The CSI Journal on
Computer Science and Engineering
Vol. 6, No. 2 & 4 (b), 2008
Pages 58-67
**Regular Paper**

# A Novel Evolutionary Approach for Two Dimensional Bin Packing

Ramin Halavati[1] Saeed Bagheri Shouraki[2] Saman Harati Zadeh[1]

[1]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[2]Electrical Engineering Department, Sharif University of Technology, Tehran, Iran

## Abstract

Packing problems arise in a wide variety of application areas. The basic problem is that of determining an efficient arrangement of different objects in a region without any overlap and with minimal wasted gap between shapes. This paper presents a novel evolutionary approach based on three new evolutionary operators for optimizing the arrangement of irregular shapes. In this approach, each chromosome represents a partial solution and the novel evolutionary operators take care of the evolution of more fitting sub-solutions until a complete suitable solution emerges. The approach is tested on standard benchmarks and presents comparable results with current problem specific approaches and notable better results in compare with traditional Genetic Algorithm.

**Keywords:** Two Dimensional Bin Packing, Evolutionary Approach, Soft Computing, Optimization.

## 1. Introduction

Packing problems arise from a variety of situations including pallet loading, textile cutting, container stuffing and placement problems. Such optimization problems are concerned with finding a good arrangement of multiple objects (2D or 3D) in a larger containing region without overlap. The usual objective of the allocation process is to maximize the material utilization and hence to minimize the wasted area. The input of such problems is usually a set of *n* shapes (2D or 3D) and sometimes some limitations on the sizes of the target plane or space. The algorithm must give a positioning of the given shapes, so that primarily they would not overlap and secondarily span over the minimal total area to minimize the wasted gaps between them.

A wide range of AI approaches have been developed to find acceptable solutions for this class of problems. One of the major themes in these approximate solutions is evolutionary approaches and more specifically genetic algorithms. In a conventional GA, each genome represents a complete solution or the instructions for producing the complete solution using a greedy approach. The task starts with generating a set of random genomes, and then, the mutation operator creates more variety in the gene pool and the crossover operator replaces parts of genomes, hoping to substitute partial good answers among different solutions.

A known weakness of such approaches is in the blind crossover operator: If a genome includes some partial solutions, these may-be-useful findings may be lost as the crossover operator splits the genome from points that are not necessarily related to the boundaries of the partial good answers [14]. On the other hand, observing the behavior of a human in solving such problems shows that partial solutions usually have a key role in decreasing the size of search space because once a group of matching parts is found, the whole group is considered as one single item and then it is combined with other parts to find more general answers.

To overcome this weakness, this paper presents a novel approach for optimization tasks such as 2D bin packing which is based on defining partial solutions as genomes. In each iteration, two sub-solutions with no mutual conflict are merged and create a more complete answer. Once a complete

solution is composed, it is replicated based on its general fitness, and then, all copies break again to sub-solutions. The rest of the paper is organized as follows: The next section presents a literature study; Section three presents our problem modeling approach; Section four presents the optimization process and the fifth section presents a discussion on the suitability of this approach followed by experimental results in section six; At last come the concluding remarks.

## 2. Background

The packing problem becomes much simpler when both objects and the containing region are rectangular in shape. Many research works have been done on two and three dimensional rectangular packing problems. However, in many practical applications, objects and containing regions may have irregular shapes. Due to the geometrical complexity introduced by irregular shapes, such problems are not as well studied as rectangular packing. There are relatively few works on two dimensional packing with arbitrary shapes, compared to 2D rectangle packing. They can be classified into two types: nesting [7] and packing [13] [19].

Nesting is an approach in which irregular objects are nested in simpler regular shapes, and these simpler shapes are then packed into an available area [18,20]. Rectangle is the most popular shape for nesting. Freeman and Shapira [12] present an approach based on enclosing irregular shapes in a convex polygon and then finding the required rectangle by iteratively basing the rectangle on each of the polygon edges. In practical cases, these type of solutions prove satisfactory only if the pieces themselves are close to rectangular shapes so that the wasted area is small. Adamowica and Albano [1] proposed another approach by nesting more than one piece in one rectangle when pieces are far from rectangular shapes. Another alternative for using rectangle packing is to nest all the pieces into identical polygons which can be used to tile the plane. These include triangles, quadrilaterals, pentagons and hexagons. Dori and Ben-Bassat [9] based their packing algorithm on using hexagons. They nested the required shapes into a polygon which minimizes the wasted area and then found a covering hexagon for each polygon, which would be suitable for tiling the plane. However, their approach is limited to convex polygons. Similar approaches can be seen in [11] and [27].

Straightforward single pass packing strategies involve taking the pieces in order and placing them on the stock-sheet according to a given placement policy. This can be repeated several times for different object orders or different placements and the best solution would be selected. Another approach is to use a more intelligent method in a single pass. Qu and Sanders [22] first sorted the pieces in decreasing length order and then placed along two adjacent edges of the stock-sheet. Dowsland and Dowsland [10] used a leftmost placement policy together with a random ordering of the pieces. Albano and Sapuppo [2] were concerned with the more complex problem in which the pieces may be placed in a number of different orientations. The same methods can be seen in [4], [26], and [27].

There is another approach which has become increasingly popular in recent years. It is to produce an initial layout and then to use small changes in order to improve it. Such an approach may either continuously seek for improvement, or may incorporate meta-heuristic techniques such as simulated annealing or tabu search in order to allow up-hill, or non-improving moves. Blazewicz and Walkowiak [3] suggested a tabu search algorithm. Sakait and Hae [25] applied genetic algorithms. They discretized the object and represented the object as a linked list of cells. The linking from one cell to the next is given in the form of an eight connectedness. Thus this approach is applicable to any arbitrarily shaped object. The total layout space is divided into a finite number of cells for mapping it into a new 2D genetic algorithm chromosome.

The genetic operators of mutation and crossover have been suitably modified to suit this problem. Keishi et al [17] developed an algorithm to handle convex-rectilinear blocks by enhancing the Bounded Slice-line Grid based packing algorithm. Maggie and Wayne [20] made use of the Sequence-Pair structure. Each rectilinear shaped block is partitioned into a set of rectangular sub-blocks, each of them individually handled in a sequence pair as a unit block. The horizontal and vertical graphs are constructed. Packing can be obtained by applying the longest path algorithm on the graphs. Chen et al. [6] have used several heuristic greedy methods to pack irregular shapes.

## 3. Problem Modeling

This section presents our problem modeling approach. The input of the algorithm is a set of polygons which must be arranged on a sheet, with no overlap and with minimal surrounding rectangle. This set of shapes is called ***Original Shapes Set***.

Each ***gene*** represents the relative relation between a positioning of two of the original shapes. Similar to almost all other algorithms, we have restricted the relations to a subset of possible relations, which are the cases where the two shapes have one common vertex, one common edge and no overlapping parts. Figure 1 presents the possible relations between two sample shapes.

A ***chromosome*** is a combination of some genes, which results in a partial or complete solution to the problem. A chromosome can be build up only if its genes create a set of connected shapes, with no overlap. Figure 2 presents a sample gene and chromosome. A ***complete chromosome*** is a chromosome with exactly one instance of each of original shapes.

The ***fitness*** of a complete chromosome is defined as the ratio of the sum of shapes areas to the area of its minimal covering rectangle, as presented in Eq.(1).

$$\text{Fitness} = \frac{\sum_{S \in \text{Chromosome}} \text{Area}(S)}{\text{Area}(\text{Minimal}\textit{CoveringRectangle})} \quad (1)$$

## 4. The Optimization Process

The core idea of the optimization process is to generate a pool of partial solutions (chromosome pool), combine the partial solutions to make complete answers, and increase the population of solutions which contribute in better complete answers.

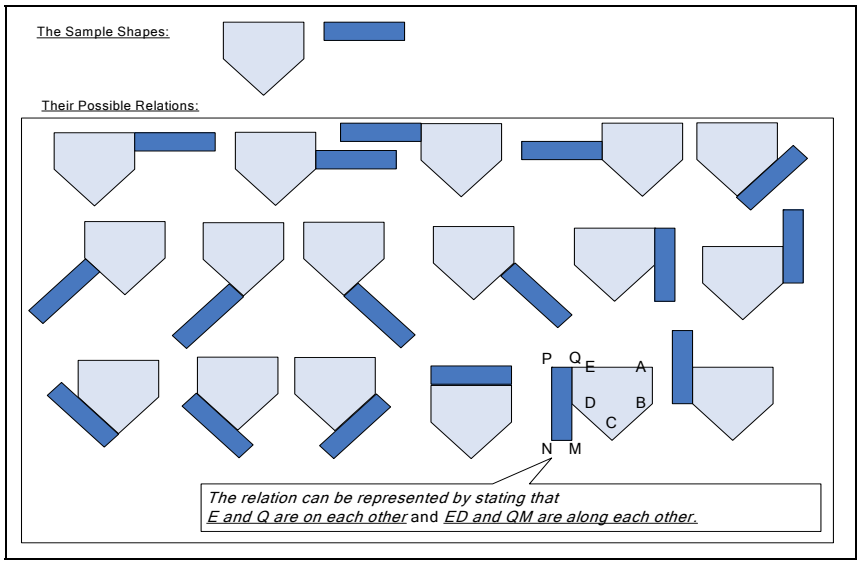


Figure 1. Possible relations between two sample shapes

This is done by using three operators: Selection, Combination, and Explosion where the selection operator chooses two members of the chromosome pool for combination; The combination operator takes two chromosomes and combines them with no overlap, if they have no common shapes and if the combination is a complete solution, gives it to the explosion operator; And the explosion operator replicates the given combination and then breaks all copies into smaller chromosomes or partial solutions. Figure 3 presents a general diagram of the process and the rest of this section presents the details.

## 4.1. The Selection Operator

The process starts by generating a set of random chromosomes, each including just one gene and repeating the following sequence of actions as long as the best result needs improvement.

At the beginning of each iteration, two chromosomes (namely C1 and C2) are picked up. Selection of the two chromosomes may be done using an arbitrary heuristic function, but to make it simpler, we have used random selection. The only restriction on the selection process is to choose C1 and C2 so that they satisfy one of the following assertions:

1. C1 and C2 have no common shapes.
2. C1 and C2 have some common shapes, but their non-common shapes have no overlap.

Asserting either of these two rules grants that the C1 and C2 can be combined by a proper selection of their relative position which is the subject of the next operator.

## 4.2. The Combination Operator

Once C1 and C2 are selected, they are removed from the pool and a new chromosome (C3) whose genes are the combination of C1 and C2's genes is generated. If C1 and C2 have at least one common shape, C3 will be connected. But if C1 and C2 have no common shapes, C3 will be composed of two separate trees of connected shapes. Therefore and to make C3 connected, a new random gene is added to C3 whose one side will be from one of C1's shapes and the other side will be from one of C2's shape. This random gene will be marked as ***connector***. The connector gene must be chosen so that the resulting combination would have no overlap. Figure 4 presents a sample of this stage.

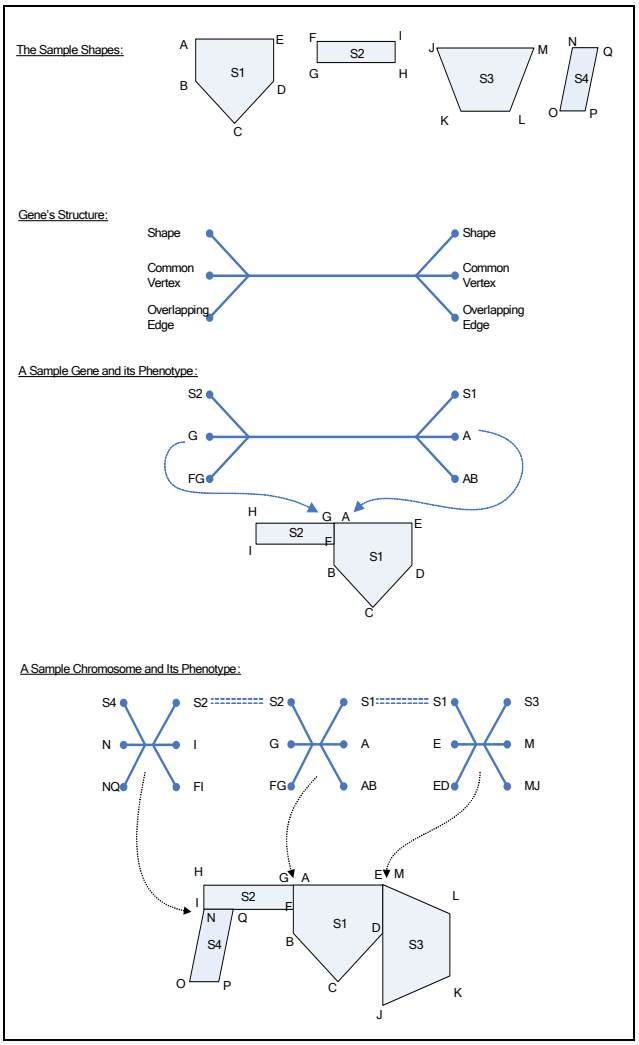


Figure 2. Gene and chromosome samples

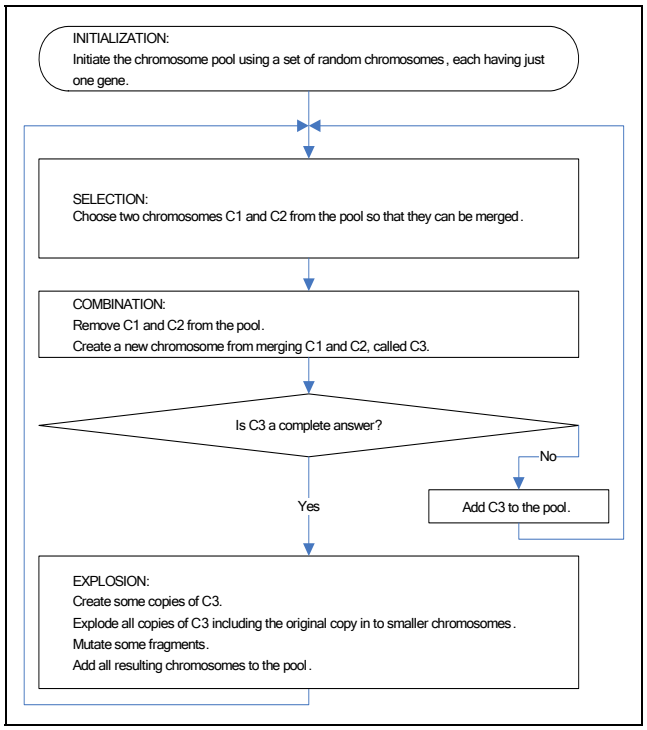


Figure 3. General diagram of the optimization process

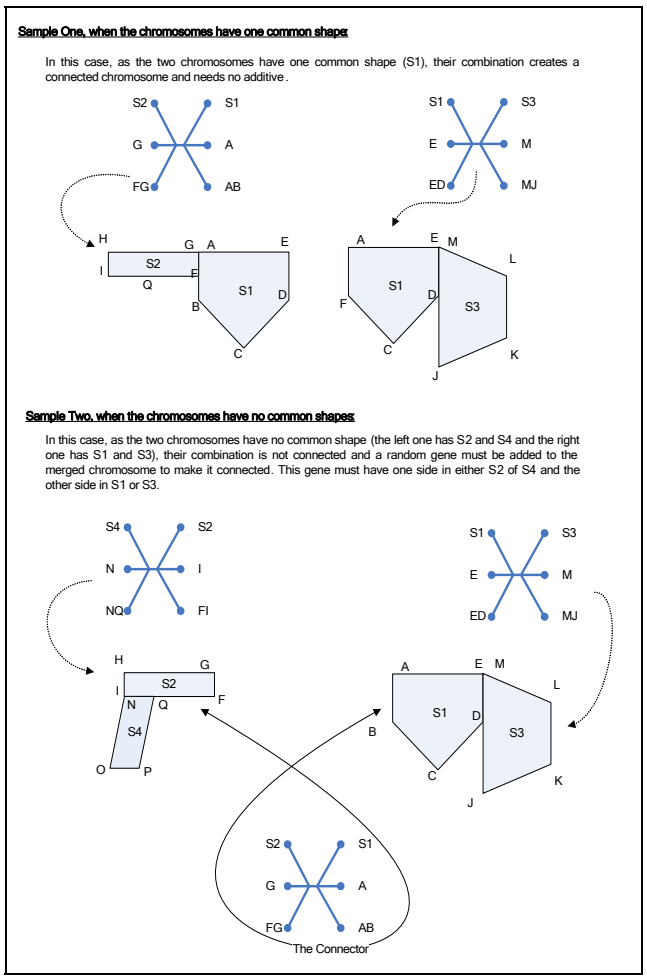


Figure 4. Samples for Combination operator

## 4.3. The Explosion Operator

Once the combination operator creates a new chromosome (C3), if the chromosome is not a complete solution (i.e., it misses one or more shapes), it is put back to the chromosome pool. But when it is complete, it is passed to the explosion operator. Using this operator, chromosomes with low fitness are eliminated and the ones with high fitness values are replicated and then all copies are broken into small chromosomes. This task is done as follows:

### 4.3.1. Computing the Relative Fitness (R.F.)

The fitness of a complete chromosome is computed as stated in section 3, Eq. (1). But to have a more dynamic force on the evolution, we use relative fitness as the ratio of the current chromosome's fitness to the average of the last best fitness values. This way, the force of fitness gradually changes, based on the average state of the evolution.

### 4.3.2. The Replications

Once R.F. is computed, the chromosome is replicated relative to its R.F. so that better chromosomes have more copies and the weaker ones have less. This number is computed as stated in Eq. (2).

$$\text{Number of Copies} = \left\lceil \frac{\textit{Relative Fitness - Base}}{1 - \textit{Base}} \times \text{Effectiveness} \times \text{Population} \right\rceil \quad (2)$$

The equation has two parameters, ***Base***, and ***Effectiveness***. The first parameter indicates the minimum R.F. required for replication and second parameter specifies how much this chromosome will affect the chromosome pool. If the value would be chosen higher, the number of copies resulting from a good result would be increased and the future search will be directed more towards this answer, but a lower selection will results in less emphasis on this finding and a more global search.

Chromosomes which gain an R.F. below ***Base*** are instantly eliminated.

### 4.3.3. The Breakings

When the chromosomes are replicated, all copies including the original ones break into smaller chromosomes. To break a chromosome, each of its genes with *connector* mark may change their state into *normal* with a certain probability which is called ***Sticking Rate*** and then, all remaining *connector* genes are removed. Chromosomes with no connector gene are eliminated as they cannot take part in the process anymore and those which have some connectors are splited into a number of connected chromosomes. With a certain low probability (***Mutation Rate***) each of the fragments may be changed a little, so that to represent a similar, but a little varying combination and then all fragments are put back into the pool. Figure 5 presents the diagram of Explosion operator. To mutate, we change just

one gene so that the relative position of only two shapes is changed.

# 5. Discussion

As stated in the previous section, the optimization begins by generating a pool of single gene chromosomes. During the process, chromosomes with more genes are created and each chromosome grows until it gains one member of each shape and becomes a complete solution. Once a complete chromosome emerges and it has a good relative fitness, i.e. it has smaller spanning area in compare to the latest results, it is replicated to create several similar copies. By the explosion of every copy of the complete solution, some smaller chromosomes are created while each of these small parts has previously shown that it is able to join other parts and create a good complete group. Therefore, it can be expected that the pool will gradually have more and more partial good solutions, i.e. solutions that can join others well and make a total good ensemble.

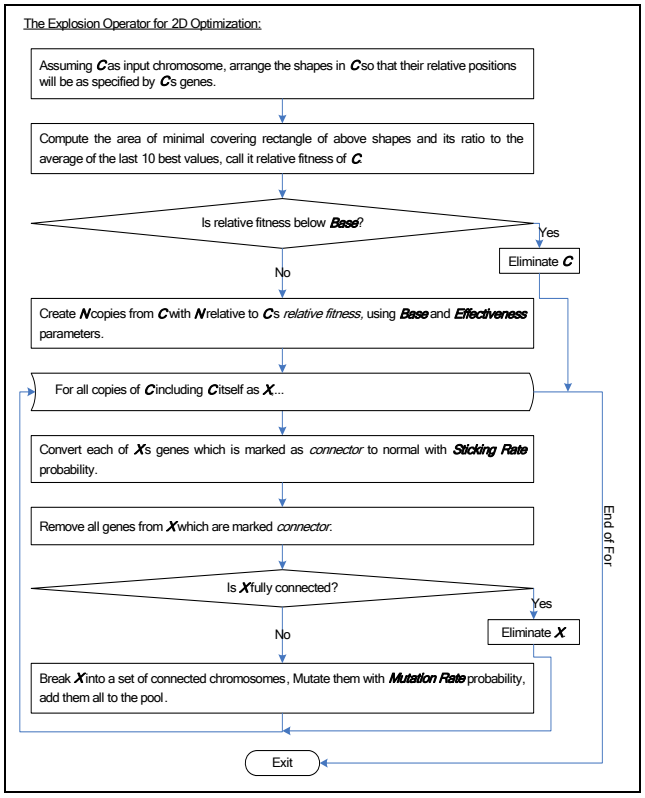


Figure 5. The diagram of Explosion operator

Repetition of joining and separating processes results in trial of different combinations of partial good solutions and therefore, the space of possible solutions is searched based on building blocks that gradually emerge. The major advantage of this approach in compare to a conventional search of states space, such as a genetic algorithm, is in the fact that here the process gradually creates some heuristics (the good partial compositions) based on its previous experiments. These heuristics can help the optimization process in searching limited areas of the search space. If the heuristic proves to be successful, it will stay and produce more copies of itself and if it fails, it is automatically and gradually removed. Therefore, although the search is not limited to a specific area, it is guided by its own experiment to search hopeful areas more. This property is analyzed in more details in section 6.4.

Four parameters affect this process: the ***base*** and ***effectiveness*** parameters control the replication, and ***sticking rate*** and ***mutation rate*** affect breakage. Base directly chooses the chromosomes which can affect the future of the process through filtering out those which have lower relative fitness values. Thus, choosing a lower *base* value results in letting the process search less promising areas more. It must be noted that as we have used relative fitness, the replication system's lower limit automatically increases while it finds better results and *base* is just a measure, stating the system's tolerance versus results which are worse than current average best results. This property is visualized in section 6.5.

The *effectiveness* parameter indicates the influence of a good answer on the pool. Choosing a high *effectiveness* results in increasing the number of fragments that are produced by one good answer and therefore, the system is guided more towards this solution. In other words, if a solution would be regarded as a heuristic, the *effectiveness* states how much we emphasis on using it. This phenomenon is tested and depicted in section 6.6.

The *sticking rate* is a measure on how fast we will emerge heuristics. With low *sticking rate*, the probability of creating big fragments from currently found solutions decreases and therefore, more repetitions of an answer are required to guide the system toward a big heuristic hence emergence of good results would be slower. On the other hand, choosing a high *sticking rate* results in faster generation of big fragments (heuristics) and this prevents the system from trial of smaller fragments in other compositions. Therefore the search would be more biased towards solutions that are very much similar to current results. More on this is presented in section 6.8.

By letting the process explore its neighborhood, the *mutation rate* parameter plays the same role in this system as in genetic algorithms. A low mutation rate results in more emphasis on combinations of currently found heuristics and therefore denser local search while a high value results in more global search. Tests and results on this parameter are presented in section 6.7.

# 6. The Experimental Results

We run eight sets of tests on the proposed algorithm. The first set was to find the appropriate parameters for the algorithm, the second set was to compare the results with the general benchmarks and the third was to make a detailed comparison with pure genetic algorithm as it is the base of most cited benchmark results such as [8], [15], and [16] where in all such approaches, G.A. is the basic search strategy but it is guided with a set of heuristics. Tests four to eight were done
to prove the parameters analysis, which was presented in section 5.

As test problems, we used M1, M2, M3, T1a-T7a from [15], Dighe1 and Dighe2 from [8], K from [5], D1-D4 from [23] and [24] and J1 and J2 from [16]. The algorithm is implemented in Microsoft Visual C++ 6.0 with standard optimizations and run on a Pentium IV 1800 MHz.

## 6.1. Parameter Tuning

To find the appropriate set of parameters (base, effectiveness, sticking rate, and mutation rate), we generated 1000 random settings in the valid parameters range and chose 5 random problems from the benchmarks. The algorithm was run 10 times with each setting and on each problem and each run was stopped until either the fitness of the best result exceeded a certain threshold or the elapsed time exceeded another threshold. Both thresholds where chosen separately for each problem and based on problem's complexity.

To represent algorithm's performance with each settings, a cost measure is defined in Eq.(3) as follows: The average iterations count for each pair of setting-problem (*AvgSP(s,p)*), and each problem regardless of settings (*AvgP(p)*) is computed; As the ratio of AvgSP(s,p) to AvgP(p) represent the relative performance of setting s in optimizing problem p, in compare with other settings, the average of this ratio for all problems is regarded as settings' cost. Table 1 represents some sample results.

Using the specified Cost function, we sorted the settings and chose the highest which was:

Base: 0.87

Effectiveness: 0.12632

Sticking Rate: 0.3367

Mutation Rate: 0.029

Noting that these settings are not necessarily the best possible settings and a much more detailed analysis of settings space is required, we have used these values in the next experiments and left a more complete analysis to a next contribution.

$$\begin{cases} T(s,p,n): \\ \quad \text{Time to find the best answer for the} \\ \quad n^{th}\text{run of problem p with } s^{th}\text{ settings.} \\ AvgSP(s,p) = \frac{1}{10}\sum_{i=1}^{10} T(s,p,i) \\ AvgP(p) = \frac{1}{10000}\sum_{i=1}^{10}\sum_{s=1}^{1000} T(s,p,i) \\ Cost(s) = \frac{1}{5}\sum_{i=1}^{5} \frac{AvgSP(s,i)}{AvgP(i)} \end{cases} \quad (3)$$

## 6.2. Efficiency Comparison with General Benchmarks

Table 2 presents the optimization performance of the proposed algorithm with the specified settings. For each test, we have run the algorithm 10 times and the best and average utilization rate (ratio of shapes area to the surrounding rectangle) along with the best utilization from literature is specified.

As stated there, in 5 of 17 benchmark problems, the proposed approach has found better results from the literature results and in the other cases, it has gained in average 3.1% lower results with one exception in which it gained 13% lower than literature.

Table 1. Some samples of random tests. The first 4 column (gray shaded) represent Base, Effectiveness, Sticking Rate and Mutation Rate. The next 25 columns (not shaded) represent the iteration counts, 5 times for each of the 5 test problems with the specified settings. The next (gray shaded) 5 columns represent the average iteration counts for each of the 5 problems, and the last column represents the weighted sum of the average iteration counts, called the cost measure.

| | Base | Effectiveness | Stk. Rate | Mut. Rate | Prob.1 Run1 | Prob.1 Run2 | … | Prob.5 Run 5 | Avg. Prob. 1 | … | Avg. Prob. 5 | Cost Measure |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.87 | 0.12632 | 0.336 | 0.029 | 267 | 234 | | 1982 | 310 | | 2010 | 0.7954 |
| 2 | 0.93 | 0.13603 | 0.011 | 0.007 | 324 | 123 | | 1244 | 335 | | 1141 | 0.8012 |
| 3 | 0.86 | 0.1067 | 0.432 | 0.015 | 464 | 556 | | 2265 | 411 | | 3101 | 0.8096 |
| 4 | 0.76 | 0.007 | 0.05 | 0.136 | 474 | 411 | | 3101 | 402 | | 3311 | 0.8154 |
| 5 | 0.7 | 0.277 | 0.27 | 0.367 | 565 | 134 | | 2012 | 601 | | 3050 | 0.8201 |
| … | | | | | | | | | | | | |

Table 2. Comparison with literature results

| Sample Name | Number of Pieces | Best Utilization From Literature (%) | Best Utilization of Proposed Approach (%) | Difference (%) | Average time to compute the best result (minutes) |
|---|---|---|---|---|---|
| M1 | 16 | 97.8 [15] | 96.15 | - 1.65 | 31 |
| M2 | 17 | 93.3 [15] | 94.37 | + 1.07 | 23 |
| M3 | 20 | 94.6 [15] | 90.86 | - 4.60 | 29 |
| J1 | 25 | 85.81 [16] | 88.8 | + 2.99 | 44 |
| J2 | 50 | 86.80 [16] | 81.15 | - 5.65 | 245 |
| D1 | 31 | 95.67 [15] | 92.9 | - 2.77 | 101 |
| D2 | 21 | 100 [15] | 91.17 | - 5.83 | 53 |
| D3 | 37 | 97.46 [15] | 93.11 | - 4.35 | 227 |
| D4 | 37 | 99.51 [15] | 86.51 | - 13.00 | 213 |
| K | 13 | 96.59 [14] | 91.15 | - 5.44 | 23 |
| T1a | 17 | 94.6 [15] | 96.16 | + 1.56 | 44 |
| T2a | 25 | 94.8 [15] | 96.54 | + 1.74 | 34 |
| T3a | 29 | 94.1 [15] | 97.5 | + 3.4 | 43 |
| T4a | 49 | 95.3 [15] | 92.11 | - 3.19 | 254 |
| T5a | 73 | 95.0 [15] | 93.86 | - 1.14 | 315 |
| T6a | 97 | 97.1 [15] | 96.41 | - 0.69 | 441 |
| T7a | 199 | 98.0 [15] | 97.5 | - 0.5 | 646 |

## 6.3. Efficiency Comparison with Pure Genetic Algorithm

As almost all of the specified literature results are achieved using some heuristics combined with genetic algorithm, all tests are also done with a pure genetic algorithm program. To compute the best parameters for the GA, we designed a GA whose mutation rate, cross over rate and cross over's number of cut points are selected using a hill climbing approach. Each test with pure G.A. is also repeated for 10 times. As stated in Table 3, our method has gained better results in compare to G.A. benchmarks in 15 of 17 cases in average with 4.71% higher utilization, equal in one case and lower in one case with 3.17%.

## 6.4. Analysis of 'Partial Solution Generation' Property

As stated in section 5, the major advantage of this algorithm lies in its ability to generate and maintain partial solutions during its process, which results in self-organized more focus around combinations that have shown better results so far. The probability of a certain combination's existence in next trials is directly proportional to the number of its occurrences in current population. Thus, for each shape, we can compute the occurrence probability of its different combinations in the next trial and find the maximum of these values as a measure of *Dominance* of one combination over the rest. Once *Dominance* value is high, it means that a certain combination is tried more than other possible combinations, and the search is focused around a certain solution. And when it is low, it means that there is no special more important position and search is run globally. Equation 4 presents this measure:

$$\begin{cases} \text{Dominance}(S) = \dfrac{\underset{G \in \text{Possible Combinations of S with other shapes.}}{\text{Max}}(\text{Number of genomes that include } G)}{\text{Number of Genomes that Include S.}} \\ \text{General Dominance} = \dfrac{\sum_{\text{All Shapes}} \text{Dominance}(S)}{\text{Number of shapes}} \end{cases} \quad (4)$$

Figure 6 presents the general dominance measure for all shapes through time, along with relative fitness values of complete shapes. As it is depicted there, once a result, better than previous ones is found, the dominance value increases as the search process is directed towards the newly found result. If the new result's neighborhood keeps showing good answers, the measure keeps staying high and while the next results are not that good, the value gradually decreases and the system is relaxed to search more different cases.

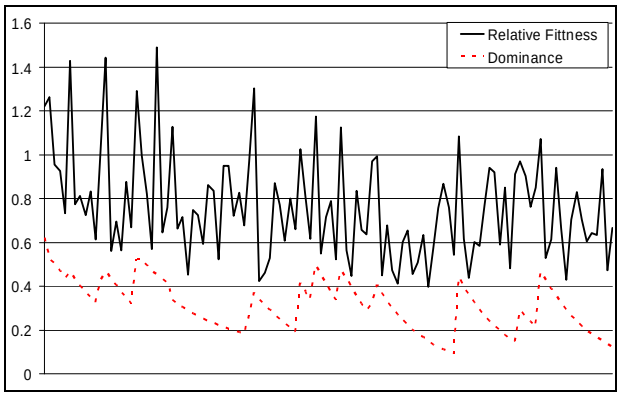


Figure 6. The Dominance Factor's relation with Relative Fitness. The horizontal axis represents time and the relative fitness values and dominance values are represented respectively with continuous and dotted lines.

## 6.5. 'Base' Parameter's Effect Analysis

As stated in section 5 we expect the *Base* parameter filter the solutions with lower fitness values. Expressing this using Eq.4's general dominance, the *Base* parameter filters the solutions that can increase their density and therefore, increase dominance. To test this, the dominance of two runs with totally similar parameter sets except for the *Base* are depicted in Figure 7. The first run uses 50% Base value and the other 90%. As it is seen the dominance of the one with lower base changes quite abruptly but the other one has a much more stable dominance. This suggests that the first one changes its attention to new solutions very often while the second one is much more selective in choosing new focus points.

Table 3. Comparison with pure genetic algorithm with samples from literature

| Sample Name | Number of Pieces | Best Utilization of Pure Genetic Algorithm (%) | Best Utilization of Proposed Approach (%) | Difference (%) |
|---|---|---|---|---|
| M1 | 16 | 91.24 | 96.15 | + 4.91 |
| M2 | 17 | 89.94 | 94.37 | + 4.43 |
| M3 | 20 | 86.24 | 90.86 | + 4.62 |
| J1 | 25 | 83.3 | 88.8 | + 5.50 |
| J2 | 50 | 81.15 | 81.15 | 0 |
| D1 | 31 | 85.22 | 92.9 | + 7.68 |
| D2 | 21 | 94.34 | 91.17 | - 3.17 |
| D3 | 37 | 82.11 | 93.11 | + 11.00 |
| D4 | 37 | 79.25 | 86.51 | + 7.26 |
| K | 13 | 86.55 | 91.15 | + 4.6 |
| T1a | 17 | 90.11 | 96.16 | + 2.05 |
| T2a | 25 | 95.51 | 96.54 | + 1.03 |
| T3a | 29 | 93.1 | 97.5 | + 4.4 |
| T4a | 49 | 86.58 | 92.11 | + 5.53 |
| T5a | 73 | 91.14 | 93.86 | + 2.72 |
| T6a | 97 | 93.5 | 96.41 | + 2.91 |
| T7a | 199 | 95.49 | 97.5 | + 2.01 |

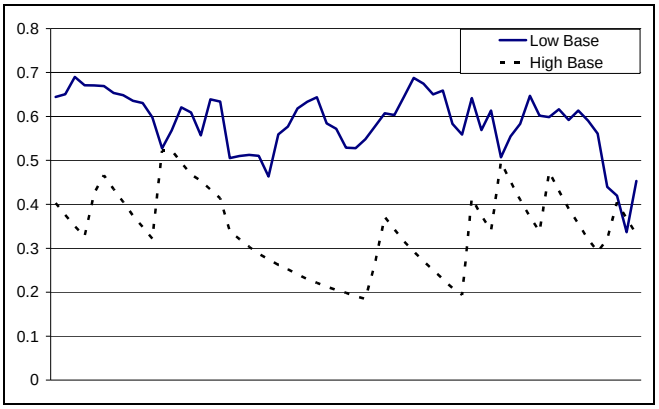


Figure 7. The effect of the *Base* parameter on search process. The diagram presents the dominance measures of two runs, one with 50% base value and the other with 90%. The vertical axis represents the Dominance and the horizontal axis represents time.

## 6.6. 'Effectiveness' Parameter's Effect Analysis

As stated before, the *Effectiveness* parameter represents the degree of focus on newly found good solutions. Representing this with the Dominance measure, we must expect higher *Effectiveness* increase the dominance more when a good solution is found in compare with lower *Effectiveness* values. To visualize this, Figure 8 presents the dominance of two runs, where one has 20% effectiveness and the other 50%.

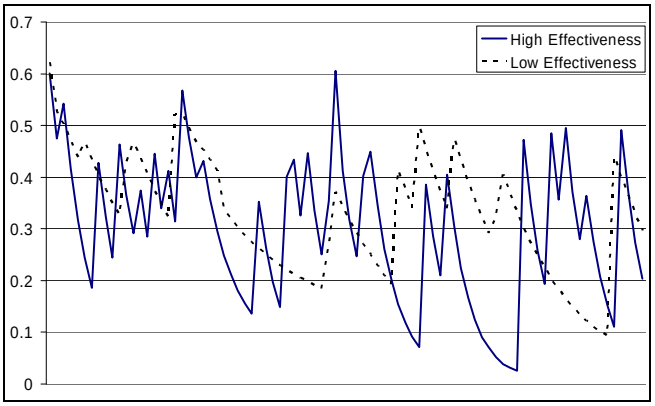


Figure 8. The effect of Effectiveness parameter on search process. The diagram presents the dominance measures of two runs, one with 20% effectiveness and the other with 50%. The vertical axis represents the Dominance and the horizontal axis represents time.

## 6.7. 'Mutation Rate' Parameter's Effect Analysis

Mutation rate cane also be analyzed from Dominance view point. Figure 9 shows that lower mutation rate results in much more focus on newly found good solutions and higher mutation rate results in more variety of tried cases. Here, the diagram depicts the dominance measure of two runs, one with 30% mutation rate and the other 10%.

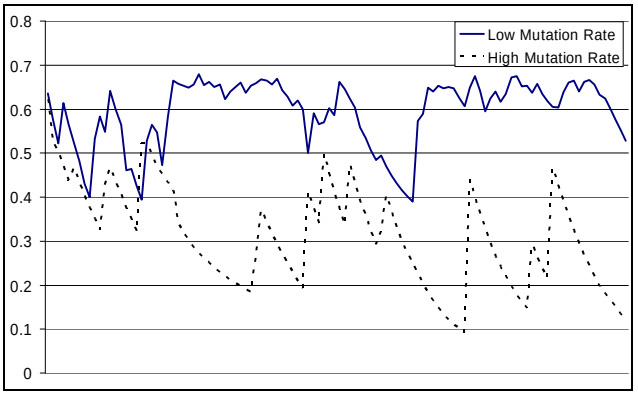


Figure 9. The effect of Mutation Rate on search process. The diagram presents the dominance measures of two runs, one with 10% mutation rate and the other with 30%. The vertical axis represents the Dominance and the horizontal axis represents time.

## 6.8. 'Sticking Rate' Parameter's Effect Analysis

As stated before, this parameter directly affects how fast we converge to big solutions and therefore, search smaller components' permutations less. Therefore, higher 'Sticking Rate' must result in faster solutions, but probably less optimized answers while there is slower convergence but higher hope of optimization with lower Sticking Rate. This is presented in Figure 10, where the best result of two runs is depicted through time. They have similar parameters except for S.R. which is 15% in one and 50% in the other. It is seen that the one with higher S.R. has found a good result quite soon, but is stuck in a local maxima while the other one has found similar result much later, but has continued to find better solutions.

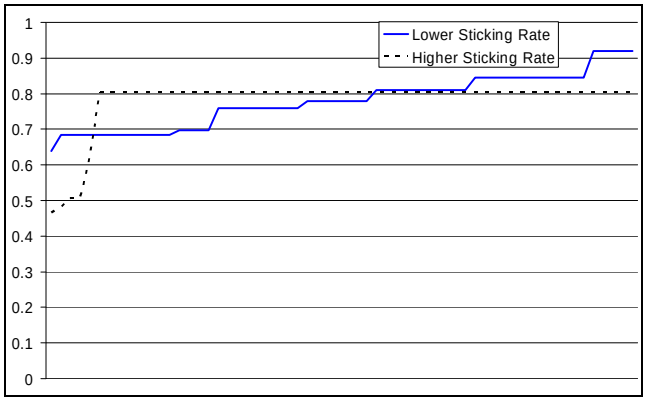


Figure 10. The effect of Sticking Rate on search process. The diagram presents the dominance measures of two runs, one with 15% mutation rate and the other with 50%. The vertical axis represents the Dominance and the horizontal axis represents time.

# 7. Concluding Remarks

Packing problems arise from a variety of situations including pallet loading, textile cutting, container stuffing and

placement problems. Such problems are optimization problems that are concerned with finding a good arrangement of multiple objects (2-D or 3-D) in a larger containing region without overlap. A novel evolutionary approach was presented which is based on 3 new evolutionary operators for optimizing the arrangement of irregular shapes. In this approach, each chromosome represents a partial solution and the operators take care of evolution of more fitting sub-solutions till an acceptable complete solution emerges.

The algorithm was compared with benchmarks from literature. Although all cited results use some heuristics to guide the search approach and the proposed method uses no predefined heuristic, its results are quite comparable with literature results and in some cases it has reached better results. Also, to make a fairer test, the approach was compared with pure genetic algorithm and has gained much better results in almost all cases. From these results, it can be concluded that the approach has been able to produce and exploit some rules of thumb during its search process as it has gained much better results in compare with a pure search method (GA) and similar results when GA was equipped some heuristics.

The proposed algorithm has four parameters which were briefly analyzed in this paper, but they must be analyzed in more detail in a later contribution. The tests were performed using parameter sets produced through generation of some random settings and choosing the one which has the best result over test cases, but more analysis on algorithm's sensitivity to these parameters can also be a next step to make this approach a general optimization approach. Also, we used a simple random selection for the *Selection* operator and implementing more complex methods can be tested in future works.

# References

[1] Adamowicz M. and Albano A. Nesting two dimensional shapes in rectangular modules. Computer Aided Design, pp 27–33, 1976.

[2] Albano A. and Sapuppo G. Optimal allocation of two dimensional irregular shapes using heuristic search method. IEEE Transactions on Systems, Man, and Cybernetics, pp 242–248, 1980.

[3] Blazewicz H. P., Walkowiak J. Using a tabu search approach for solving the two-dimensional irregular cutting problem. Annals of Operations Research, pp 313–327, 1993.

[4] Burke, E., Hellier R.S.R., Kendall G., and Whitwell G., Irregular Packing Using the Line and Arc No-Fit Polygon, Operations Research, vol.58, no.4 part 1, pp. 948-970, 2010.

[5] Burke E. and Kendall G., Applying Simulated Annealing and the No Fit Polygon to the Nesting Problem. Proceedings of the World Manufacturing Congress, Durham, UK, pp. 27-30, 1999.

[6] Chen P., Fu Z., Lim A., and Rodrigues B., Two dimensional Packing for Irregular Shaped Objects, In proceedings of the thirty-sixth Hawaii International Conference on System Sciences (HICSS-36) 2003, Big Island, Hawaii, USA.

[7] Dagli C. H. and Poshyanonda P., New approaches to nesting rectangular patterns. Journal of Intelligent Manufacturing 8, 177-190, 1997.

[8] Dighe R. and Jakiela M. J., Solving Pattern Nesting Problems with Genetic Algorithms Employing Task Decomposition and Contact Detection. Evolutionary Computation 3, 239-266, 1996.

[9] Dori D. and Ben-Bassat M.. Efficient nesting of congruent convex figures. Communications of the ACM, pp 228– 235, 1984.

[10] Dowsland K. and Dowsland W. Heuristic approaches to irregular cutting probelms. Working Paper EBMS, 1993.

[11] Egeblad J., Nielsen B.K., and Odgaard A., Fast neighborhood search for two-and three-dimensional nesting problems, European Journal of Operational Research, vol.183, no.3, pp. 1249-1266, 2007.

[12] Freeman H. and Shapira R. Determining the minimum area encasing rectangle for an arbitrary closed curve. Communications of the ACM, pp 409–413, 1975.

[13] Fujiyoshi K, Murata H, Arbitrary convex and concave rectilinear block packing using sequence-pair, Proceedings of the 1999 international symposium on Physical design.

[14] Goldberg. D.E. Genetic algorithms in search, optimization and machine learning, Addison-Wesley, 1989.

[15] Hopper E, Two-dimensional Packing utilizing Evolutionary Algorithms and other Meta-Heuristic Methods, PHD Thesis, University of Wales, Cardiff, 2000.

[16] Jakobs S., On genetic algorithms for the packing of Polygons, European Journal of Operations Research 88, 165-181, 1996.

[17] Keishi S. N., Yoji S. and K. The multi-bsg: stochastic approach to an optimal packing of convex-rectilinear blocks. ICCAD98, pp 267–274, January 1998.

[18] Lesh N., Marks J. , McMahon A., Mitzenmacher M. , New Heuristic and Interactive Approaches to 2D Rectangular Strip Packing, Proceedings of 18th International Joint Conference on Artificial Intelligence, August 2003.

[19] Liu J.F., Li G., and Geng H.T., A new heuristic algorithm for the circular packing problem with equilibrium constraints, Science China Information Sciences, vol.54, no.8, pp.1572-1584, 2011.

[20] Maggie Z. and Wayne W. Arbitrary rectilinear packing based on sequence pair. ICCAD98, pp 257–266, 1998.

[21] Pasha A., Geometric bin packing algorithm for arbitrary shapes, MS-Thesis, Center for Intelligent Machines and Robotics, Department of Mechanical and Aerospace Engineering,University of Florida, 2003.

[22] Qu W. and Sandersc J. A nesting algorithm for irregular parts and factors affecting trim losses. International Journal

of Production Research, pp 381–397, 1987.

[23] Ratanapan K. and Dagli C. H., An object-based evolutionary algorithm for solving irregular nesting problems. In: Proceedings for Artificial Neural Networks in Engineering Conference (ANNIE '97), vol. 7, ASME Press, New York, pp. 383-388, 1997.

[24] Ratanapan K. and Dagli C. H., An object-based evolutionary algorithm: the nesting solution. In: IEEE (eds.) Proceedings of the International Conference on Evolutionary Computation 1998, ICEC '98, IEEE, Piscataway, NJ, USA, pp. 581-586, 1998.

[25] Sakait J. and Hae C. Two-dimensional packing problems using genetic algorithms. Engineering with Computers, pp 206–213, 1998.

[26] Selow R., Neves F., and Lopes H.S., Genetic Algorithms and Heuristic Rules for Solving the Nesting Problem in the Package Industry, Trends in Intelligent Systems and Computer Engineering, pp.189-207, 2008.

[27] Umetani S., Yagiura M., Imahori S., Imamichi T., Nonobe K., and Ibaraki T., Solving the irregular strip packing problem via guided local search for overlap minimization, International Transactions in Operational Research, vol.16, no.6, pp.661-683, 2009.

**Ramin Halavati** received his PhD in 2010 and MSc in 2003, both in Computer Engineering and from Sharif University of Technology, and received BSc of Software engineering in 2000 from Iran University of Science and Technology. He is now a researcher at Electrical Engineering Department of Sharif University of Technology and his main research interests are Pattern Recognition, Image Processing, and Machine Learning.
**E-mail:** halavati@ee.sharif.edu

**Saeed Bagheri Shouraki** received his PhD in Fuzzy Systems Control from UEC Japan in 2000, MSc of digital Electronics from Sharif University of Technology in 1986, and Bs. of electronic in 1984. He is now a faculty member of Electrical Engineering Department of Sharif University of Technology and his main research interests are Robotics, Fuzzy Systems, Soft Computing, Fuzzy Controllers, Data Acquisition Systems, Adaptive Signal Processing, and Networks
**E-mail:** bagheri_s@sharif.edu

**Saman Haratizadeh** received his PhD in Computer Engineering from Sharif university of Technology in 2008, where he has also received his M.Sc and B.Sc in 2002 and 2000. He is currently an AI researcher lecturing in the University of Tehran and Alzahra University. His main research interest is bio-inspired computing and he is currently working in artificial life systems, artificial emotions and reinforcement learning domains.
**E-mail:** harati@gmail.com

**Paper Handling Data:**

Submitted: 10.02.2006
Received in revised form: 07.25.2011
Accepted: 08.12.2011
Corresponding author: Dr. Ramin Halavati,
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.