process may result in solutions including collisions between small items, thus requiring a post-processing step of the obtained data.

The approach adopted here avoids the pitfalls of external penalization by the continuous mapping (i.e. it is updated at each step of the process) of the complex space of valid solution onto a simplified space. Although this additional mapping step increases the complexity of the algorithm, it confers to the algorithm a more universal character, as there is one less empiric parameter to be defined.

Actually, the proposed algorithm does not explore the whole space of possible solutions, focusing instead on a reduced space, that contains at least one optimal solution. It is the space of connected small items and large object produced by the constructive approach and no-fit polygon. One can observe that it is possible to construct a connected solution for any and from any non-connected solution of a placement problem that allows free translation without increase in cost. This reduces the search through irrelevant points and enhances the performance of the algorithm.

Simulated Annealing

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

Simulated annealing originates in the Metropolis algorithm, a simulation of the recrystallization of atoms in metal during its annealing (gradual and controlled cooling). During annealing, atoms migrate naturally to configurations that minimize the total energy of the system, even if during this migration the system passes through high-energy configurations. The observation of this behavior suggested the application of the simulation of such process to combinatorial optimization problems (Kirkpatrick et al., 1983).

Simulated annealing is a hill-climbing local exploration optimization heuristic, which means it can skip local minima by allowing the exploration of the space in directions that lead to a local 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 from the Boltzman distribution

$$P(\Delta E) = e^{-\frac{\Delta E}{kt}} \tag{2}$$

where $P(\Delta E)$ is the probability of the optimization process keeping a modification that incurs 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. 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.

Algorithm

Consider the problem of minimizing a 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^* in 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 a lower cost), the new solution is automatically kept. If this increase is positive (meaning the new solution has a greater cost), a random number r is uniformly generated between 0 and 1. If r is smaller than the probability calculated by (2), x^* replaces x as the new current solution. Otherwise, x^* is discarded.

As can be seen in Fig. 2, the algorithm executes iterations at a fixed temperature until a specific stop condition is 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. When either of the conditions is met, the algorithm proceeds to the next temperature of 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. Typically, additional conditions are defined which stop the algorithm before the cooling schedule ends. A common such condition is a maximum number of iterations allowed during which the algorithm performs no significant progress (a situation known as "frozen state").

```
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* ← PickRandomNeighbor(x)
8:       ΔE = F(x*) − F(x)
9:       if ΔE < 0 then
10:         x ← x*
11:      else
12:         r ← <random uniform number between 0 and 1>
13:         if r < e^{−ΔE/(k·t)} then
14:            x ← x*
15:         end if
16:      end if
17:   end while
18: end while
19: <Display solution x>
```

Figure 2. The simulated annealing optimization algorithm.

Simulated Annealing Parameters

The success and efficiency of the simulated annealing process depends significantly on the careful construction of a cooling schedule and the definition of the appropriate initial temperature. Geometric cooling is used in this work, since it often leads to good results, by allowing the system to settle at an optimal configuration as the temperature falls. In geometric cooling, the value for the temperature is given by $t_i=\alpha\, t_{i-1}$. The parameter α is usually chosen around 0.95, while the determination of an appropriate initial value remains a difficult problem. The initial temperature must be chosen high enough such that algorithm does not get stuck in a sub-set of solutions and all the solution space is explored in the initial phase with high temperatures (the values of ΔE of the cost function are very high). One must notice thought that there is a saturation of the effect of $T_{initial}$ on the initial behavior of the algorithm. For higher values of T, the probability of acceptance of newly explored solutions is very close to 1, not increasing further with T. So, excessively high initial temperatures do not increase the effectiveness of the initial space exploration, but only increase the duration of the cooling schedule. A proposed heuristic to determinate the initial temperature determination (Heckman and Lengauer, 1995) is $T_{initial} = 3\ \sigma E / ln(P)$, where σE is the standard deviation of the cost function obtained through some iterations of the algorithm, and P is the desired probability of acceptance of initial solutions, typically selected between 0.85 and 0.5.

The Proposed Algorithm

Local Exploration meta-heuristics such as simulated annealing must evaluate a large number of solutions to ensure the quality of the obtained results. This implies that the basic operations of such algorithms must be efficiently performed. The no-fit polygon is used to efficiently avoid overlapping among the small items as well as to place them inside the large object. The proposed algorithm is a constructive approach with parameters controlled by the simulated annealing: the rotation applied to each small item and its placement. Three possibilities to define the sequence with which the small items are placed are studied: larger first, random permutation and weight sorted.

No-Fit Polygons and Collision-Free Region

In robot motion planning this concept is also known as Minkowski sums and configuration space obstacle (Latombe, 1991). Burke et al. (2006) present a comprehensive description of an algorithm for the no-fit polygon generation. Minkowski sums can be calculated very efficiently for convex small items. The result of a Minkowski sum of two convex small items is a convex polygon built from the edges of the original small items (notice this construction is unique). Non-convex small items can be decomposed into convex polygons in a pre-processing step, as the transformations applied (rotations and translations) do not affect such a decomposition.

In this publication the concept of an inner-fit polygon is used, which is derived from the no-fit polygon and represents the feasible set of points for the placement of the small item inside a large object. The inner-fit polygon can be computed by sliding a small item along the internal contour of the large object (Dowsland et al., 2002).

The no-fit polygon induced by fixed small item i to the moveable small item j is represented as NFP_{ij}. The inner-fit polygon induced by the large object to the moveable small item j is represented as IFP_j. The collision-free region for a given small item j is obtained by the following equation:

$$CFR_j = IFP_j - \bigcup_{i \in sfsi} NFP_{ij} \tag{3}$$

where *sfsi* is the set of fixed small items. The collision-free region can result in a set of multiple disconnected polygons. The polygons that represent the collision-free region are named in this paper, as region.

Small Item Placement

The placement of a small item is controlled by parameters (θ,r,f). Those parameters are translated into the rotation applied to each small item and its relative position to the large object and small items already placed. The rotation is straightforwardly derived from the parameter θ of the small item. The relative position to the collision-free region is derived from r and f using a more complex mapping as follows.

Parameter f defines which of the many possible regions from the collision-free region to place the small item in. For that, the m available collision-free regions are sorted according to some criteria (in this work they are sorted by the x coordinate of their leftmost vertex). The index j in this sorting of the particular region where the small item will be placed is derived from parameter f as follows:

$$j = \lfloor f \cdot m \rfloor \tag{4}$$

This is equivalent to assigning each region in the sorting to the interval [j/m, $(j+1)/m$), where j is the zero-based index of the region in the sorting. As f is defined in the interval [*0,1*), the shape to which the small item is assigned is determined by the interval which contains f.

To illustrate the selection of regions, consider following example for a particular small item at a given moment of the process where there are four collision-free regions A_1, A_2, A_3 and A_4. The region sorting produced the sequence (A_2, A_1, A_3, A_4). Then, the small item shall be placed in the region:

$$\begin{array}{ll} A_2 & if \quad f \in [0,1/4) \\ A_1 & if \quad f \in [1/4,1/2) \\ A_3 & if \quad f \in [1/2,3/4) \\ A_4 & if \quad f \in [3/4,1). \end{array} \tag{5}$$

As the shape and number of regions in the collision-free region change with the position of the small items already placed inside the large object, the points on each region's perimeter (where the new small item will be placed) are mapped onto a normalized variable $r \in [0,1)$. This is done by picking a reference point on the perimeter (in this work this point is the leftmost point in the region). This reference point corresponds to the placement point for $r=0$. From this point, the perimeter is traced counterclockwise and its points are mapped uniformly onto the interval [*0, 1*).

Generation of Placement Solutions

The initial solution is generated at random. At each step, a new solution is generated from a transformation of the preceding solution. As defined by the simulated annealing algorithm, this new solution is picked from the neighborhood of the preceding solution. The process is governed by a neighborhood heuristic.

A single small item is randomly selected and only one of its parameters (θ,r,f), also chosen at random, is randomly modified. As all the parameters (θ,r,f) are defined in [*0,1*), the modification consists in adding a random number Δ generated in [*-1/2,1/2*) and applying a modulus 1 arithmetic to the result.

Rejected solutions do not contribute to the progress of the optimization process. Therefore, the distributions of the individual Δ parameters 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 distribution of Δ for that specific parameter is modified in order to have its standard deviation reduced (resulting in a lower modification amplitude). When the modification leads to an accepted solution, the distribution of Δ for that parameter is modified to increase its standard deviation (resulting in a larger modification amplitude).

This work proposes to continuously map a subset of the whole space of feasible solutions onto a simplified space. The subset of feasible solutions considered by the process is the subset of *connected solutions*. For each small item to be placed, its collision-free region is calculated. The small item is then placed on the perimeter of this region, ensuring thus that it will always be connected to at least one small item or large object.

Heuristics for the Sequence of Placement

The sequence with which the small items can be placed is determined by one of the following heuristics given as examples:

- Larger-first: this deterministic selection heuristic is used in combination with bottom-left placement heuristics. It sorts the small items by area and places the larger small items first. This heuristic usually leads to satisfying results, but like with all deterministic heuristics it is possible to build problem instances for which this heuristic *never* reaches the global optimum. The larger-first heuristic is used in Dowsland et al. (2002).
- Random Permutation: this probabilistic heuristic introduces a randomized small item selection step to the simulated annealing. With a given probability, instead of changing the small item parameters, the simulated annealing algorithm swaps the placement order of two small items. This new solution candidate is evaluated in the same way as a regular simulated annealing solution, and accepted or

rejected analogously. The random permutation is used in literature by Gomes and Oliveira (2006).

- Weight sorted: this heuristic adds a new parameter *w* to the small item parameters (θ,*r*,*f*). The *w* parameter is also taken on the interval [*0,1*), and generated by the simulated annealing similarly to the known parameters. The placement sequence is such that the small items with larger *w* parameters are placed first. This heuristic turned out surprisingly difficult to adjust (the convergence of the algorithm becomes very dependent of the chosen magnitude for the weights), and leads to unstable results.

Handling Non-Placed Small Items

Herz and Widmer (2003) proposed general guidelines for applications of local-search meta-heuristics to combinatorial optimization problems. One of the proposed guidelines is: "The topology of the space induced by the cost function must not be too flat". In other words this means that the existence of contiguous solutions with identical costs is harmful to the algorithm performance.

To eliminate those cost collisions, the cost of a given solution can be modified in order to reflect how close this solution is to having a non-placed small item fitted in the large object. In the proposed solution, for each non-placed small item, a limited-depth binary search is performed to find a scale factor (between 0 and 1) that, when applied to the small item, would allow it to be fitted in the large object.

The described handling scheme for non-placed small items induces relatively high costs since each additional search level requires for every non-placed small item an additional scaling and placing operation. Actually, since only the best scale factor is relevant, the search may be aborted for small items whose scale level is known to be lower than one previously found for another small item.

Function Cost Evaluation

The simulated annealing algorithm produces at each iteration a solution for the problem composed of

- A sequence of placement for the small items,
- For each small item, the parameters (θ,*r*,*f*).

To evaluate the cost function for this solution, the small items are placed sequentially in the large object.

The placement process of a single small item occurs in the following sequence (see Fig. 3):

- The rotation θ is applied to the small item,
- The collision-free region inside the large object is calculated considering the already placed small items,
- The parameter *f* is used to determinate in which region of the collision-free region the small item shall be placed,
- Using the parameter *r*, the small item is placed on the perimeter of the chosen region.

When the collision-free region has no sufficiently large area for a small item, it is not placed in the large object. Then, the limited-depth binary search is performed to find a scaling factor that, when applied to the small item would allow it to be fitted in the large object.

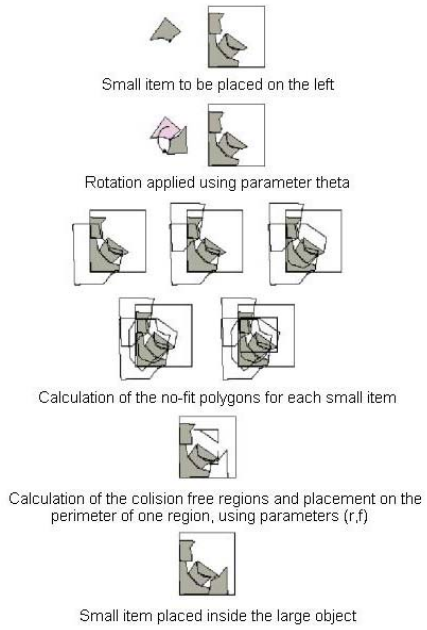


Figure 3. Example of placement of a single small item during the optimization process.

Results

All problem instances studied here have a solution where all small items can be fitted in the large object. That allowed the adoption of artificial stop conditions to simplify the study. On all problems, the large object area is *10%* larger than the total area of the small items. The optimization method was implemented using a modified version of the *PolyBoolean* library. The random-number generation uses the Mersene-Twister generator (Matsumoto and Nishimura, 1998). A simple geometric cooling with α*=0.95* was adopted. The binary search was executed with a fixed depth equal four (leading to 16 possible scale levels). The placement sequence was produced by the *larger-first* heuristic.

The behavior of the optimization process is illustrated through cost function (energy) histograms of the search while the temperature diminishes. For a given temperature, a gray-level histogram of the distribution of the cost function at that temperature is plotted. The resulting graph shows a plot of cost histograms (horizontal bars) and temperature (dots) *versus* the number of iterations. Darker horizontal bars in the histogram, indicate a higher frequency of occurrence of a particular level of energy at a given temperature (see Fig. 4). ´

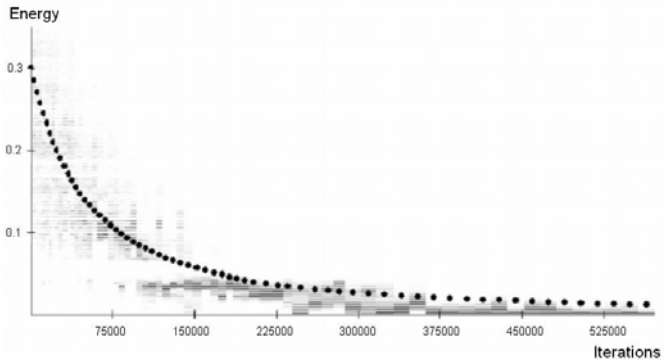


Figure 4. Visualization of the simulated annealing macro-properties through sequences of temperatures and cost function histograms (a generic optimization process is shown).

One can notice in the Fig. 4 that as the temperature decreases, the number of accepted solutions per iteration decreases (as more solutions are rejected). Consequently, the number of iterations spent at each temperature increases, effectively leading to a slower cooling schedule.

When evaluating the algorithm performance from the obtained results, one must take into account the fact that usually, a solution as good as the final one is found in far fewer iterations (on the results presented here it happens at *1/4* or less of the total iterations) than it takes for the algorithm to converge. Of course, letting the algorithm take its course is the only generic way to know if any previously found solution will be the best found, but this suggests that an algorithm that keeps track of the best found solution may be *interrupted* and still return a satisfactory solution.

Small Puzzle

The first example is a fairy simple puzzle with four non-convex non-congruent small items. The non-convex small items are decomposed into convex polygons in a pre-processing step. This decomposition does not affect in the final solution. Figure 5 shows the final solution of this problem and Fig. 6 the cost-function histograms. Based on the cost-function histograms, two distinctive phases of the process can be recognized. The first phase is characterized by a energy level higher than *0.15*. The second phase begins after *25000* iterations and has a energy level smaller than *0.15*. Convergence occurred after *350000* iterations, but the best solution was found much earlier, around *25000* iterations (about four minutes on a 2 GHz Pentium 4 processor).

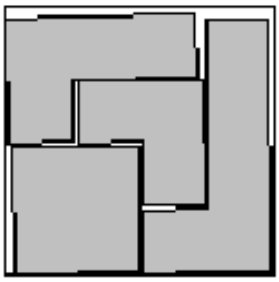


Figure 5. Final solution of a small puzzle with four small items.

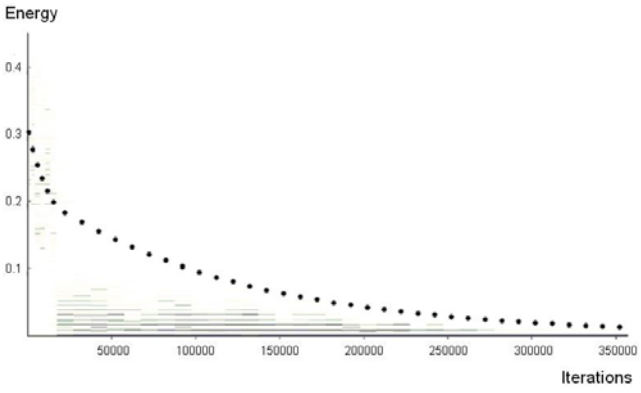


Figure 6. Cost-function histograms for the small puzzle.

Tangram Puzzle

The tangram puzzle consists of the placement of seven convex non-congruent small items. Figure 7 shows the final solution of this problem and Fig. 8 the cost function histograms. In this nesting problem, the simulated annealing algorithm encounters a phase transition when the two greater triangles settle at their final position. The convergence occurred after *525000* iterations, but the final solution was found at around *135000* iterations (about twenty-five minutes on a 2 GHz Pentium 4 processor).

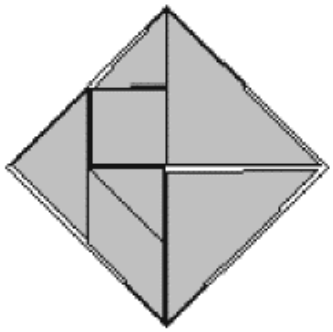


Figure 7. Final solution of a tangram puzzle with seven polygons.

As can be seen in Fig. 8, the temperature is lowered as the cooling schedule progresses, leading the system to lower energy states. The transition between states is not smooth. Two distinctive phases can be recognized. This corresponds to a macro-organization of the system, when the two larger triangular small items settle at their final configuration, leaving the position of smaller small items to be defined.

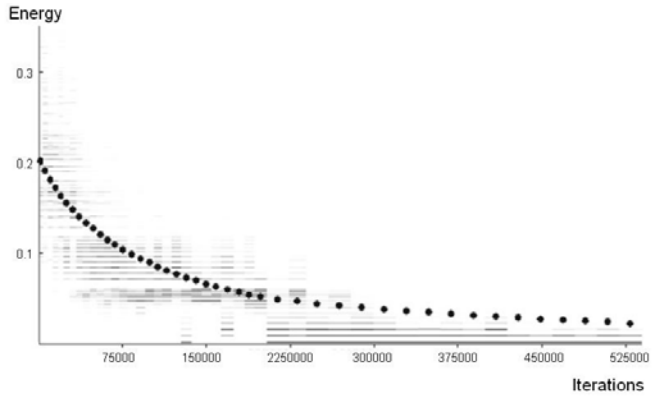


Figure 8. Cost-function histograms for the tangram puzzle.

As presented in section *Simulated Annealing Parameters*, the geometric cooling parameter α defines how the temperature falls according to $t_i = \alpha\, t_{i-1}$. The cost function histograms shown in Fig. 8 were obtained using $\alpha = 0.95$. An instance of the tangram puzzle was processed with $\alpha = 0.90$, leading to a faster cooling. As can be seen in Fig. 9, while the process takes less iterations to converge, it does *not* converge to the global optimum, getting "frozen" at a suboptimal configuration.

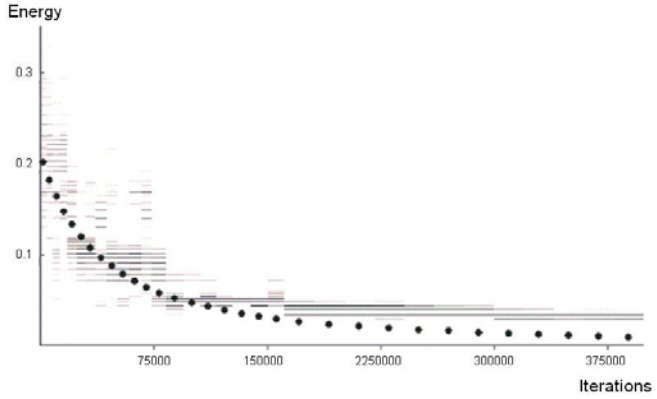


Figure 9. Cost-function histograms for a "fast cooling" of the tangram puzzle.

Influence of the Search Depth when Sorting Solutions

The effect of the modification of the cost function described in section *Handling Non-Placed Small Items* can be seen in Figs. 10 and 11 for the problem described in section *Small Puzzle*.

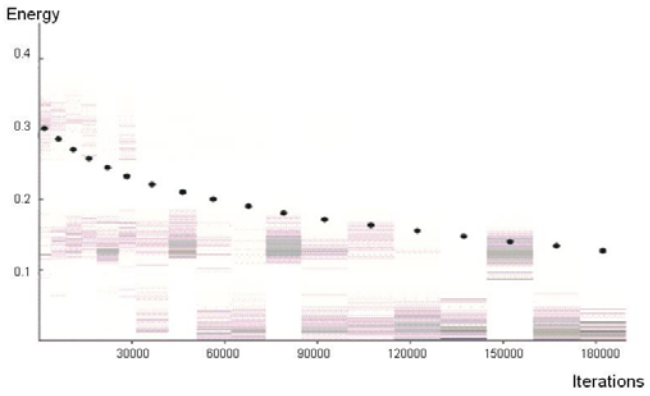


Figure 10. Energy histogram for 256 levels of scale search. Notice the almost continuous behavior of the cost function.

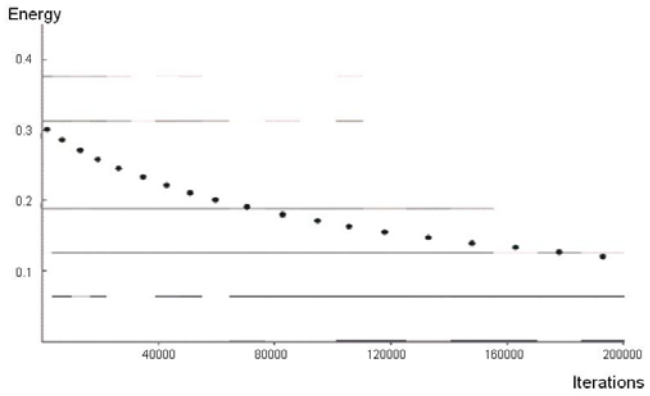


Figure 11. Energy histogram for 2 levels of scale search. The discrete behavior of the cost function is evident.

When the number of search levels is increased, the convergence is smoother and reaches optimal solutions in fewer iterations. This can be explained by the reduction of the gap between the discrete levels the cost function may assume. As mentioned previously, one of the main difficulties of the placement problem with fixed large objects is the fact that the cost function assumes only discrete levels, while the optimization variables are continuous.

There is of course a tradeoff to be considered, as the computational cost of each iteration increases with the number of search levels. For this particular (and extreme) instance, while the process with 256 levels reached optimal solutions in fewer iterations, the process with 2 levels reached optimal solutions in *less time*, as each iteration was much faster.

Influence of the Placement Order Heuristic

For the problems above, the *larger-first* deterministic heuristic had a much better performance than the *random permutation* and *weight sorted* probabilistic heuristics. In particular, for the *tangram puzzle*, the *weight sorted* heuristic displayed an unreliable behavior (often it converged to a sub-optimal configuration).

The better performance of the deterministic heuristic can be easily explained by the fact that it does not spend additional iterations to find the proper placement order. The popularity of the *bottom-left* heuristic can be understood by its low computational cost. Besides that, by naturally nesting the small items next to the walls of the large object (this is an heuristic applied almost exclusively to rectangular large objects), it keeps a single large unobstructed area (in opposition to several small free areas), where a larger variety of small items may be fitted. The deficiency of this approach is that there is no guarantee that for every placement the *bottom-left* heuristic will lead to an optimal solution.

For instance, one can easily build a problem instance where the combined *bottom-left* and *larger-first* heuristics cannot reach an optimal solution. The problem instance "Framed Square" shown in Fig. 12.(b) illustrates the deficiency of this deterministic heuristic. In this example, the larger piece (the square) must *not* be placed first, as that will produce only configurations with the square in contact with the large object. This simple instance of the placement problem is easily solved by both the *weight sorted* and the *random permutation* heuristics, but its optimal solution is unattainable by the *larger-first* heuristic, as illustrated in Fig. 12.(a). It is possible as well to produce instances where *larger-first* fails for other deterministic placement heuristics.

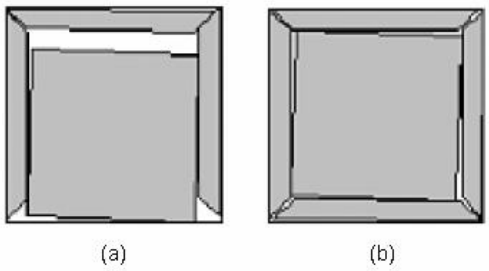


Figure 12. (a) Problem instance whose optimal solution cannot be reached through larger-first and bottom-left heuristics combined. (b) Problem instance solved with a placement order produced by the simulated annealing.

Conclusions

The placement problem deals with the task of minimizing the waste of space that occurs on a rotational placement of a set of irregular bi-dimensional small items inside a bi-dimensional large object with fixed dimensions. The survey made by Wascher et al. (2005) classified only 5 papers as related to the specific subject

studied here. However, these 5 papers describe exclusively translational approaches. One of the main difficulties of the placement problem with fixed large objects is the fact that the cost function assumes only discrete levels, while the optimization variables are continuous.

Herz and Widmer (2003) proposed general guidelines for the application of local-search meta-heuristics to combinatorial optimization problems. One of the proposed guidelines is: "The topology of the space induced by the cost function must not be too flat". To avoid the resulting cost collisions, the cost was modified in order to estimate how close the current solution is to having a non-placed small item fitted in the large object. To do so, a limited-depth binary search is performed for each non-placed small item to find a scaling factor (between 0 and 1) that, when applied to the small item, would allow it to be fitted in the large object. The search depth must be chosen with care, it must be not too large as the cost of each iteration increases. Satisfactory values of the scaling factor were determined for the studied problems.

The usual approach to simulated annealing applied to the kind of complex spaces discussed here is external penalization. While at first this technique greatly simplifies the problem, it also introduces the additional difficulty of determining the adequate amount of penalization to be applied to external points. The most common penalization heuristic for external solutions of the kind of problem studied here is to apply penalization based on the overlapping area of colliding polygons. While this heuristic leads to very computationally efficient iterations of the optimization process, it can lead to an invalid optimal solution. The external penalization is avoided by using no-fit polygons. The layout is created in a constructive approach, where the small items are sequentially positioned.

The placement of a small item is controlled by the following simulated annealing parameters: the rotation applied and the placement of small items. Since rejected solutions do not contribute to the progress of the optimization process, the distribution is adapted to increase the number of accepted solutions. The progress of the simulated annealing algorithm is visualized through cost-function histograms.

References

Art, R. C., 1966, "An approach to the two-dimensional irregular cutting stock problem", IBM Cambridge Scientific Centre, Report 36-Y08.

Bennell, J. A. and Dowsland, K. A., 2001, "Hybridizing tabu search with optimisation techniques for irregular stock cutting", *Management Science*, Vol. 47, pp. 1160-1172.

Burke, E. K., Hellier, R. S. R., Kendall, G., Whitwell, G., 2007, "Complete and robust no-fit polygon generation for the irregular stock cutting problem", *European Journal of Operational Research*, Vol. 179, pp. 27-49.

Dowsland, K. A., Vaid, S. and Dowsland, B. W., 2002, "An algorithm for polygon placement using a bottom-left strategy", *European Journal of Operational Research*, Vol. 141, pp. 371-381.

Dyckhoff, H., 1990, "A typology of cutting and packing problems", *European Journal of Operational Research*, Vol. 44, pp. 145-159.

Fowler, R. J., Paterson, M. and Tanimoto, S. L., 1981, "Optimal packing and covering in the plane are np-complete", *Inf. Process. Lett*., Vol. 12, pp. 133-137.

Gomes, A. M. and Oliveira, J. F., 2006, "Solving irregular strip packing problems by hybridising simulated annealing and linear programming", *European Journal of Operational Research*, Vol. 171, pp. 811-829.

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

Herz, A. and Widmer, M., 2003, "Guidelines for use of meta-heuristics in combinatorial optimization", *European Journal of Operational Research*, Vol. 151, pp. 247-252.

Hifi, M. and Hallah R. M., 2003, "Hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes", *International Transaction in Operational Research*, Vol. 10, pp. 195-216.

Jakobs, S., 1996, "On genetic algorithms for the packing of polygons", *European Journal of Operational Research*, Vol. 88, pp. 165-181.

Kirkpatrick, S., Gellat, C. D. and Vecchi, M. P., 1983, "Optimization by simulated annealing", *Science*, Vol. 220, pp. 671-680.

Latombe, J. C., 1991, "Robot Motion Planning", Kluwer Academic Publishers.

Leonov, M. V., 2004, Polyboolean library. http://www.complex-a5.ru/polyboolean/.

Matsumoto, M. and Nishimura, T., 1998, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator", *ACM Trans. on Modeling and Computer Simulation*, Vol. 8, pp. 3-30.

Wascher, G., Haussner, H. and Schumann, H., 2007, "An improved typology of cutting and packing problems", *European Journal of Operational Research*, Vol. 183, pp. 1109-1130.