01 Jan 1997

An Object-Based Evolutionary Algorithm for Solving Rectangular Piece Nesting Problems

Kanchitpol Ratanapan

Cihan H. Dagli
Missouri University of Science and Technology, dagli@mst.edu

Recommended Citation

K. Ratanapan and C. H. Dagli, "An Object-Based Evolutionary Algorithm for Solving Rectangular Piece Nesting Problems," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 1997*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1997.
The definitive version is available at https://doi.org/10.1109/ICSMC.1997.638076

An Object-Based Evolutionary Algorithm for Solving Rectangular Piece Nesting Problems

Kanchitpol Ratanapan and Cihan H. Dagli
Smart Engineering Systems Lab.
Engineering Management Department
University of Missouri-Rolla

ABSTRACT

Nesting problems have been tackled by researchers using a vast number of algorithms in the past. Most of the algorithms, however, need to perform on a one-dimensional space. Therefore, the problem must be transformed into a one-dimensional space problem similar to the traveling salesman problem. Consequently, loss of solutions due to the dimensional reduction may occur. In this study, an object-based evolutionary algorithm for rectangular piece nesting problems is proposed. This methodology is created on truly two-dimensional space, allowing new mechanisms (i.e., individual representation, initialization, etc.) and new object-based genetic operators (i.e., hill-climbing, mutation, and recombination operators) to perform effectively on the space. Since, no dimensional reduction is used: therefore, no solution losses during the searching. Simulation/Animation of the layouts shows the continual improvement by using this method over generations. Experimental results are promising.

1. INTRODUCTION

The cutting and packing problems represents an important issue in business and in research. Thousands of articles related to the problems were published since the 1960s. Many solution methods have been proposed but none of these methods has offered a perfect solution to the problems due to their NP-complete nature [1],[2]. Recent research reports that two million dollars could be saved per year for a textile manufacturing if the average improvement in efficiency of a layout problem is increased by only 0.1% [2]. Comprehensive reviews of the problems can be found in [3],[4],[5].

Nesting problems present the most difficult issues on two-dimensional cutting and packing problem. The problem is known by different names for different industries, e.g, the marker problem for textile industry, the part-nesting problem for ship-building industry, and the floor planning problem in VLSI chip-building industry.

The objective of this research is to develop a methodology that can solve the nesting problems directly on the two-dimensional space so that all possible solutions could be found. This new methodology is called an Object-Based Evolutionary Algorithm (OBEA). It has been created based on a convergence of three ideas, namely evolutionary algorithm, graphical data manipulation, and simulation/animation. This new methodology has been tested against some rectangular versions of nesting problem.

2. EVOLUTIONARY ALGORITHMS

Thirty years ago, three different algorithms based on the evolution theory of Charles Darwin--namely Evolutionary Strategies (ES), Evolutionary Programming (EP), and Genetic Algorithm (GA)--were independently developed for solving complex problems, e.g., machine learning and optimization problems. The idea of the principle of natural selection and genetics operations is used in these three algorithms. Population, genetic operators, and fitness-based selection are the major components of each algorithm; however, the way those components are applied differs slightly to each algorithms.

Surprisingly, none of these algorithm were widely accepted until about ten years ago. Thousands of articles concerned with theoretical, empirical, and application, were published. However, only two books that brought the three algorithms together have been published, the first book by D. B. Fogel [6] and the other by Bäck [7]. In Bäck's book, a general algorithmic outline of an evolutionary algorithm (EA) is given as follows.

$t = 0$;
Initialization $P(0) = \{\bar{a}_1(0),...,\bar{a}_\mu(0)\} \in I^\mu$;
Evaluation $P(0) : \{\Phi(\bar{a}_1(0)),...,\Phi(\bar{a}_\mu(0))\}$;
While ($\iota(P(t)) \neq$ true) do
  Recombination $P'(t) = r\Theta_r(P(t))$;
  Mutation $P''(t) = m\Theta_m(P'(t))$;
  Evaluation $P''(t) : \{\Phi(\bar{a}''_1(t)),...,\Phi(\bar{a}''_\lambda(t))\}$;
  Selection $P(t+1) = s\Theta_s(P''(t) \cup Q)$;
  $t = t+1$;
Od ;

This guideline can be simplified as follows. The algorithm starts with setting a generation number, $t = 0$. Then, an initial population, $P(0)$, is created by a set of individuals, $\bar{a}$, for μ individuals and considered as the zero generation. Each individual in the zero generation is evaluated by an evaluation function, Φ, to find its fitness value. After that, the next generation will be created iteratively by performing some genetic operations until the termination criteria, ι, are met. This next generation is produced by performing a recombination operator, $r\Theta_r$, and a mutation operator, $m\Theta_m$, on the current population. The new individuals of new size λ will be calculated. Finally, a selection process, $s\Theta_s$, will select and move some individuals of size μ to create a new population for the next generation where $t = t + 1$.

The structure of EA is computationally simple but powerful. Lately, one of the most popular problems for evaluation new algorithms--the Traveling Salesperson Problem (TSP)--is also tackled using these algorithms. Experimental results presented by various authors look very promising [8],[9],[10]. In fact, this TSP approach has been used for solving the nesting problem for more than twenty years. When the Evolutionary Algorithms showed the promising results on the TSP, many researchers adopted and applied these algorithms to tackle the nesting problems as well. Again, most results received by using these new TSP methods are also good.

0-7803-4053-1/97/$10.00 © 1997 IEEE

To solve the nesting problem using the TSP method, a two-step approach is obtained. The first step is to form a sequence of pieces. The characteristic of this piece sequencing problem and the TSPs are compatible. A technique that can solve the TSP is also good for solving the sequencing problem as well. The second step is to place them on a region next to each other and then calculating the packing density based on a given placement policy, e.g., from left to right, from bottom to top, or from lower-left to upper-right.

However, solving the two-dimensional problem by transforming it into a TSP which is a one-dimensional problem, and then using an EA to find a good sequence might not be appropriate due to the dimension reduction of the problem. A reasonable doubt in using the idea of the TSP for solving the nesting problem exists. The search space of the TSP is only one-dimensional. It might not cover the whole solution of the nesting problem. Unfortunately, no one has ever proven this issue.

For the doubt of the space loss to be eliminated, a two-dimensional search space technique needs to be created to guarantee that all solutions can be discovered. One possible solution is to use the EA directly on the two-dimensional space as the search approach. However, the traditional EA can not be used in this matter because the solution for the nesting problem is not a set of points on the two-dimensional space; rather, it is a set of pieces or object on that space. Therefore, a new version of the algorithm that works with objects needs to be created. All new representations, mechanisms, and genetic operators must be created in such a way that this objected version of the EA can be performed. Even more, all possible solutions should be found. This new algorithm is called an Object-Based Evolutionary Algorithm (OBEA).

3. OBJECT-BASED EVOLUTIONARY ALGORITHMS

The outline of the OBEA has only a minor changes from the traditional EA. This new EA employs all the main steps of the EA from initialization to termination criteria. However, an additional operator named hill-climbing is added for helping in movement of all pieces. Following is the general outline of OBEA.

$t = 0$;
Initialization $P(0) = \{\bar{a}_1(0),...,\bar{a}_\mu(0)\} \in I^\mu$;
Evaluation $P(0) : \{\Phi(\bar{a}_1(0)),...,\Phi(\bar{a}_\mu(0))\}$;
While ($\iota(P(t)) \neq$ true) do
*** Hill-climbing $P'(t) = h\Theta_h(P(t))$;
Recombination $P''(t) = r\Theta_r(P'(t))$;
Mutation $P'''(t) = m\Theta_m(P''(t))$;
Evaluation $P'''(t) : \{\Phi(\bar{a}'''_1(t)),...,\Phi(\bar{a}'''_\lambda(t))\}$;
Selection $P(t+1) = s\Theta_s(P'''(t) \cup Q)$;
$t = t+1$;
Od ;

A. Mechanisms

All mechanisms--namely, individual representations, fitness evaluation function, and termination criteria--are created and performed in totally different ways by those traditional EA. Following are details of these mechanisms used in the OBEA.

Individual Representations: A solution of the problems, such as nesting problem, needs more than a set of points in the two-dimensional space. It requires a set of small regions called pieces onto a larger region called plane. Furthermore, those pieces should neither be overlapped from each other nor the plane. In other words, overlapping and over boarding of the pieces and the plane can not be acceptable. Obviously, a solution of the nesting problem is too complicated to represent as a set of two-dimensional vertices. Each piece needs to be treated as a solid object. Each piece must have it own space in the plane. Also, the plane itself must be a solid object where pieces can be placed on top of the plane. These pieces and the plane is formed a solution of the problem or a layout of the problem. The plane and the pieces can be any shape and any orientation. Figure 1 illustrates an individual representation of the OBEA on an infinite two-dimensional plane.

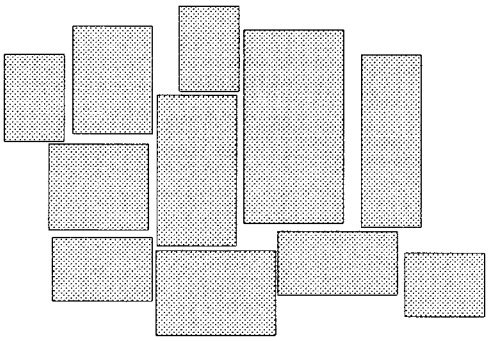

Figure 1 An individual representation of OBEA

Initialization Process: Information used in this process includes the size of the plane, the size of the pieces, the number of pieces that need to be placed on the plane, and etc. The initialization process uses that information to create a solid plane as the first step. Then, each piece is placed on the plane in random order and location. It is possible that a piece cannot be placed on the first tried. Several attempts may help to find a position for a piece to be place. Re-initialization will be done if a piece cannot be placed after some exhausted attempts. Usually, more than one layout will be created during initialization. This set of the layouts is called an initial population or the zero generation of layouts as shown in figure 2.

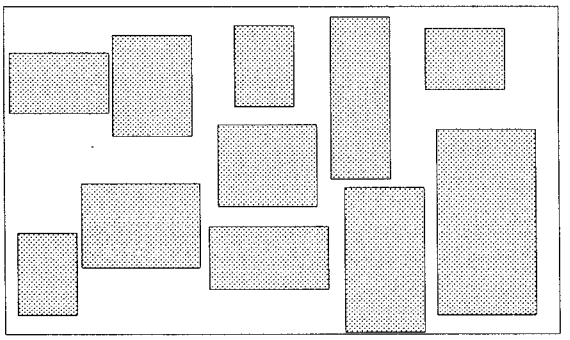

Figure 2 A solution of an initial population

Fitness Evaluation Function: The fitness value can be calculated directly from the plane depending on the objectives of the problem. If the objective of the problem is to minimize the space used for rectangular pieces on a rectangular steel sheet, the fitness value might be the maximum value of the last piece on the sheet. Figure 3 illustrates the maximum value of the last part as the fitness value. This fitness evaluation function is performed after the

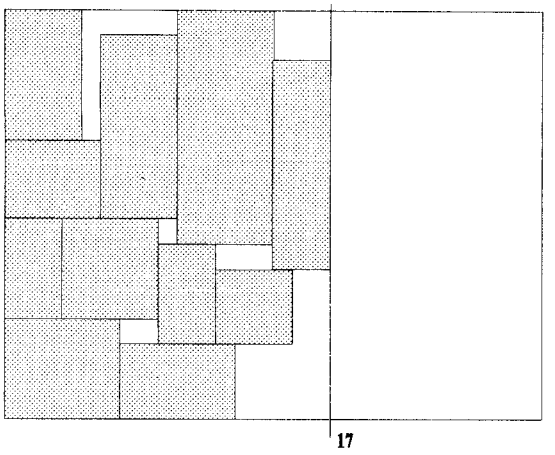

Figure 3 The maximum value of all pieces is the fitness value

initialization is done and after any individual has been changed in any generation.

Termination Criteria: Usually, the algorithm is terminated by specifying a maximum number of generations. However, some additional termination criteria may be added, such as a maximum number of generations in which the best value remains the same. Other termination criteria can be introduced upon requirement or problem objective.

B. Genetic Operators

For an algorithm to be called the Evolutionary Algorithm, some genetic operators need to be presented. The idea behind the operators is to change the location or orientation of the pieces so that different layouts are created. Some operators will make a big change; and some will make only slight change. Three groups of operators can be categorized based on their actions and results after they are performed on the layouts. All basic elements of each operation are obtained from the two-dimensional transformations of graphical data manipulation. These three groups of operators, namely hill-climbing, mutation, and recombination, are described as follows.

Hill-climbing operators: In two-dimensional space, the layout changing can be easily made by moving a piece. If the change is performed on a single layout and its fitness value has not gotten worse, this situation is called hill-climbing. For the Object-Based Evolutionary Algorithm, some genetic operators categorized as hill-climbing operators are "translation", "rotation", and "relocation".

a. Translation operators. This operator is performed on a piece on a plane by moving the piece one unit at a time toward the gravitational force. The center of the force can come from a single point or multiple points of the two-dimensional plane. When this force is applied, it makes all pieces move at the same time. The mathematical representation of the force obtained from the graphical data manipulation is called a unit translation. The unit translation equation can be represented as matrix multiplications shown as follows

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_X & T_Y & 1 \end{bmatrix}$$

where x and y is a position of the lower-left corner of a rectangle; x' and y' is a new position of the rectangle; and T_x and T_y = -1, 0, or 1 [11]. For example, if the lower-left corner of the plane is the center of the force, all piece will be moved toward that Corner one unit for each piece in one generation. T_x and T_y in this case is equal to -1. However, some operations will not perform, if next move is created the overlapping. After the move operations are performed for a number of generations, all pieces will be moved and packed at the lower-left corner of the plane.

b. Rectangular Rotation operator. In the rectangular version of the nesting problem, each of the four corners can be used as a rotation reference point. However, the word "rotation" may not be fully appropriate because two steps are always required. The first step is to rotate a rectangle using a pre-determined rotation reference point ninety degrees different from the original in a virtual space. The second step is to move the piece back to a position of the reference point. For example, the selected pivot point is lower-right Corner of a piece. The rotation operator will rotate the piece for minus 90 degrees and then move the new lower-right corner back to the previous lower-right corner (See figure 4).

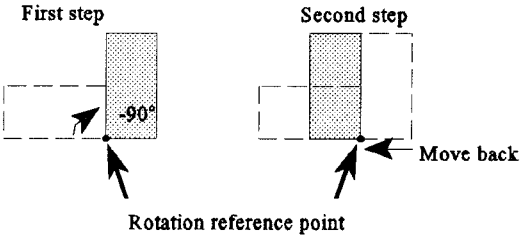

Figure 4 The two step rectangle rotation

In mathematic terms, this rectangular rotation is a composite transformation of a rotation and translation which can be written in term of matrix multiplication as follows.

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} \cos 90 & \sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_X & T_Y & 1 \end{bmatrix}$$

c. Relocation operators. The operators are done on a piece on a layout by relocating to a vacant position within the maximum value of the last piece. The piece should not be overlapped with the other, or over boarded, or over the maximum value line as shown in figure 5.

Mutation: If an operator is performed on a layout and the fitness value could possibly get worse, this operator is called a mutation. Mutation will create an offspring that look more different than the hill-climbing. There are many operators considered mutation one. Following are examples of the mutation operators.

a. Relocation-away operator. The idea of this operator is to relocate a piece to another place out of the maximum value so that all pieces will have some room to be reorganized. If the operation performs without any condition, the target position in a plane might not be empty because of a dense of pieces. One

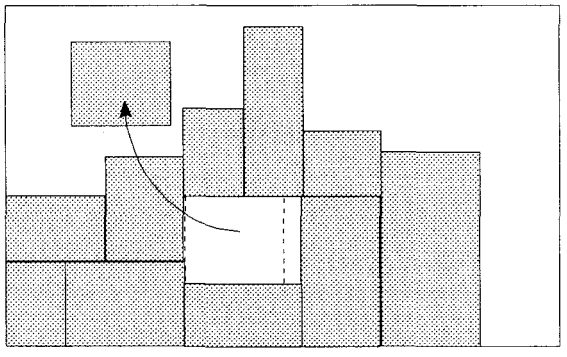

Figure 5 Jump a piece to a new position with in maximum value.

possible solution is to jump a piece in the opposite direction of the gravitational force where there is a guarantee of vacant space. For example, if the force is pulled to the lower-left corner of the plane, the piece should be jumped to upper-right corner. Figure 6 illustrates a jump-away result.

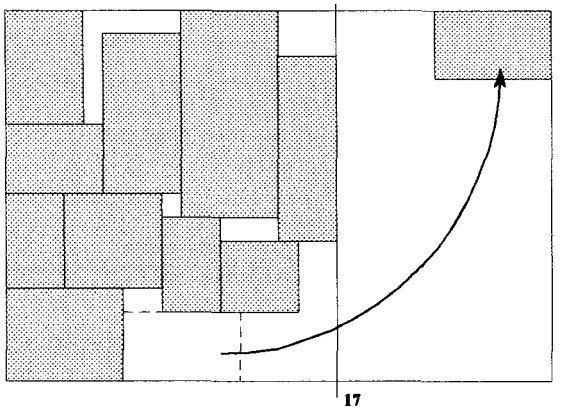

Figure 6 A piece is relocated by jump-away operator.

b. Point mutation. A relocated piece might create a better solution for the plane. However, if some pieces exist in the new location, the overlapping technique must be performed so that the space is available for relocation. Every piece, which starts from the overlapping position through the right-most piece, needs to be moved to a position where no overlapping occur. Each piece will be moved toward the opposite direction of gravitational force, or moved toward the right end of the plane. Figures 7 and 8 depict the mutation steps.

c. Area mutation. Similar to the point mutation, pieces that starts from an overlapping position need to be moved. The difference is that the whole specific area needs to be indicated. The specific area can be any shape or any size. All pieces overlapped with the mutation area will be packed together and relocated to a new location. The overlapping on the new location is possible. The overlapping technique used in point mutation will be performed to create a new layout. Figure 9 shows that four pieces are overlapping with the selected mutation area. Therefore, the four pieces will be packed together and moved to a new location.

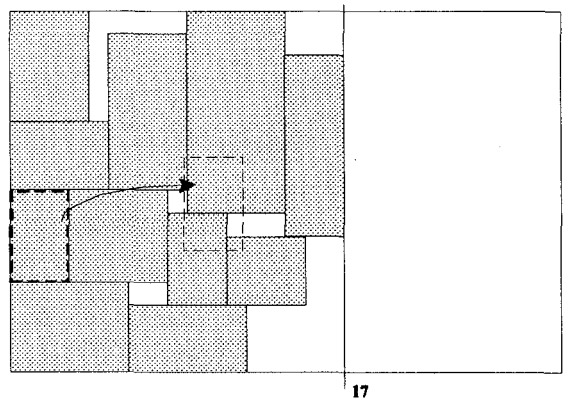

Figure 7 A piece is relocated to a new position created an overlapping.

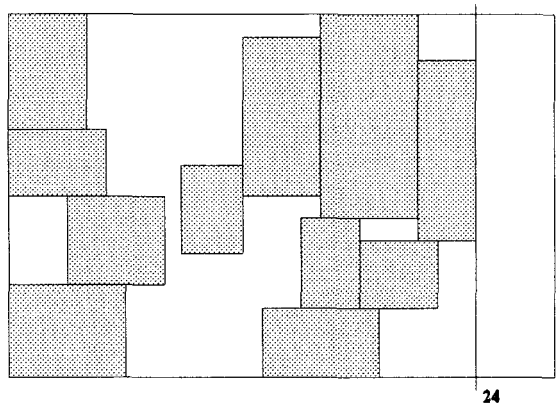

Figure 8 Overlapping result

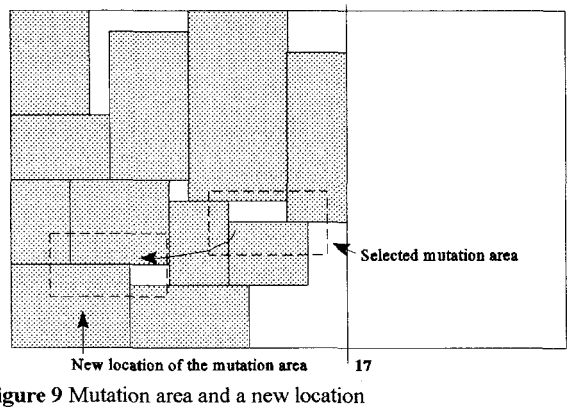

Figure 9 Mutation area and a new location

Recombination: This operator will be performed on two or more layouts by changing information to each other to create.

a. Point crossover. There are many ways to create point crossover. Suppose that two layouts are to crossover. A position on the first layout is selected. Then a piece on that position is determined. At the same time, another piece on the same position on the second plane is also determined. The piece from each layout will switch to another layout which create a duplicated piece on each

layout. If redundant pieces are not allowed, a replacing technique may be used by eliminating the existing piece. If an overlapping area exists, the overlapping technique as in point mutation will be used to solve the problem.

b. Area crossover. The crossover combines the idea of area mutation and the point crossover to create more possible recombinations of layouts. The area also can be any shape. All pieces in the crossover area will be switched to another layout. It is possible to use more than two layouts to do crossover. All redundant pieces will be eliminated, and the overlapping problem will be solved.

4. EXPERIMENTAL RESULTS

Simulation/animation is done on 2 different sets of pieces. Table 1 shows the first set of pieces which consists of 31 pieces with 6 different patterns. Table 2 shows the second set of pieces which consists of 21 pieces with 4 different patterns.

Pattern no.	Width (x)	Height (y)	Piece no.
1	12	10	1, 2, 3, 4, 5
2	10	11	6, 7, 8, 9, 10 ,11 ,12
3	9	13	13, 14, 15, 16
4	4	5	17, 18, 19, 20
5	9	10	21, 22, 23, 24, 25, 26
6	6	8	27, 28, 29, 30, 31

Table 1 The first set of testing pieces.

Pattern no.	Width (x)	Height (y)	Piece no.
1	12	12	5, 10, 14, 18
2	12	10	1, 4, 8, 9, 21
3	12	9	2, 3, 13, 17, 19, 20
4	12	8	6, 7, 11, 12, 15, 16

Table 2 The second set of testing pieces.

The first experiment is done using the first set by randomly placing all thirty-one pieces onto a stock sheet in order to create an initial layout. The initial layout has the packing density only 42.40 % (Figure 10).

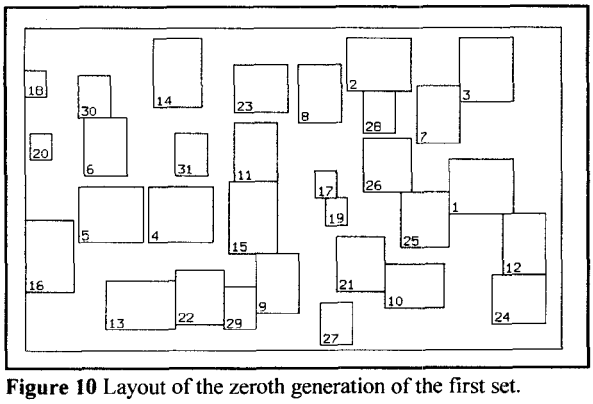

Figure 10 Layout of the zeroth generation of the first set.

The second generation is then created by performing the hill-climbing operators on the layout. This procedure is repeated until some termination criteria are met. In this case, it is repeated until the 1000th generation is reached. Figure 11, 12, and 13 show the improvement of the packing density of higher generations. The packing density increases from 42.40% in the zeroth generation (Figure 10), to 67.30 % in the 300th generation (Figure 11), to 80.00% in the 600th generation (Figure 12), and to 91.88 % in the 900th generation (Figure 13). Although the packing density of the final generation (Figure 14) remains the same as that of the 900th generation, the location of the pieces of the final generation is different from the location of the pieces of the 900th generation. This means that the evolution mechanism is still working. Therefore, if the preset termination condition is higher than 1000, it is possible to get a layout with a higher packing density.

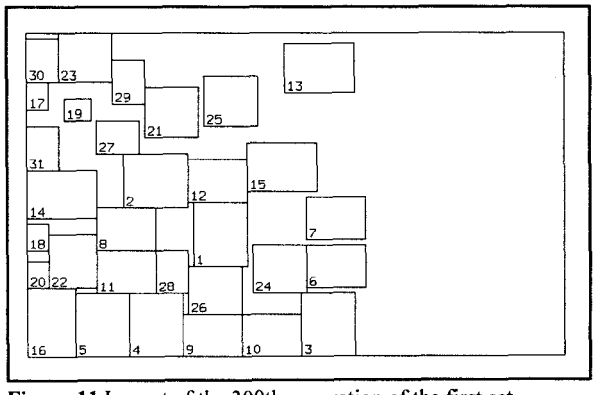

Figure 11 Layout of the 300th generation of the first set (packing density = 67.30 %).

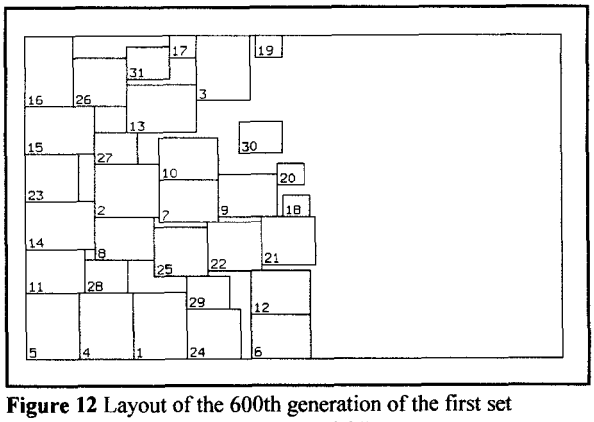

Figure 12 Layout of the 600th generation of the first set (packing density = 80.00 %).

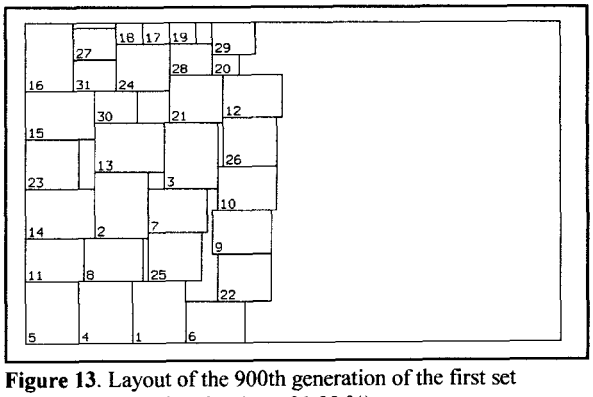

Figure 13. Layout of the 900th generation of the first set (packing density = 91.88 %).

The second set of pieces is created from a perfect rectangular which is cut into smaller pieces. Using the same algorithm, the initial layout

is created. After running for 1000 generations, the final layout is obtained (Figure 16). Surprisingly, the final layout is not the perfect layout, even though it contains no space among the pieces. Although the layout does not have a 100% packing density, it may still be considered to be one of the best solutions because layout has no internal scrape.

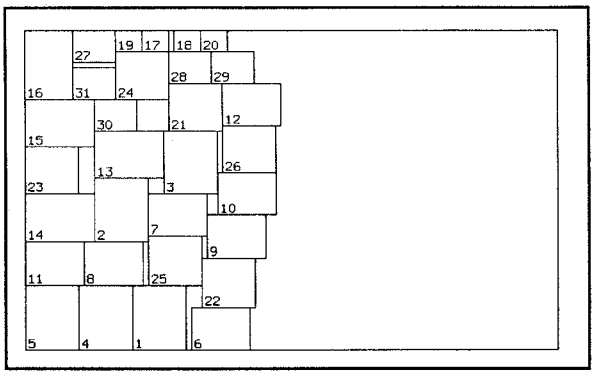

Figure 14 Layout of the 1000th generation of the first set

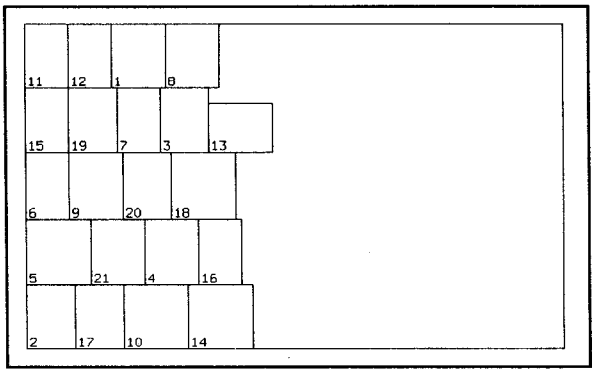

Figure 15. Layout of the 1000th generation of the second set.

Figure 16 and 17 show the packing density of the zeroth to the 1000th generation of both test examples respectively.

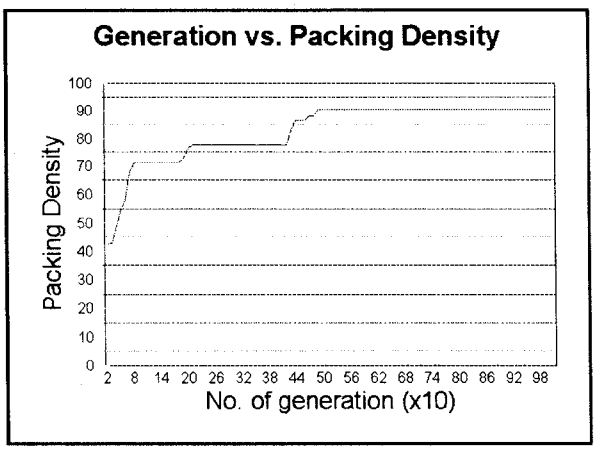

Figure 16 The packing density over generation of the first set.

5. CONCLUSION AND FUTURE WORKS

The idea of a multi-dimensional representation in EA is introduced in this paper. The Object-Based Evolutionary Algorithm proposed shows a new way of problem representation. All new mechanisms and genetic operators are developed. The advantage of this approach is the ability to exploit the whole solution space. Population concept and interaction between individuals is added to the algorithm to improve overall performance. The experimental results on rectangular version look very promising. More experiments on different problems such as irregular piece nesting problem is interesting to perform. Some restriction may be eliminated from idea of general purpose algorithm. New genetic operators may be introduced to maximize the algorithm to solve more general problem. Parallel implementation of this new algorithm may reduce in production times and lead to possible money saving.

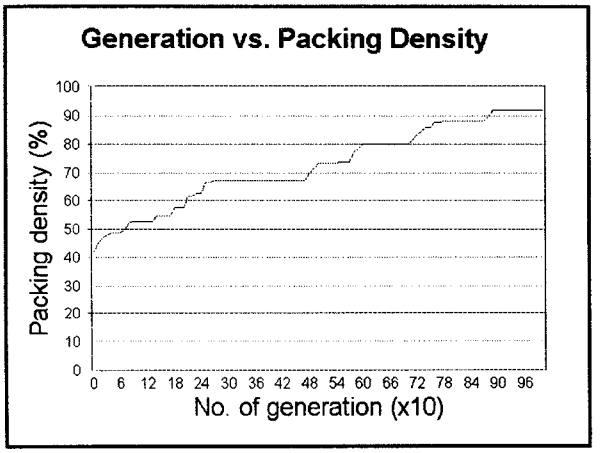

Figure 17 The packing density over generation of the second set.

6. REFERENCES

[1] M. R.Garey and D. S.Johnson, *Computer and Intractability: a Guild to the Theory of NP-Complete*, W.H. Freeman, 1979.

[2] Zhenyu Li and Victor Milenkovic,"Compaction and Separation Algorithms for Non-convex Polygons and their Applications", *European Journal of Operation Research*, V. 84, 1995, p. 539-561

[3] Kathryn A. Dowsland and William B. Dowsland, "Solution Approaches to Irregular Nesting Problems", *European Journal of Operation Research*, Vol. 84, 1995, pp. 506-521.

[4] H. Dycknoff and U. Finke, *Cutting and Packing in Production and Distribution: A typology and Bibliography*, Physica-Verlag: Heidelberg, Germany, 1992.

[5] C. H. Cheng, B. R. Feiring and T. C. E. Cheng, "The Cutting Stock Problem--A survey," *IJPE*, Vol. 36, No. 3 1994, 291-305.

[6] David B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, NewYork, 1995.

[7] Thomas Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press: New York, 1996.

[8] G. Rudolph, "Global optimization by means of distributed evolution strategies," *Parallel Problem Solving from Nature--Proceedings 1st Workshop PPSN I*, Vol. 496 of Lecture Notes in Computer Science, Springer: Berlin, Germany, 1991, p. 209-213.

[9] David B. Fogel, "Applying Evolutionary Programming to Selected Traveling Salesman Problems," *An International Journal of Cybernetics and System*, Vol. 24, 1993, 27-36.

[10] Gilbert Syswerda, "Schedule Optimization Using Genetic Algorithms," *Handbook of Genetic Algorithms*, Van Nostrand Reinhold: New York, 1991, p.332-349.

[11] Donald Hearn and M. Pauline Baker, *Computer Graphics*, Prentice-Hall International Editions, 1986.