Hybrid Approach for Optimal Nesting
Using a Genetic Algorithm and
a Local Minimization Algorithm

Kikuo Fujita, Shinsuke Akagi and Noriyasu Hirokawa
Department of Mechanical Engineering
for Industrial Machinery and Systems,
Osaka University
Suita, Osaka, JAPAN

Abstract

In layout design problems including blank nesting, the positions
and directions of layout elements must be determined so as to
minimize the total space. It is difficult and computationally
time-consuming to find the optimal solution for such layout
problems, because they include a lot of underlying combinational
conditions. In this paper, we present an approach for
optimal nesting by combining a genetic algorithm and a local
minimization algorithm. In the approach, the genetic algorithm
is used for handling the combinations which are represented
in the string, and the local minimization algorithm is used for
determining the embodiment layout under the fixed combinations
so as to minimize the scrap volume which is corresponding to
the fitness value in the genetic algorithm. And we present an
example for showing the effective nesting result produced by this
approach.

1 Introduction

Layout problems are found in various kinds of design problems
such as blank nesting problems and component placement
problems in plant design and VLSI design. Those layout
problems, in general, include many combinational conditions and
it is difficult to find an optimal solution by means of traditional
hill-climbing methods of mathematical programming techniques.
Therefore, heuristic search procedures are applied for many
cases, and recently some probabilistic approaches such as
simulated annealing methods (SA, [van Laarhoven, 1987]) and
genetic algorithms (GA, [Goldberg, 1989][Michalewicz, 1992])
are used. As for the nesting problems, Cheok et al. introduced
some heuristics [Cheok, 1991] and Jain et al. applied simulated
annealing [Jain, 1990]. Kröger et al. used a genetic algorithm
for the case that the pieces are rectangular [Kröger, 1991].
Among these probabilistic approaches, the interests on genetic
algorithms are going to be grown as a novel and effective
approach in the fields of combinational optimization and global
optimization problems.

The origin of genetic algorithms was found in the studies for
simulating the mechanism of the natural evolution and selection
by John Holland [Goldberg, 1989]. In genetic algorithms,
the design variables to be optimized are coded as a string
corresponding to a gene, and it is optimized through the
repeating manipulations with the genetic operators for a set of
strings. Its characteristics are summarized as follows: (1) It
is a probabilistic approach using a set of tentative solutions,
which is called a population of individuals in the terminology
of genetic algorithms. (2) Binary or character strings are used
for representing a solution. And, (3) how subparts of strings, that
are corresponding to building blocks, effect on the optimality is
important for better search performance.

As aforementioned, Kröger et al. used a genetic
algorithm for the rectangular nesting problems [Kröger, 1991].
And, Shahookar et al. applied another genetic algorithm
to the standard cell placement problem in VLSI design
[Shahookar, 1990]. However, they dealt with simple problems
that layouts could be represented only with discrete or
combinational variables without continuous variables. Their
approaches can not be immediately applied to the cases that the
pieces with free shapes should be arranged, in which the layout
of a piece must be represented directly with its position and
direction in a coordinate system.

In this paper, we propose a hybrid approach combining a
genetic algorithm and a local minimization algorithm for such hard nesting problems. In the approach, the nesting problem is considered as a global optimization problem which includes a number of multiple-peaks, and the description of a layout is hierarchically separated into the two parts; the combinational part and the continuous part. Accordingly, a genetic algorithm for combinational optimization is used for the former and a local minimization algorithm in a continuous subspace is used for the latter, respectively.

2 Formulation of Nesting Problem

Firstly, the nesting problem and its formulation as an optimization problem are described.

In this paper, we consider the nesting problem that the plural pieces, Piece\(_i\) (\(i = 1, \ldots, n\)), are arranged into a plate with a certain breadth, \(B\), so as to minimize the scrap volume (Fig. 1). The formulation of this kind of nesting problem is shown in the following.

Besides, the shapes of pieces are assumed to be convex polygons in order to reduce the computation time in the computational example which will be shown in the later section.

2.1 Design Variables

The position of the key point in a coordinate system; \((x_i, y_i)\), and the direction of the key axis; \(\theta_i\), for the respective piece, Piece\(_i\), are taken as the design variables. The position of the plate is determined so as to fit its left-side edge to the piece which is located in the most left position and to fit its lower-side edge to the piece which is located in the lowest position under the fixed positions and directions of the pieces. After that, the necessary height of the plate, \(h\), and the overhang value, \(\delta\), can be calculated by referring the determined position of the plate.

2.2 Constraints

The two following constraints must be considered.

- The constraint in order to avoid any overlaps among the pieces: The overlap between Piece\(_i\) and Piece\(_j\) is denoted with overlap\(_{i,j}\). Then, the total of the overlaps are represented with the following formula:

\[
\text{Overlap} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{overlap}_{i,j} 
\]

Then, the constraint is formulated as follows.

\[
\text{Overlap} = 0
\]

- The constraint in order that all of pieces are located within the breadth, \(B\), of the plate: This constraint is defined as follows by using the overhang value, \(\delta\).

\[
\delta = 0
\]

2.3 Objective Function

The objective is to minimize the scrap volume, Scrap, as mentioned above. This can be defined as the following equation by using the variables shown in Fig. 1.

\[
\min : \text{Scrap} = (B + \delta) \times h - \sum_{i=1}^{n} a_i 
\]

where, \(a_i\) is the area of Piece\(_i\).

Besides, the reason why the part, \((B + \delta)\) in Equation (4) is not represented simply as \(B\) is that the overhang constraint defined with Equation (3) may be violated in the process of the local minimization, which will mentioned in the later section, though it should be satisfied at the end of the process.

3 Hybrid Approach for Nesting Problem

3.1 Building Block Hypothesis in Genetic Algorithms

Genetic algorithms are the search algorithm using population of strings, and how subparts of strings effect on the optimality and how they are preserved and produced through the genetic operators are important for better search performance, as mentioned in Introduction. The efficiency and advantage of genetic algorithms is theoretically explained with “building block hypothesis” (e.g., [Goldberg, 1989]).

The concept called schemata is usually introduced for discussing superiority or inferiority of the subparts of strings. In genetic algorithms, the respective string itself is not meaningful,
3.2 Concept of Hybrid Approach

When the building block hypothesis is contrasted with the nesting problem, the following context emerges: In the nesting problem, there are a variety of substructures of layouts which are effective and meaningful for reducing the scrap volume. If some of them could be well merged each other into the whole layout, better layouts would be produced. Figure 2 gives an example of building blocks in the nesting problem mentioned in the later section, where a new layout called ‘Child’ is produced by merging two layouts called ‘Parent 1’ and ‘Parent 2’ through a crossover operator. Child inherits some substructures; for example, ‘h’ and ‘e’ from Parent 1, and ‘j’, ‘a’, ‘b’, ‘c’ and ‘f’ from Parent 2. These substructures are corresponding to building blocks. Moreover, the respective substructure of plural pieces can be recognized to be composed of a set of pairs of pieces, which we call “meaningful neighboring relationships”. For example, the ordered substructure, “j, a, b, c and f”, is broken into the pairs, “j and a”, “a and b”, “b and c”, and “c and f”. In this concept, the pair not only means a list of two pieces but also includes the dimensional relations; in which directions they are touching each other. This point is different from the traditional coding method used for solving the traveling salesman problem with genetic algorithms.

Accordingly, we define the string for nesting problems based on the meaningful neighboring relationships. That is, the chained list of the pairs forms an ordered list of pieces corresponding to the string used in genetic algorithms. However, traditional genetic algorithms manipulate only the combinations within the ordered list. And it can not directly manipulate the embodiment positions and directions of the pieces, which are represented with a large number of continuous variables. Therefore, we combine a local minimization algorithm with a genetic algorithm in order to handle them.

This hybrid concept summarizes as the following approach for the nesting problem:

**Genetic algorithm** — The neighboring structures of the pieces are manipulated with the genetic operators.

**Local minimization algorithm** — The embodiment positions and directions of the pieces are determined by numerical search algorithm under the neighboring relationships fixed with the genetic algorithm.

3.3 Formulation for Hybrid Approach

The formulation of the nesting problem shown in the previous section must be arranged for the above hybrid approach. The arrangement is shown in the following.

The neighboring relationships, i.e., the ordered list of the pieces, $Piece_i$ ($i = 1, \ldots, n$), are restricted in the order, $Piece_{e_1}$, $Piece_{e_2}$, $Piece_{e_3}$, …, $Piece_{e_n}$, which is used as a string in the genetic algorithm. In addition to the order, the string includes the relatively positional relationships between the two pieces which are in the neighbor.

Corresponding to the string representation, in order to maintain the neighboring relationships in the process of local minimization, we introduce the following supplementary objective function in addition to the constraints and objective function which were mentioned in Section 2.

$$\min. \text{Dist}2 = \sum_{i=1}^{n-1} \text{distance}_{e_i, e_{i+1}}^2$$

where, $\text{distance}_{e_i, e_{i+1}}$ is the distance between $Piece_i$ and $Piece_j$. 

![Building Block Diagram](image-url)
The axes of the local coordinate systems are defined as the differential angle between the line segment and both of the $x$ and $y$ directions. The two local coordinate systems are introduced. And then, the position of the key point and the angle of the key point is its origin and that the key direction is its $x$ axis. Based on the local coordinate systems, the position of its key point, the part in the outside of both pieces, and the part on applying genetic operators, which will be mentioned in the next section.

With the above denotation, the design variable vector $u$, which is manipulated with the local minimization algorithm, is defined as follows:

$$u = (\phi_1, \psi_1, l_1, \phi_2, \psi_2, \ldots, l_n, \phi_n)$$

Besides, the key point of the respective piece is set to the center point of the circumscribed circle which has the minimum radius of its various circumscribed circles for the convenience of computation.

## 4 Hybrid Algorithm for Nesting Problem

The hybrid algorithm for the nesting problems based on the concept mentioned in the previous section is shown in this section.

### 4.1 Outline of Hybrid Algorithm

Figure 4 shows the outline of the algorithm, in which the local minimization process is included in the evaluation step of individuals in the genetic algorithm. The genetic algorithm is composed of evaluation, scaling, selection, crossover and mutation steps as similar with traditional genetic algorithms. The local optimization algorithm is applied for determining the embodiment layout based on the string and the initial values of the design variables $u$ which are determined through the above genetic operation steps.

The respective steps of the approach are explained in the following subsections. Besides, as for the genetic operations, there are many versions and it is important to select effective ones from various ones, as mentioned in the previous section. The following genetic operators used in our algorithm are circumspectly selected for the nesting problem.

### 4.2 Population Exchange

In the genetic algorithm, as shown in Fig. 4, first of all, the initial population which consists of $N$ individuals is randomly generated. After that, the next population is produced from the current population through the various genetic operations. And these population exchanges are repeated until the repetition number reaches sufficient iterations.

### 4.3 Evaluation

The fitness value $f_i$ of an individual, which is corresponding to the objective function to be maximized in traditional optimization
techniques, is calculated from the scrap volume $Scrap_i$ of the layout with the following equation (e.g., [Rao, 1991]).

$$f_i = \left( \frac{Scrap_{\text{max}} - Scrap_i}{Scrap_{\text{max}} - Scrap_{\text{min}}} \right)^2 \quad (i = 1, \ldots, N)$$

where, $Scrap_{\text{max}}$ and $Scrap_{\text{min}}$ are the maximum and minimum value of the scrap volumes of the individuals in a population, respectively. Through the above equation, the fitness value for a respective individual can be translated into the relative one.

### 4.4 Scaling

Next, the following operations are applied for the fitness values determined by Equation (7) in order to effectively execute the selection which is mentioned in the next subsection.

- **σ truncation** (e.g., [Goldberg, 1989]) — Based on the average $f_{\text{avg}}$ and the standard deviation $\sigma$ of the fitness values, the individuals which satisfy the equation; $f_i < f_{\text{avg}} - c \sigma$, are eliminated from the population pool for the selection which is mentioned in the next subsection.

- **Linear Scaling** (e.g., [Goldberg, 1989]) — Moreover, the fitness value of the rest of individuals are rearranged with the following equation.

$$f'_i = af_i + b$$

where, the coefficients $a$ and $b$ are determined so as to satisfy the equations; $f'_{\text{max}} = c_{\text{mul}} f'_{\text{avg}}, f'_{\text{avg}} = f_{\text{avg}}$. Besides, a $C_{\text{mul}} = 1.6$ is used in the computational example mentioned in the next section.

These rearrangements are expected to effect on maintaining the variety of the individuals in a respective population and finding globally optimal layouts.

### 4.5 Selection

After rearranging the fitness values, the pairs of individuals, which are the parents for mating, are selected from the population pool based on the fitness values, $f'_i$, until the number of pairs reaches the necessary number. For this selection step, we use the two strategies; “expected value plan” and “elitist plan” (e.g., [Goldberg, 1989]). With the expected value plan, the number of the copies of a respective individual selected is certainly determined according to the expected number calculated from the fitness value, and this plan is effective for reducing the stochastic errors of selection. With the elitist plan, the individual with the highest fitness value, i.e., the best layout in the parent population, is directly copied into the child population without applying the crossover and mutation operators.

### 4.6 Crossover

Two child individuals are generated by mating a selected pair of parent individuals with the crossover operator which is called “order crossover” (e.g., [Shahookar, 1990]) under the crossover probability $P_c$.

Figure 5 illustrates the order crossover. It operates as follows:

(i). The individuals of the selected pair is named ‘Parent 1’ and ‘Parent 2’, respectively.

(ii). The crossover point is selected randomly.

(iii). As for the left part of the crossover point, the substring is simply inherited from Parent 1 to Child 1. And as for the right part, the genes, i.e., pieces, which are not included in the left part of Parent 1 are arranged into Child 1 in the sequence that they are included in Parent 2. With these procedures, a new full string, Child 1, is produced.

(iv). The operation (iii) is repeated by exchanging the role of the Parent 1 and Parent 2 for producing Child 2.
On the other hand, the relatively positional relationship with the previous and behind pieces, i.e., initial values of \( b_i \), \( f_i \), and \( y_i \), is inherited from the parents in the following way:

- Fundamentally, each piece simply inherits the relatively positional relationship from its parent. That is, each piece \( Piece_i \) is re-connected with other pieces, taking the variables, \( l_i \), \( f_i \) and \( y_i \), with it.
- In the case that the piece, \( Piece_k \), was the first one in the parent string and that is going to be moved to the other position in the child string, the initial values of \( l_k \) and \( f_k \) are set up randomly.

In Fig. 5, the symbols; ‘+’, ‘-’ and ‘=’, represent from which parent the relationship is inherited. the symbol ‘+’ represents that it is inherited from \( Parent_1 \), and the symbol ‘-’ represents that it is set up randomly.

The advantage of this crossover operator is that the substructures, i.e., the chains of meaningful neighboring relationships, included in the left parts of the strings are likely to be preserved and the effective and suitable substrings, i.e., building blocks, are likely to be gradually grown to the longer substrings from left to right.

An example of the order crossover in which it acts effectively for reducing scrap volume will be shown in Fig. 9 in the computational example.

### 4.7 Mutation

Furthermore, the mutation operator named “remove and reinsert” (e.g., [Manderick, 1992]) is applied to a respective string under the mutation probability \( P_m \).

Figure 6 is an example of the mutation. First a piece is randomly selected from the string and it is removed from the string. Second the removed piece is inserted into the other position which is also randomly selected, involving the relatively positional relationship in the similar way of the crossover operator. The purpose of introducing this kind of operator is to avoid rapidly losing the variety of the schemata and to find the globally optimal solutions.

### 4.8 Realizing the Layout — Applying Local Minimization

Finally, the respective individual which is determined as a string through the above genetic operations is embodied into a actual layout. That is, the layout is determined with a local minimization procedure under the layout structure defined in the string, which are constrained with the neighboring relationships defined with Equation (5) and the initial values of the design variables \( u \) inherited from the strings of its parents. In this paper, we use Quasi-Newton method as a local minimization algorithm. In order to formulate the problem as an unconstrained minimization problem, the following objective function is defined by including the supplementary objective function; Equation (5), and the constraints; Equations (2) and (3), into the primary objective function; Equation (4), as penalty terms.

\[
\begin{align*}
\text{min. Obj} & = \text{Scrap} + k_1 \times \text{Dist}^2 \\
& + k_2 \times \delta^2 + k_3 \times \text{Overlap}^2
\end{align*}
\]  

where, \( k_1 \), \( k_2 \) and \( k_3 \) are weighting factors. Besides, \( k_1 = 250 \) and \( k_2 = k_3 = 500 \) are used in the computational example mentioned in the next section. These are the operations used in the hybrid algorithm. The population exchanges involving these operations are iterated in several tens times and then desirable layout will be obtained.

### 5 A Computational Example

In this section, a computational example where 12 pieces are arranged into a plate is demonstrated. In this example, the parameters on the genetic algorithm is set as follows; the crossover probability; \( P_c = 0.6 \), the mutation probability; \( P_m = 0.03 \), the population size; \( N = 31 \). Incidentally, these values and other setting mentioned in the previous section are roughly set up based on references and they are not sufficiently adjusted.
Figure 7: Convergence of GA

Figure 7 shows the converging history of the algorithm. The vertical axis of the graph is scrap volume, the horizontal axis is generations. In the figure, the history of minimum, average, and maximum values of scrap volumes in the population of respective generations are shown. It is recognized that the set of solutions were gradually converging, and that after the 32nd generation the variety of the strings has been almost lost, and the solutions have been converged.

Figure 8 shows the history of the best layouts in some generations corresponding to Fig. 7. In the respective layout, the alphabetical symbols are the indexes for the respective pieces. Because the elitist plan is used in the selection step of the genetic algorithm as mentioned in the previous section, the best layouts in the generations are not always exchanged. From the history of the best layouts, it is ascertained that the substructures of the layouts are gradually grown into the better layouts according to the aim of using the order crossover.

Figure 9 shows an example of the crossover operations between the 24th and 25th generations, which produced the best layout in the population of the latter generation. In the example, the parents resemble each other in their strings, but they are slightly different in their embodiment layouts. The child is produced by mating them at the crossover point. Though the initial layout of the child has two overlap areas and a little overhang value, it is arranged into a better layout than both of the parents by the local minimization procedure.

The above calculation takes about 12.5 hours with a Sun SPARC station 10 / Model 41. However, the computation time for calculating the overlaps in Equation (1) and the distances in Equation (5) is very large, and if it could be reduced, the calculation time would be much improved.
6 Summary

This paper presents a hybrid approach for the nesting problems by combining the genetic algorithm and the local minimization algorithm. The characteristic of the approach is that the combinational conditions of layouts are represented with a form of ordered lists, which representation enables the hybridization of the two algorithms. The computational example shows the effectiveness and potential of the approach. However, since the property of solutions and convergence are dependent on the kinds of genetic operators such as crossover and mutation, their various parameters, the population size, etc., which is generally found in the applications of genetic algorithms (e.g., [Goldberg, 1989][Starkweather, 1992]), it is necessary to improve the performance of the hybrid algorithm by comparing the kinds of genetic operators and the values of parameters.

Acknowledgments

We would like to acknowledge the support provided by The Ministry of Education, Science and Culture of Japan through Grant-in-Aid for General Scientific Research 044521319.

References
