01 Jan 1998

An Object-Based Evolutionary Algorithm: The Nesting Solution

Kanchitpol Ratanapan

Cihan H. Dagli

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

An Object-Based Evolutionary Algorithm: The Nesting Solution

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

ABSTRACT

The nesting problems have received considerable attention and have been addressed by a variety of algorithms. Recently, evolutionary algorithms have been adopted for solutions. Most of these algorithms, however, require a search in one-dimensional space; thus a transformation of the problem to a single dimension, as in the sequencing problems, is needed. Unfortunately, this restricts the search space. In this study, an object-based evolutionary algorithm for the nesting problems is proposed. The methodology is created in a true two-dimensional space, allowing object-based mechanisms and object-based evolutionary operators to perform effectively on the space without restricting search alternatives. Implementation of the algorithm is conducted using grid representation where no overlapping is allowed. Layout simulation/animation over generations shows the continual improvement by this method. Experimental results of packing density on rectangular and irregular versions of the nesting problem are up to 94.41% and 82.34%, respectively. For industrial-size data, five hundred and forty-three pieces are tested. The final packing density is 74.89%.

1. INTRODUCTION

The cutting and packing problems are of both academic and practical interest. Most industries have been dealing with these problems using a variety of techniques for decades. Hundreds of articles related to the problems have been published since the 1960s. A vast number of approximate solutions to the problems have been proposed in the literature, due to their NP-complete characteristics [1,2].

Nesting problems pose some of the most difficult issues on two-dimensional cutting and packing problems because they involve nesting of either regular or irregular pieces on regions. The main objective of the nesting problem is similar to other cutting and packing problems in that scrap needs to be minimized. For example, if a set of specific sizes of rectangular pieces is needed to be cut from a continuous sheet with a constant width, the patterns of these pieces will be nested in such a way that the length of the sheet is minimized.

The nesting problem is known by different names for different industries, e.g., a marker problem for the textile industry, a part-nesting problem for the shipbuilding industry, and a floor planning problem in the VLSI chip-building industry. However, the word "layout problem" is generally understood in most industries. 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].

Recently, many researchers approach these problems by reducing the complexity of the problems from two to one dimensional space problem, making it similar to a sequencing problem. They then solve the latter problem instead. No report is found that indicates the solutions obtained from the one-dimensional space are as good as the one received directly from two-dimensional space. The possible loss of the search space may occur due to this dimension reduction. To eliminate this, there is a need for a search method that could obtain the solution directly from the two-dimensional space is of interested.

2. OBJECT-BASED EVOLUTIONARY ALGORITHMS

In the previous researches, a new methodology called an object-based evolutionary algorithm (OBEA) is proposed [6]. The method utilizes the benefits of evolutionary algorithms [7,8], graphical data manipulation [9], and simulation/animation [10]. The outline of the method has minor changes from the evolutionary algorithm given in Bäck [7]. However, an additional hill-climbing operation is added to enable the movements of pieces. Following is the general outline of OBEA.

0-7803-4869-9/98 $10.00©1998 IEEE

$t = 0$;
Initialization $P(0) = \{\bar{a}_0(0),...,\bar{a}_\mu(0)\} \in I^\mu$;
Evaluation $P(0) : \{\Phi(\bar{a}_0(0)),...,\Phi(\bar{a}_\mu(0))\}$;
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 $P'''(t) : \{\Phi(\bar{a}'''_0(t)),...,\Phi(\bar{a}'''_\lambda(t))\}$;
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 considered as the zero generation. Each individual is evaluated by an evaluation function, Φ, to find its fitness value. After this step, the next generation will be created iteratively by performing some operations until the termination criteria, ι, are met. This next generation is produced by performing a hill-climbing operator, $h\Theta_h$, a recombination operator, $r\Theta_r$, and a mutation operator, $m\Theta_m$, on the current population. The new individuals of new size λ are then 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 $t = t + 1$. More information to this guideline can be found in [7].

All other mechanisms--namely, individual representations, initialization process, evolutionary operators, fitness evaluation function, and termination criteria--are developed and implemented in a totally different way than the conventional evolutionary algorithms.

A. Individual Representation

Though the search mechanisms in the evolutionary algorithm could search for a set of points in an n-dimensional solution space, it cannot search for a set of areas or pieces. A solution of some problems, such as nesting problems, needs more than a set of points in the two-dimensional space. It requires a set of locations that correspond to the pieces on the regions. Those pieces should neither overlap each other nor extend out of the regions. Obviously, it is much too complicated to represent the nesting problem using the representation of the evolutionary algorithm due to the variable sizes of vertices representing those pieces. Therefore, each piece needs to be treated as a solid object. Also, the regions themselves must be bounded by solid edges, and each piece must be placed within a region. The collection of locations of these pieces is a solution of the problem, or the layout.

B. Initialization Process

Information used in the initialization might include the shape of the regions, the shape of the pieces, the number of those pieces that need to be placed, etc. The process uses this information within the first step to create the bounded regions. Then, each piece is placed within the regions in random order and location. However, it is possible that some piece may not be placed because the selected location is not vacant. Therefore, several attempts are allowed in finding an empty place for the piece. Also, some operators described in the following subsections, such as rotation, can be called to create more flexibility in placing the piece. All pieces may be reinitialized if there exists at least one piece that cannot be placed after exhausted attempts. The regions that contained all required pieces are called an initial solution or initial layout. More than one layout can be created during initialization. The whole set of these initial solutions is called an initial population, or the zeroth generation of layouts.

C. Fitness Evaluation Function

The fitness value can be calculated directly from the regions depending on the objectives of the problem. If the objective of the problem is to minimize the nesting area used by all pieces on a rectangular steel sheet, the fitness value might be the maximum position, or the rightmost position of all pieces on the sheet.

D. Termination Criteria

Usually, the algorithm is terminated by specifying a maximum number of generations. Some additional termination criteria may be added, such as maximum number of generations for which the best value remains the same, maximum computing time allowed in some real time applications, or a preset acceptable value is reached. In some situations, the average fitness of each generation can be used instead. Other termination criteria can be introduced upon requirement or objective of the particular problems.

E. Object-Based Evolutionary Operators

For an algorithm to be called the evolutionary algorithm, some evolutionary operators, also known as genetic operators, need to be presented. The ideas behind the operators are to change the location or orientation of some pieces so that a different layout is created. Some operators will make a significant change while some will make only a slight change. Three groups of operators can be categorized based on their actions and the results after being 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 below:

Hill-climbing Operators: In the two-dimensional space, the layout modification can be made by moving a single pattern. If the change done improves the fitness value of the layout, or at least remains the same, this situation is called hill-climbing. The evolutionary operators categorized as hill-climbing operators are "translation," "rotation," "rectangular-rotation," "touch-point," "piece-sliding" and "relocation."

a. Translation Operator. This operator is performed on a piece in a region by moving the piece one unit at a time toward preset gravitational forces. The center of the force can come from a single source or multiple sources on the two-dimensional plane. When these forces are applied, it makes all pieces move toward them at the same time. The mathematical representation of the operator obtained from the graphical data manipulation is called a unit translation. This unit is represented as a gravitational force with a direction. The unit translation equation can be represented as a matrix vector product 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 [9]. For example, if the lower-left corner of the two-dimensional plane is the center of the force, all pieces will be moved toward that corner, one unit for each piece in one generation. T_x and T_y, here, are equal to -1. However, some operations will not be performed if the next move creates overlapping. After the translation operator is performed for many generations, all pieces will be moved and packed at the lower-left corner of the region.

b. Rotation Operator. This operator can rotate an object to any particular degree. For example, in the hexagon piece packing problem, a specific degree of rotation, sixty degrees, may be needed. However, in the general case of the nesting problem where the pieces have alternative shapes, rotation in all degree must be allowed. Using the following matrix representation adopted from graphical data manipulation, an object which is the representation of the piece on the working space can easily be rotated to any degree of orientation.

$$[x' \; y' \; 1] = [x \; y \; 1] \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

c. Rectangular-Rotation Operator. For nesting problems that deal with rectangular pieces, a ninety-degree rotation operator may be required. Each of the four corners can be used as a rotation reference point. However, the word "rotation" may not be fully appropriate because this operator is created from two-step operators. The first step is to rotate a rectangle in an imaginary space using its lower-left corner as a reference point for ninety degrees. The second step is to move the piece back to the position of the reference point. For example, the selected pivot point is the 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. This operator is also very useful in irregular nesting problems. In mathematic terms, this rectangular rotation is a composite transformation of a rotation and a translation operator which can be written in the term of matrix multiplication as follows.

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

d. Relocation operator. It operates on a piece in a region by relocating the piece to a vacant position within the maximum value or the right most position of all pieces. The pieces should not overlap to each other, or extend beyond the boundary of the region, or over the maximum value line.

e. Touch-Point Operator. This operator contains two translation and one rotation operators. It is performed when any part of a piece touches another piece or the boundary of the region. This piece will be rotated about a selected touch point for a small degree. A matrix representation of the operator can be shown as follows.

$$[x' \; y' \; 1] = [x \; y \; 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -T_x & -T_Y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_Y & 1 \end{bmatrix}$$

These three matrix components are used for calculating a new location and a new orientation of the object. The three steps corresponding the matrix components are described as follows. First, the object is translated from the original position for $(-T_X,-T_Y)$ so that the touch point position is moved to a referent point, or lower left corner of a rectangular enclosure of the piece. Then, the object is rotated about that reference point. Finally, the object is

translated for (T_x, T_y) so that the touch point position returns to its original location.

f. Piece-Sliding Operator. The operation of this operator is similar to that of the rectangular rotational operator in that the object changes it coordinate and orientation simultaneously. The translating distance and a degree of rotation of the object are set to a small value so that the object will be moved in the sliding motion. The following is a matrix representation of this operator.

$$[x' \; y' \; 1] = [x \; y \; 1] \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

Mutation Operators: An operator that creates an offspring layout in which its fitness value can worsen is so-called a mutation. The mutation operator may create an offspring that either looks alike, or looks a lot different from the original layout. Several operators considered mutations are presented as follows:

a. Relocate-Away Operator. The idea of this operator is to relocate a piece to another position out of the maximum value line so that the rest of the pieces will have some room to be reorganized. In a densely packed situation, if the operation performs piece relocation within the maximum value line, the target area might not be vacant. One possible alternative to create a new offspring is to relocate the piece in the opposite direction of the gravitational force where a vacant space is guaranteed. For example, if the force is pulled to the lower-left corner of the plane, the piece should be relocated to the upper-right corner of the region.

b. Point-Mutation Operator. Relocate-away operator might eventually help in creating a better solution, and relocating a piece within the maximum value line may create a different look of the layout. Nevertheless, some pieces may already occupy that area. 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 begins in an overlapping position needs to be moved to the nearest position where no overlapping occurs. Each piece will be moved in the opposite direction of the gravitational force, or in the other words, moved toward the right end of the region.

c. Area-Mutation Operator. Similar to point mutation operator, all pieces which start from an overlapping position need to be moved. The difference here is that a whole specific area needs to be determined. The specific area can be any shape or any size. All pieces overlapped in the mutation area will be packed together and relocated to a new location. However, overlapping on the new location is also possible. The same overlapping technique used in the point mutation will be performed to create a non-overlapping layout.

Recombination Operators: This operator will be performed on two or more layouts by changing information to each other to create. The example of the operators are point-crossover and area-crossover.

a. Point-Crossover Operator. There are many ways to create a point crossover. For example, two layouts are chosen to be crossovered. A position on the first layout is selected, and a piece on that position is determined. Assume that the chosen piece is the piece labeled no. 7. At the same time, another piece on the same position on the second layout is also determined. Assume that the chosen piece on the other layout is the piece labeled no. 3. Next, the piece from each layout will switch to another layout creating a duplicated piece on each layout. The first layout now has two pieces of no. 3 but lost piece no. 7 while the second layout has two pieces of no. 7 but no piece no. 3. Since, the redundant pieces are not allowed in this step, the replacing technique may be used by replacing the piece no. 7 on the original of piece no. 3 so that the duplicating piece does not exist on the first layout. A similar process is to be performed on the second layout. Also, although an overlapping area is likely to exist, the overlapping solving technique used in point mutation will be called and be performed on the task as a part of this operator.

b. Area-Crossover Operator. The crossover is developed by combining the idea of area mutation and point crossover to create more possible recombinations of layouts. The area selected for performing the crossover can also be any size and shape. All pieces covered by the crossover area will be switched to another layout. Like the point crossover, it is possible to have more than two layouts involved in the area crossover. All redundant pieces are eliminated after the operation, all missing pieces are replaced, and the overlapping problem are resolved.

3. IMPLEMENTATION

Data representation used in OBEA is grid system. The main advantage of the method is the flexibility in implementing the system, i.e., a slower computer can use a lower resolution, and a faster computer can use a higher resolution.

4. EXPERIMENTAL RESULTS

In the previous research, two toy problems were used to show the potential in finding solutions using the methodology for some nesting problems [6]. Those results alone may not show the advantages over other methodologies. Some comparisons may be needed to present the usefulness of the OBEA.

A. Rectangular Nesting Problems

Data used for this comparison is obtained from [11]. The average of packing density of his testing results is 93.46%. Thirty-seven pieces from eight different patterns are tested (see Table 1). All pieces are allocated on a continuous sheet with a constant width of 30 units where a fixed population size of 91 is used and five hundred generations are run.

Table 1 The regular piece set used for comparison.

Pattern no.	Width (x)	Length (y)	Piece no.
1	12	10	1- 3
2	8	11	4- 7
3	9	13	8-13
4	4	5	14-17
5	9	10	18-20
6	6	8	21-26
7	10	12	27-35
8	15	7	36-37

In this work, most object-based evolutionary operators are applied. Ten trials are run. The best result, obtained form four out of ten times, is 94.41% while the worst result, two out of ten times, is 92.83%. The average of the packing density is 93.77%. The population used in this experiment are 30 and the termination generation is 200. Two different examples of the best results are shown in Figure 1 and 2.

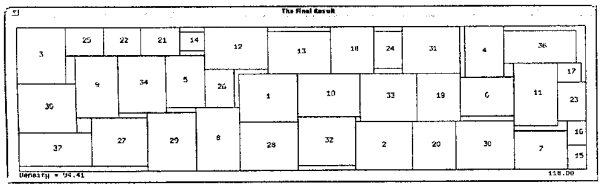

Figure 1 An example of the best result after ten trials

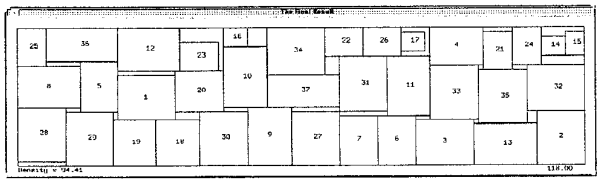

Figure 2 Another example of the best result

B. Irregular Nesting Problems

In this implementation, the irregular nesting problems are tested. The data of this test set is also obtained from [12]. The experimental results are obtained by generating 30 individuals using 30 irregular pieces on a continuous sheet with a width of 60 units. After generating for one thousand generations, the best result is around 82.34 % and the average is around 78.89% of packing density using lowest resolution, R=l. Figure 3 illustrates the best result of the population.

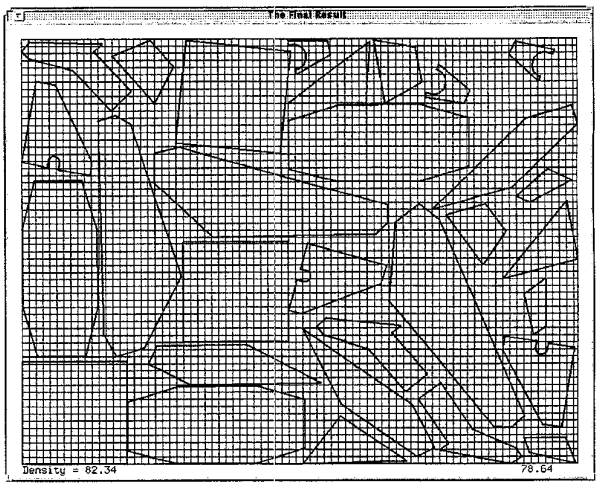

Figure 3 The Best result is obtained after the first 1000 generations (R=1).

Note that this packing density is calculated by counting number of squared units used by all pieces, 3885, then divided by the rightmost position of the rightmost piece, 78.64, and multiplied by the height of the sheet,60, which it is equal to 82.34. As seen, if the size of squared unit or grid is large, the gaps among the pieces are also large. The quality of the solution is depending on the size of the grid. This can be improved using a smaller size of the grid, higher resolution, so that only the smaller gaps can occur. However, the smaller grid size is selected to use on the second evolution, or after the best result from the first evolution is received. The data used in the above example is very small compared to the real data used in industry. However, to test the industrial data size, the higher resolution, e.g. R=10, is not possible to use due to the limits of computer storage and a very long period of computing time, up to two weeks per trial. In this study, five-hundred and forty-three pieces are run. Three layouts are used as the population. The result after running for five thousand generations using the first order resolution (R=1) is 80.78%. The result after the second evolution of two thousand generations using the second order resolution (R=2) is 77.25%. The final result after the third evolution of one thousand generations using

the fifth order resolution (R=5) is 74.89% (see Figure 4). This re-representation using higher resolution representation is used to start from the second evolution. Although much better quality of the final result is obtained, the time spent for this step is very long compared with the time that used during the first evolution. Further decrease the size of the grid will result in shortening only a small nesting length while the time spent for generating the final solution is increases dramatically.

5. CONCLUDING REMARKS

The object-based evolutionary algorithm is successfully applied to solve both the rectangular and irregular nesting problems. The idea of multi-dimensional representation for EA is introduced. The advantage of this approach is the ability to exploit the whole solution space. Animation during the experiments can enlighten the ways to discover new operators, answer a number of questions, and present this methodology perfectly with the least explanation. Population idea and the interaction between individuals are added to empower the algorithm on overall performance. All solutions from the algorithm yield no overlapping patterns. The experimental results on the rectangular and irregular nesting problem are around 93.7% and 78.9% respectively.

For the future work, some new evolutionary operators may be introduced to customize some specific problems. Some pre-processor and post-processor may also be used to reduce unnecessary computation. Improve versions of shape-filling algorithms for the internal representation may be needed. An algorithm for calculating the area of an irregular piece may be required. Intensive experimentation with different problems may be tested. Robust parameters for the general nesting problem may be done by using different methods such as design of experiments, taguchi method, or neural networks. Parallel implementation for the OBEA on parallel computer may be helped in reducing the total computation time.

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] Kanchitpol Ratanapan and Cihan H. Dagli, "An Object-Based Evolutionary Algorithm for Solving Rectangular Piece Nesting Problems," *Proceedings of 1997 IEEE International Conference on Systems, Man and Cybernetics*, Vol. 2, Oct. 1997, pp. 989-994.

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

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

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

[10] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill Inc., 1991.

[11] Pipatpong Poshayanonda, *Genetic Neuro-Nestor*, Ph.D. Dissertation, Engineering Management Department, University of Missouri-Rolla, 1994.

[12] Kanchitpol Ratanapan and Cihan H. Dagli, "An Object-Based Evolutionary Algorithm for Solving Irregular Nesting Problems," to be appeared in *ANNIE'97*, 1997.

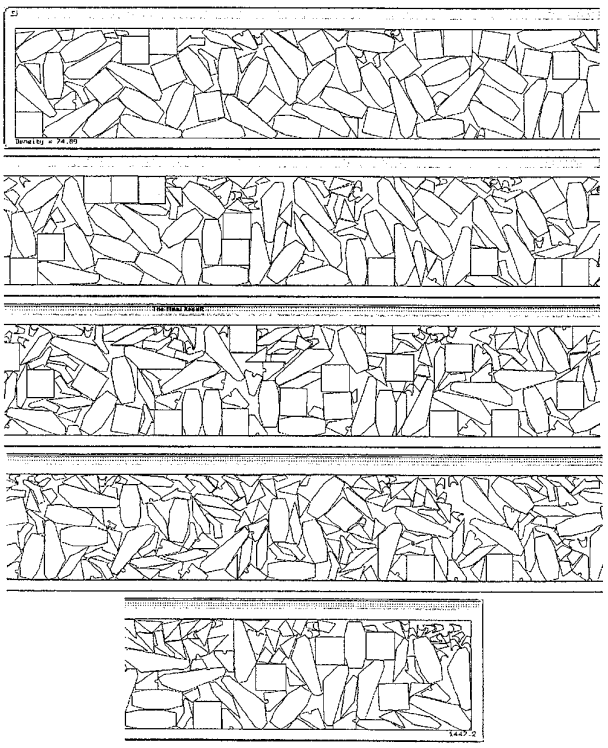

Figure 4 The result of the industry size nesting problem at the fifth order resolution.