Journal of Intelligent Manufacturing, 15, 201–218, 2004

Genetic neuro-nester

PIPATPONG POSHYANONDA and CIHAN H. DAGLI

Smart Engineering Systems Laboratory, Engineering Management Department, University of Missouri-Rolla, Rolla, MO 65401, USA
E-mail : dagli@umr.edu

Received March 2002 and accepted June 2003

In this paper, the integration of artificial neural networks and genetic algorithms is explored for solving uncured composite stock cutting problem, which is an NP-complete problem. The input patterns can be either rectangular or irregular, and the proposed approach can accommodate any orientation and size restrictions. A genetic algorithm is used to generate sequences of the input patterns to be allocated. The scrap percentage of each allocation is used as an evaluation criterion. The allocation algorithm uses the sliding method integrated with an artificial neural network, based on the adaptive resonance theory (ART1) paradigm, to allocate the patterns according to the sequence generated by the genetic algorithm. The results obtained by this approach give packing densities on the order of 80–95%.

Keywords: Nesting, genetic algorithms, neural networks, optimization

1. Introduction

Different competitive thrusts have played leading roles in creating manufacturing strategies and systems during the past decades (Dagli, 1994). Nowadays, flexibility of the manufacturing system has become a competitive thrust. Mass production systems used in the past decades were operationally focused and emphasized short-term financial performance. Flexible manufacturing systems that respond to differing customer demands quickly replaced the mass production systems having only product focus. A flexible automated manufacturing system that integrates the product design and the production system must be created.

Currently, a large majority of firms have, or are in the process of, transferring their product design process into an electronic environment. There is a need for an automated stock cutting system that provides flexibility with efficient usage of high cost materials as the use of composite materials has increased dramatically over the past decade (Dagli *et al*., 1990). In addition to the flexibility of the manufacturing systems, cost still plays a major role in the stock cutting. A 1% reduction of the high cost materials in some industries, means millions of dollars in savings each year. Besides the major cost reduction of direct materials, the automated system also provides direct labor and overhead cost savings to the company. Inventory and manufacturing lead times are also reduced by system automation.

This research work provides performance enhancement to production systems in the stock cutting industry. It concentrates on nesting uncured composite materials, which is categorized as a two-dimensional stock cutting problem. The problem is to create the cutting patterns for the required patterns that are not restricted to any shapes or orientations. An automated system that generates nests with minimum scrap is proposed.

The problem of allocating irregular and/or rectangular patterns arises frequently in applications where it has to be determined how a set of two-dimensional shapes will fit onto a large stock sheet of finite dimensions, in such a way that the resulting scrap is minimized. This problem is common to many industries such as aerospace, shipbuilding, clothing, shoe manufacturing, VLSI design, steel manufacturing, flat glass and furniture.

The two-dimensional single plate rectangular

pattern allocation problem is NP-complete (Garey and Johnson, 1979). The problem becomes more complicated for multiple plates, variable length plates or irregular patterns. Because of the diversity of structures in real world stock cutting problems, there exists no general standard method for solving them. Solution approaches proposed over the years can be categorized into three major areas: optimization, heuristics, and the use of emerging technologies. The approaches proposed are summarized in many survey articles (Hinxman, 1980; Golden, 1976; Haessler and Sweeney, 1991; Dowsland, 1992; Dyckhoff, 1989). In the following paragraphs, the research work carried out in these areas is discussed.

1.1. *Optimization approaches*

Initially, one-dimensional stock cutting problems, wherein the other two dimensions of the items being cut are assumed constant, are studied. A mathematical formulation of the one-dimensional problem was first proposed by Kantorovich in 1939, but it was not published in English until 1960 (Kantorovich, 1960). Problems in these areas have been formulated into mathematical programming problems by many researchers (Paull, 1956; Metzger, 1958; Vajda, 1958). Linear programming is used by Eisemann (1957), Gilmore and Gomory (1965), Farley (1988, 1990), Haessler (1980). Gilmore and Gomory (1961) used the technique called delayed column-generation. The approach is later modified and used by Dyckhoff (1989), Scheithauer (1991), Haessler (1980), Heassler and Sweeney (1991).

Besides the linear programming methods, other approaches such as branch-and-bound (Pierce, 1966; Goulimis, 1990); Monte-Carlo simulation (Duta and Fabian, 1984), and, especially, dynamic programming have been used extensively in this problem setting (Richter, 1992; Hahn, 1986; Adamowicz and Albano, 1976a, b). Christofides and Whitlock (1977) propose a tree-search algorithm. Gemmill and Sanders (1991) uses optimization homotopy which is a stochastic technique.

The proposed optimization approaches have restrictions in application due to the NP-complete nature of the problem. Many applications involve an astronomical number of variables, so the amount of time required to generate solution patterns may not be practical. Furthermore, switching from an optimal fractional-valued solution to an optimal integer-valued solution is not easy. If the demands are in small enough quantities, then the optimal integer-valued solution may be quite different from the original optimal fractional-valued solution. To overcome these difficulties various heuristic approaches are proposed.

1.2. *Heuristic approaches*

Albano (1977) offers an interactive algorithm to improve the two-dimensional layout in which decision maker interventions are included. Albano and Orsini (1980) combines heuristic and exact techniques to find an approximate solution.

Dagli and Nisanci (1981), and Dagli and Tatoglu (1983, 1987) propose heuristic algorithms which can process both rectangular and irregular patterns. At the first stage of Dagli and Tatoglu's (1983, 1987) procedure, initial allocation of patterns to the plates is made through mathematical programming; then, based on this initial allocation, detailed two-dimensional allocation is accomplished through heuristic algorithms in the second stage. Extensions of this work that combine heuristics and optimization methods are presented in Dagli (1988a, b).

Beasley (1985), Ghandforoush and Daniels (1992) propose heuristic methods to handle guillotine-constrained problem. Daniels and Ghandforoush (1990) propose an algorithm to solve the non-guillotine-constrained two-dimensional cutting stock problem. A mistake in their equation is later commented on and updated by Dowsland (1992) and George (1992).

Wang (1983) proposes two combinatorial methods that solve a constrained rectangular stock cutting problem where only guillotine cuts are allowed. Wang's algorithm is improved by Oliveira and Ferreira (1990) to increase computational speed, using reduced memory. Another modification and extension of the Wang's algorithm is proposed by Viswanathan and Bagchi (1988, 1993).

Chung *et al*. (1990) proposed the heuristic solution for allocation of irregular patterns on the resource that is highly irregular, with not only irregular boundaries but also defective areas. Haims and Freeman (1970), Hinxman (1973), Adamowicz and Albano (1976b), Albano and Sapuppo (1980), Bengtsson (1982), Qu and Sanders (1989), Dietrich and Yakowitz (1991), Foronda and Carino (1991) are some of the other

papers that propose heuristic solutions to the stock cutting problem.

1.3. *Emerging approaches*

Instead of using only optimization or heuristic approaches as a solution to the stock cutting problem, many researchers have been investigating the possibility of combining these methods into a solution approach to overcome the disadvantages of each of them. An example of this type of solution is proposed by Schollmeyer *et al*. (1991). In their model, a heuristic expert system generates many possible solutions using knowledge bases that contain the rules related to cutting restrictions, nesting methods, and allocation goals. Then, they use a linear programming model to select those solutions which will result in a minimum amount of scrap.

During the past decades, new technologies such as simulated annealing, neural networks, and genetic algorithms, or hybrid approaches, have also been used as solution approaches. They are discussed in the following paragraphs.

Simulated annealing proposed by Kirkpatrick *et al*. (1982) is inspired by the metal annealing process. It has the ability to avoid entrapment at a local minimum, which is very important in solving optimization problems. Simulated annealing is designed to optimize functions of several hundred variables or more, and is especially attractive when the functions are not smooth and have many local minima. An algorithm that uses simulated annealing as a solution to the bin packing problem is proposed by Kampke (1988). Harris and Zinober (1988) implement this technique to solve the problem of reducing the number of stacks of different panel types around the operator during cutting in the wood industry. Other researchers (Dagli, 1990; Dagli and Hajakbari, 1990; Lutfiyya *et al*., 1992) apply this technique to the two-dimensional nesting problems. Lirov *et al*. (1992), combines simulated annealing with case based reasoning as a knowledge-based solution to a cutting stock problem.

Neural networks could contribute greatly to the solution of the stock cutting problem due to their ability to identify various pattern configurations and their parallelism (Dagli, *et al*., 1990). The neocognitron is a multistage network which simulates the human vision system. Because this network is capable of training with both unsupervised learning and supervised learning, it is selected in the literature as a network paradigm to identify scrap patterns generated for the combination of various composite patterns. This paradigm is also used as a feature selector for various patterns to determine the weights that measure the degree of match between patterns as a part of the simulated annealing solution approach (Dagli, 1990).

Poshyanonda *et al*. (1992a, b) use a back-propagation neural network in the pattern generation phase and produce these patterns for a linear programming model to find the solution. They use adaptive resonance theory (ART1) to identify the scarps generated by the process in each stage (Poshyanonda and Dagli, 1992).

Genetic algorithms also play an important role in many optimization problems, including stock cutting problems. This technique simulates the natural evolution process which is simple but powerful. Gemmill (1992) used genetic algorithms as a solution to an assortment problem and compared his result with Beasly (1985). Genetic algorithms have outperformed the traditional approach when the number of possible stock sizes increased. Poshyanonda and Dagli (1992) combine the genetic algorithm with a heuristic method to solve a nesting problem. Genetic algorithms are used to generate a proper sequence of the required patterns to the heuristic nester by exploring the search space.

The following observations can be made regarding the approaches proposed for solving stock cutting problems:

(1) Heuristics play an important role, since they are flexible enough to take into account various additional restrictions and objectives appearing in practice. The quality of heuristics is generally problem specific, and they can identify a pattern which is ''good'' for the particular problem in question. They also can be integrated into intelligent decision support systems easily.

(2) In almost all proposed linear programming models, a two stage approach is used. Initially various cutting patterns are generated based on given stock sheet dimensions and input pattern shapes, then a decision variable is assigned for each cutting pattern and a linear programming model is formulated.

(3) Generally, linear programming approaches are not widely used in practice. Most of the time, heuristics are preferred for selecting patterns. This is

basically due to the large number of possible patterns combinations (in the order of hundred millions) which cannot be represented in linear programming formulations.

(4) The linear programming approach cannot be applied to any of the irregular-shaped problems due to its difficulty in generating overlapping constraints. Most of the literatures related to irregular-shaped patterns are heuristics based.

(5) However, heuristic approaches do not produce optimal solutions and are yet constrained to a specific problem. An approach should produce a near optimal solution, should be able to handle generic irregular-shaped patterns, and should not require extreme computation time. New approaches such as simulated annealing, artificial neural networks and genetic algorithms seem to satisfy these needs and produce a better solution to the stock cutting problems. Each approach has advantages and disadvantages over the others.

A combination of these techniques that utilizes the advantages of each may produce a better solution to the problem. A solution approach that combines these emerging techniques to solve a general two-dimensional cutting stock problem is described in the following sections.

2. The proposed model

The patterns used in this study are not restricted to any particular shapes or orientations. Holes within a pattern are not considered to be as a material to be utilized. Margins between patterns that may be required by the cutting operations are omitted. The composite material of the stock sheet is assumed to be equally distributed over the material's surface and the stock sheet is defined as rectangles which can be divided into two different types, namely, sheets which are restricted in both width and length, and sheets restricted only in width having infinite length. In the problem of multiple stock sheets which are restricted to both width and length, all of them must have the same dimension.

The objective of the problem is to minimize the cost function $f(x)$, which is in this case, the total area required to allocate patterns on a two dimensional space. The problem of allocating a set of N required patterns $\{\mathrm{P1}, \ldots, \mathrm{PN}\}$ on stock sheet(s) can be defined as

$$\begin{aligned}\text{Minimize } f(N) &= \text{Total area of the stock sheet used to allocate } N \text{ patterns} \\ &= \text{width of the stock sheet} * \text{total length of the sheet used}\end{aligned}$$

without violating one of the following constraints:

- There must be no overlaps between patterns.
- Patterns must be totally allocated within the stock sheet boundary.

In this paper, the width of the stock is fixed for each of the nesting problems, therefore, the objective function of the problem can be defined as minimizing the total length of material used. In other words, this is a problem of minimizing the maximum x coordinate of all patterns allocated on the stock sheet(s) which can be defined as

$$\text{Minimize } f(x) = \max\{x_{ij}\}$$

subjected to

$$\begin{aligned}&x_{ij} \geq 0,\\ &x_{ij} \leq L,\\ &y_{ij} \geq 0,\\ &y_{ij} \leq W, \text{and}\\ &P_i \text{ does not overlap with } P_k\end{aligned}$$

where W is the width of the stock sheet(s), L is the length of the stock sheet(s) and (x_{ij}, y_{ij}) is the location of the jth point that represents the pattern $i(P_i)$.

To be able to compare the performance between different results, packing density of the stock sheet is used as a comparison unit which can be defined as

$$\text{Packing density} = \text{Total usage area/ Total required area}$$

2.1. *Representation scheme*

In this paper, each pattern is represented by a list of vertex coordinates $[(x_1, y_1)\ldots(x_M, y_M)]$ where M is the number of vertices used in representing the pattern. During the allocation process the degree of overlap among patterns on the stock sheet should be tested. With the coordinate representation, the overlap test can be performed by checking intersections between line segments of the patterns allocated on

the stock plate. The time required by this process increases exponentially when the number of patterns allocated on the stock sheet is increased.

In order to overcome this difficulty, the matrix representation (Dagli, 1990) is used. Each pattern is represented by a matrix of size K by L which is the smallest rectangular enclosure of the irregular shaped pattern. Each value of the matrix at location i, j, (V_{ij}) is defined as

$$V_{ij} = \begin{cases} 1 & \text{if location } i,j \text{ is inside the pattern} \\ 0 & \text{otherwise} \end{cases}$$

An example of the matrix representation is given in Fig. 1(a) and (b). The values of K and L are determined based on the desired precision for the computations.

A stock sheet is represented by the matrix $S(t)$ where $S_{ij}(t)$ is the value at the location (i, j) on the stock sheet after the first t patterns have been allocated. $S_{ij}(t)$ is computed as below:

$$S_{ij}(t) = S_{ij}(t-1) + P_{ij}(t) \quad \text{and} \quad S_{ij}(0) = 0$$

where $S_{ij}(t-1)$ is the value of the matrix at the location (i,j) on the stock sheet after the first $t-1$ patterns have been allocated; $S_{ij}(0)$ is the value of the matrix at the location (i,j) on the stock sheet at the beginning of the allocation process; $P_{ij}(t)$ is the value of the matrix representation at location (i, j) of the tth pattern to be allocated on the stock sheet.

A value in the stock sheet matrix which is greater than one is an indication of an overlap. Although the overlap test can be performed easily and quickly, there is a need for a large memory for this representation scheme. Consequently, only required matrices are generated by the algorithm while the overlap test is performed.

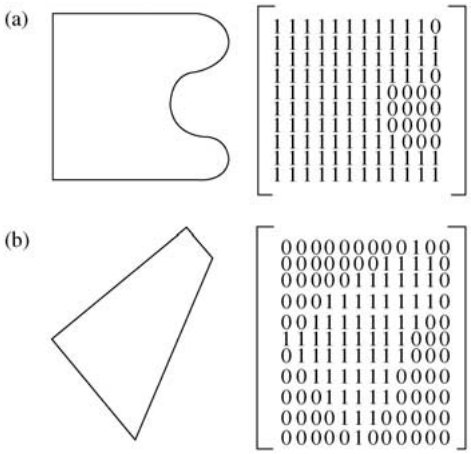

Fig. 1. Binary representations.

2.2. *The center of gravity*

In this paper, the center of gravity (area) (Cx, Cy) is used as the reference point for each pattern. It is calculated using the corresponding matrix representation. The formulas are

$$Cx = \frac{\sum_{i=1}^{L}(\sum_{j=1}^{K} Pij)^{*} i}{\sum_{i=1}^{L}\sum_{j=1}^{K} Pij}, \quad Cy = \frac{\sum_{j=1}^{K}(\sum_{i=1}^{L} Pij)^{*} j}{\sum_{i=1}^{L}\sum_{j=1}^{K} Pij}$$

2.3. *Pattern orientation*

Different orientations of a pattern may produce different scrap areas. In this paper, a rectangular enclosure, which can accommodate the whole irregular-shaped pattern, is used as an approximation of the area required to allocate the pattern on a stock sheet. A pattern with different orientations may produce different sizes of rectangular enclosures. The smallest enclosure found defines the upper bound of the area required to allocate the pattern.

In order to find the smallest rectangular enclosure of a pattern, 90 different orientations $(\theta = 0, \ldots, 89)$ of the pattern are obtained and the angle $\theta_{\min}$ that produces the smallest rectangular enclosure is selected as the orientation of the pattern. For precision purposes, rotations and rectangular enclosure computations are performed using list of vertex coordinates representation. The selected orientation of the pattern is then converted to matrix representation.

The rotated pattern is obtained as follows:

$$\begin{bmatrix} x_{k\theta} \\ y_{k\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix}$$

where (x_k, y_k) denotes the coordinates of the kth vertex of the original pattern and $(x_{k\theta}, y_{k\theta})$ denotes the coordinates of the kth vertex of the rotated pattern.

$$\mathrm{RC}_{\theta} = \left[\max|_{k=1}^{M}\{x_{k\theta}\} - \min|_{k=1}^{M}\{x_{k\theta}\}\right] * \left[\max|_{k=1}^{M}\{y_{k\theta}\} - \min|_{k=1}^{M}\{y_{k\theta}\}\right]$$

Among the 90 different rectangular enclosures

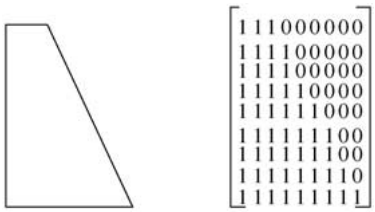

Fig. 2. Binary representation.

$(\theta = 0, \ldots, 89)$, the size of the minimum rectangular enclosure is

$$RC_{\min} = \min|_{\theta=0}^{89} RC_{\theta}$$

The pattern in Fig. 1(b) is reoriented in such a way that the minimum rectangular enclosure is produced (Fig. 2). As a result, the rectangular enclosure which defines the area required by the pattern is reduced from an 11×12 to a 9×9 unit. This orientation is used by the allocation algorithm which is described in the following sections.

2.4. *System architecture*

The system developed in this research consists of three main components: pre-processing module, detailed allocation algorithm (part 1–2), and sequence generator. The first part of the detailed allocation algorithm is performed for every sequence generated by the sequence generated. The second part is performed only after the best performance sequence is determined. The system architecture of the hybrid system is illustrated in Fig. 3.

2.5. *Pre-processing module*

A pre-processing module is required to interface with a CAD software so that the system can read the required parts and create their representation schemes used by the system. A simple heuristic is also employed at this stage. Patterns are reoriented in such a way that they require minimum area for their individual allocations (see pattern orientation section for more detail). These reoriented patterns are used later by the detailed algorithm to generate nesting patterns.

2.6. *Detailed allocation algorithm*

Pattern allocation is performed sequentially based on the order given by the sequence generator. During

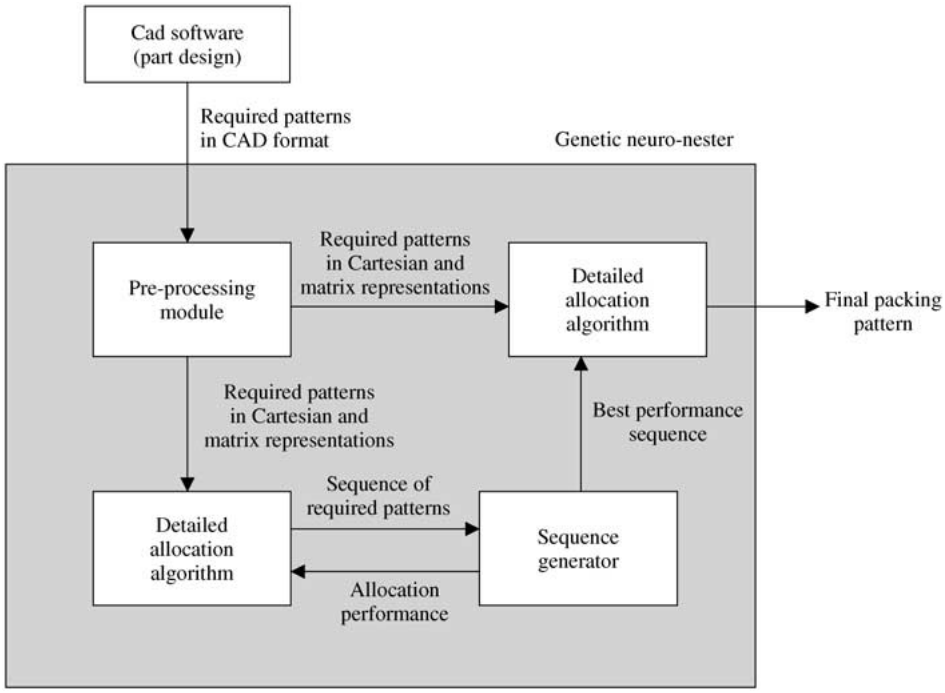

Fig. 3. System architecture.

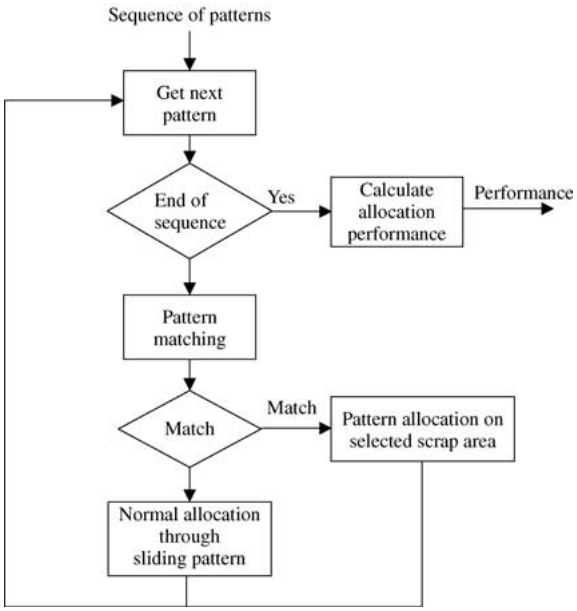

Fig. 4. Flow chart of the detailed allocation algorithm.

each pattern allocation, an incoming pattern is placed as closely as possible to the previously allocated pattern to minimize the cost function. However, many scrap areas are produced as a result of this process. One of these scrap areas might be reconsidered as an allocation area if it is large enough to accommodate the next allocation. This will result in a higher packing density value of the nesting pattern at that stage. The generated scrap areas are collected and compared with the incoming pattern to find a match. If there is a possible match, the selected area is used for the pattern allocation, otherwise, a strategy called ''sliding pattern'' is employed. The flow chart of the detailed allocation algorithm is summarized in Fig. 4. Details of the ''sliding pattern'' strategy and the scrap area selection used in the research are described extensively in the following sections.

2.6.1. *Sliding pattern*

''Sliding pattern'' is a heuristic method used to relocate the current pattern on a stock sheet without affecting any other allocated patterns. It tries to minimize the total required area, the objective value, which is the total length of material used times the width of the stock sheet. This objective value can be stated as

$$\text{Objective value of sliding pattern} = x * n + y$$

where x is the coordinate of the pattern along the X axis of the stock sheet's matrix representation; y is the coordinate of the pattern along the Y axis of the stock sheet's matrix representation; n is the width of the stock sheet $(y_{\max})$.

By using this approach, when the stock sheet is empty the first allocated pattern is placed at the lower left hand corner (the minimum value along both X and Y axes) so the objective value is minimized. The following patterns are allocated along the Y axis until there is not enough space that can be used to allocate a pattern, then, a new row of patterns along the X axis is formed. The direction of the pattern allocations is demonstrated in Fig. 5.

Although a pattern is reoriented in such a way that it requires a minimum area in its individual allocation, different orientations of the pattern may create different results once it is actually allocated on the stock sheet. Four different orientations are generated from the pattern by rotating it 0°, 90°, 180° and 270°. These four combinations require the same amount of space for an individual allocation. ''Sliding pattern'' is performed on all combinations and only the best result is selected. Based on its objective value, the candidate pattern needs to move to the left most possible on the stock sheet, which is the minimum value along the X axis. However, another objective must also be considered. The pattern must be packed along the Y axis if it is possible, hence, it is moved downward toward the origin. Each movement is terminated if one of the following conditions occur; new move produces more overlap area, or the resulting pattern can not be fit within the current stock sheet. These movements are performed for all

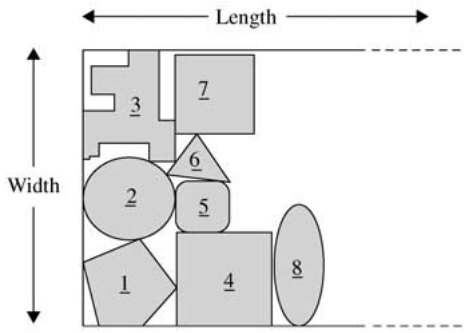

Fig. 5. Allocation sequence.

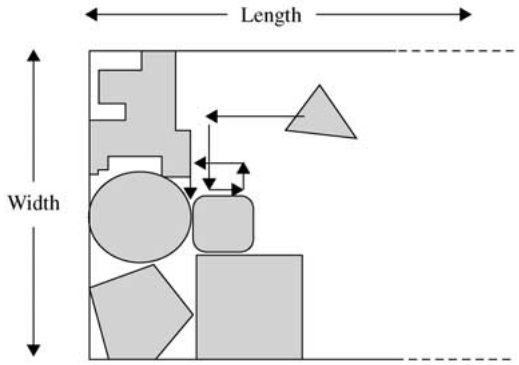

Fig. 6. Rotation in allocation.

four combinations but only the one with the minimum objective value is selected as the final solution. An example of pattern allocation by sliding pattern method is demonstrated in Fig. 6.

2.6.2. *Area selection*

In the proposed detailed allocation algorithm, scrap areas generated by previous allocations, which are the complements of the allocated patterns (Fig. 7), may be reused to increase the packing density of the nesting pattern at a particular stage. The status on the stock sheet at each allocation stage is considered as the current configuration of the problem. It carries information on both previously allocated patterns and generated scrap areas. The areas used by allocated patterns are indicated by ones in the matrix representation. The scrap areas which are the complement of the allocated patterns are indicated by zeros. These scraps areas are extracted from the stock sheet using the following steps:

(a) Scan each row of the matrix representation and create line segments from consecutive zero values.

(b) Merge overlapping line segments from consecutive rows to create irregular-shaped scrap areas. All zero values are converted into ones, therefore, the matrix representation of scrap areas are formed.

(c) Construct Cartesian coordinate representation from the minimum and maximum x values of each line segments.

Only one of the generated scrap areas can be selected as the allocation area for the input pattern at a particular stage. The selected scrap area (C_k) must satisfy the following conditions:

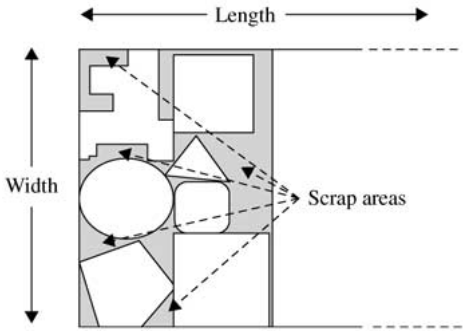

Fig. 7. Scrap areas generated.

$$C_k \in \{C_1, \ldots, C_m\}$$

$$\sum_{l=1}^{m} C_l = S - \sum_{j=1}^{i-1} P$$

$$C_k = \min\{C_j | C_j \geq P_i, j = 1, \ldots, m\}$$

where P_i is a pattern that is allocated at stage i, m is the number of scrap areas generated at stage i and S is the total area of the stock sheet.

From the equations given above, only the best (smallest possible) fitted scrap area is selected as the allocation area for the pattern (P_i). The selection process can be done directly by comparing an incoming pattern to each of these irregular-shaped scrap areas through the use of an artificial neural network.

An adaptive resonance theory (ART) paradigm, which is an extension of competitive learning, proposed by Carpenter and Grossberg (1987) is selected as the artificial neural network for the scrap selection procedure to enhance performance of the matching process. An ART network is a neural network structure that has two layers of processing namely competitive layer and input layer. These two layers are connected to each other through top-down and bottom-up connections. A feedback mechanism, controlled by a vigilance parameter, between the competitive layer and the input layer is added to be able to learn new information without destroying the old one. The information in an ART network is sent back and forth between layers until the resonance state is achieved. The time required for the network to achieve the resonance state is much less than the time used to change the weight values of the processing elements, therefore, no learning occurs before the resonance state is accomplished. There are two causes that lead to the resonance state. If the input

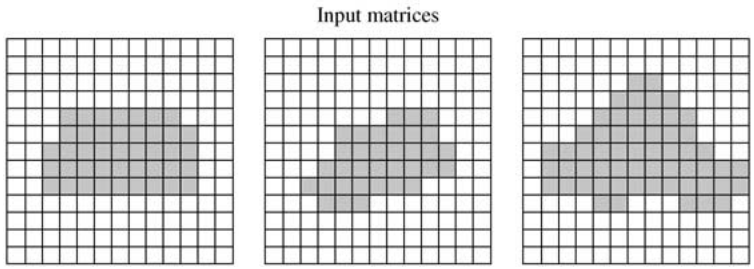

Fig. 8. Binary representation of scrap areas.

information is previously learned by the networks, the resonance state can be obtained quickly and the stored pattern is updated. On the other hand, if the input information is not immediately recognized, the rest of the stored patterns will be rapidly searched for a match. If no match is found, the network will go into the resonance state to store the input information as a new pattern.

Even though these scrap areas can be directly matched with an input pattern, different orientations of the scrap areas or the orientation of the input pattern may cause a mismatch. Before any comparison, scrap areas are transformed into Cartesian coordinates so that they can be rotated to the standard form which is the orientation that generates the minimum rectangular enclosure. Then, new matrix representations for the scrap areas in standard form are generated and used as the training patterns for the ART1 network (Fig. 8). All the scrap areas generated in the previous stage are stored in the network, ready to be compared with the incoming input pattern.

It is important to note that the input sequence of training patterns into the ART1, which is in this case, scrap areas, has an impact on the patterns stored in the network's memory. The bigger patterns will be eliminated by the smaller patterns if they are previously stored in the ART1 memory. This is not a desired characteristic; hence, ART1 is modified for this purpose. While scrap areas are trained, the pattern matching process of ART1 is omitted and every scrap area is stored in the network's memory. As a result, all scrap areas are kept in the network to compete to be the best match to the next incoming pattern.

When an input shape needs to be allocated on the stock sheet, it is presented to the ART1 network to locate a possible match among previously generated scrap areas stored in the network's memory. All stored scrap areas compete with each other to find the best match with the incoming pattern. If there is a possible match between an incoming pattern and a scrap area, a processing element which contains the best fit scrap will win the competition (Fig. 9). The winning area is then used for the allocation. However, before it is placed onto the area, the input pattern needs to be rotated by the angle that the winning scrap forms. The orientation of the scrap area was saved earlier during the scrap gathering process. If the selected scrap area is larger than the part to be allocated, sliding method is used to position the new part in the scrap area.

2.7. *Sequence generator*

The detailed allocation algorithm depends heavily on the sequence of the input shapes. Different sequences of same set of patterns may result in different packing densities. This is a permutation problem similar to the well-known traveling salesman problem. The sequence of the input shapes given to the algorithm can be mapped to all the cities traveled by the traveling salesman. The total material used is similar to the total cost of the salesman's travel. However, this traveling salesman problem cannot be solved by an old-fashioned optimization method because the cost function is non-linear. Genetic algorithm which is suitable for these types of problems is selected as a solution technique in the sequence generator module.

The sequence generator maps each possible sequence into a string (chromosome). Each element of the string is a pattern identifier. Position of the element in the string corresponds to the rank of the corresponding pattern in the input sequence. The initial population is randomly created. The fitness rating of each string is the allocation performance of the corresponding sequence by the detailed allocation algorithm. Offsprings are repeatedly generated

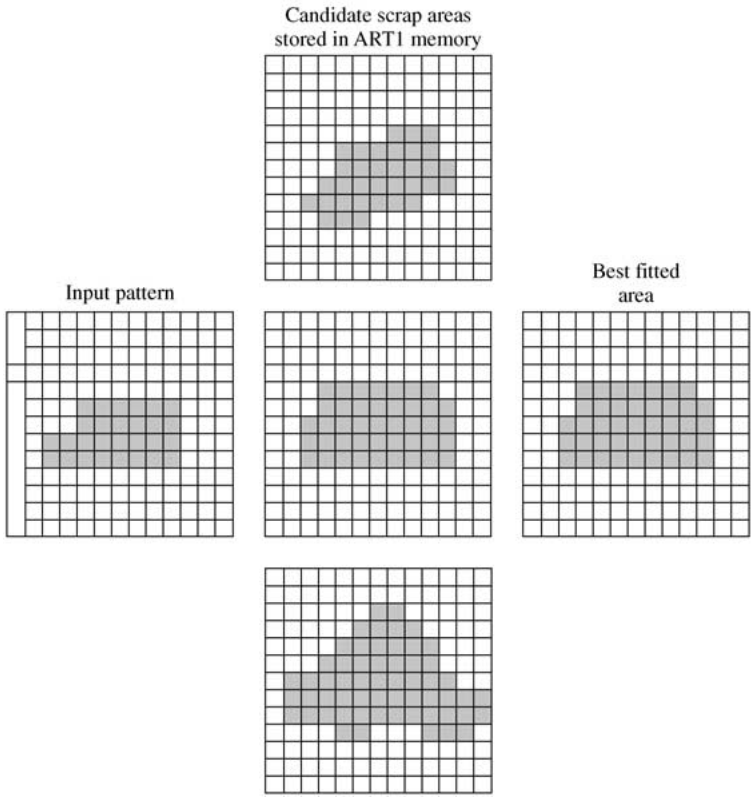

Fig. 9. Selecting best scrap area for allocation through ART1.

through genetic operators until a satisfactory result is encountered.

2.7.1. *Genetic operators*

In any permutation problem, the main concern is the order of the sequence. Different sequences of the same set of elements result in different solutions. In a permutation problem, every element of the problem must exist in each possible solution, but may be in different positions. The standard crossover operator where a random crosspoint is selected and the substring of the parents are exchanged to form the offstrings is not suitable for permutation problems. Standard crossover violates permutation constraint since after crossover every element of the parent may not exist in an offstring and some of the elements may exist more than once. The standard mutation operator, where some randomly selected genes are flipped (ones changed to zeros, zeros changed to ones) has the same problems, violates permutation constraints. There are some reordering operators that combine features of inversion and crossover into a single operator. Partially matched crossover (PMX) and order crossover (OX) stated in Oliver *et al*. (1987) attempt to preserve absolute position in ordering in their offsprings. According to Oliver *et al*. (1987) OX shows better performance, thus OX is selected as the crossover operator for our sequence generator module.

Because the OX operator does not give sufficient exploration into new search space, a mutation operator must also be considered. As the mutation operator inversion is selected. Inversion creates a new pattern string from the parent by inverting the order of a portion of the parent's string, all the elements of the parents are kept in the new string, none repeated.

2.7.2. *Fitness scaling*

The sequence generator module uses three genetic operators to generate new offsprings: reproduction, OX and inversion. A reproduction operator simply duplicates the individual strings into the next

generation based on their fitness values, which is in this case, their packing densities resulting from the detailed allocation module. Using the normal selection rule, the extraordinary individuals would take over a significant portion of the population and cause a premature convergence. To prevent a premature convergence a simple procedure called linear scaling (see Goldberg, 1989, for more details) is used.

In order to find their raw fitness values, the new individuals generated by the genetic operators are evaluated by the detailed allocation module, which performs the actual layout based on the sequence specified by each individual. Repeated strings such as the ones that are generated by the reproduction operator do not need to be re-evaluated. Their raw fitness values can be directly copied from their parents. As a result, the time required for each generation is tremendously reduced when system converges.

This section only discusses the model formulation and the solution approach to solve the stock cutting problem. The details of how to implement each module of the model is discussed in the following sections. Experiments used to test the performance of the model, the values of the parameters used, and the results obtained are also summarized.

3. Implementation of the model

All of the programs that are used to implement the model are written in C++ except the interface modules used in the communication process of the parallel system which are implemented in C because of the unavailability of the communication support routines in the available C++ compiler used.

As described earlier in the previous section the solution model consists of three major components, pre-processing module, detailed allocation module, and sequence generator. The implementation of these modules is discussed in the following paragraphs.

3.1. *Pre-processing module*

The pre-processing module reads required patterns from the AutoCAD software and converts them into Cartesian coordinates and matrix representations used by the system. AutoCAD must provide these patterns in the plotter format by plotting these individual patterns into separate *.PLT files. The software is customized to read a file created by AutoCAD software which must be configured for a laser jet printer with a resolution of 300 DPI as its plotter device. This module then reorients the pattern in question corresponding to its rectangular enclosure to provide the minimum required area. Finally, it generates both Cartesian coordinates and matrix representations according to the pattern's new orientation. The resolution of the matrix representation used in each program may vary, the system implemented in this research let users specify different resolution for different problems.

3.2. *Detailed allocation module*

This module is the most important module of the system. It reads both formats of the required parts generated by the pre-processing module and performs the actual layout based on a sequence given by the sequence generator. Required parts and dimension of stock sheet(s) used are given in configuration files. Parts are allocated sequentially by the algorithm based on their sequence given to the module using the ''sliding pattern'' heuristic along with the artificial neural network (ART1).

3.2.1. *Sliding pattern*

Sliding pattern heuristic first allocates the pattern on the far end of the parts allocated on the stock sheet. Then it tries to relocate the pattern toward the left bottom position of the current stock sheet to reduce gaps between the pattern itself and the allocated patterns on the stock sheet so the total required length can be reduced. Steps and constraints used to relocate the pattern are stated in the previous chapter. The movement is only one unit of the matrix representation on each step. Even though the time required in relocating each pattern is very high, the total computational time used does not increase when the number of parts allocated on the stock sheet increases. Once the process is finished, the Cartesian coordinates of the pattern is updated and referenced to its new locations.

3.2.2. *Area selection*

The scrap areas are extracted from the stock sheet by scanning each row of the matrix representation of the stock plate (or the active sheet in multiple plate problems). The result of scanning the sheet is a pool of line segments as sections of scrap areas. These line

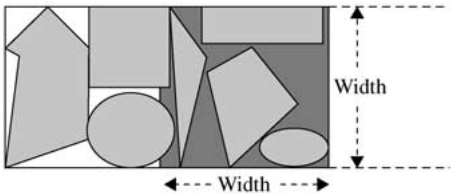

Fig. 10. Active portion of stock sheet during allocation.

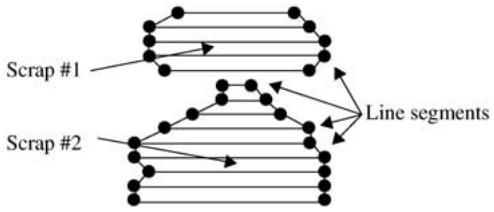

Fig. 11. Line segment representation of scrap areas.

segments are later combined and form scrap areas. However, this process consumes significant amounts of computational time, only an active portion of the stock plate at each stage is scanned. A rectangular area with the size of (the width of the stock plate) unit at the end of the stock sheet is considered as the active portion of the stock sheet at each allocation stage (Fig. 10). Furthermore, short line segments of scraps are ignored in order to reduce the scan time. Only significantly long line segments are included in the scrap segment pool so the scrap areas that are smaller than the required parts are eliminated at this stage.

Line segments are grouped together and form a scrap area based on their relative positions on consecutive rows (whether the corresponding intervals overlap or not). If the amount of overlap is less than half of the shorter segment's length, the segments should be considered as different scrap areas. Consequently, two scrap areas with a small connected portion are separated from each other (Fig. 11).

After the scrap areas are formed, the algorithm eliminates the areas that are smaller than the current input pattern from the pool so the number of the candidates is reduced. For the remaining scrap areas Cartesian coordinates (vertex coordinate lists) are obtained from the list of line segments, these areas are then reoriented to the standard form where the selected orientation produces the smallest rectangular enclosure, new matrix representations are generated from reoriented scrap areas. Due to the fixed size of the input layer of the ART1 network, the variable sized scrap areas cannot be directly trained into the ART1 memory. These areas must be transformed into matrices of the same size that must be sufficiently large to accommodate major area of a scrap, yet not too big concerning the memory capacity and the time complexity of the selection process. The center of each area is used as a referenced point to the center of the standard matrix for its transformation. Finally, the transformed matrix can be used to train the network (Fig. 12).

During the testing process, same transformations should be applied to the incoming pattern. Before it is inputted to the ART network, incoming pattern should be reoriented to form the smallest rectangular enclosure and its standard size matrix representation should be generated. During the testing process for each incoming pattern the best fit scrap pattern stored in the ART1 memory is searched. The vigilance parameter used in the matching process is set to one. The reason for using high vigilance values in this application is the need of a perfect match between the pattern and a scrap area. The selected scrap area must be able to accommodate the incoming pattern without creating any overlaps on the stock sheet. If there is no possible match between any scrap areas with the

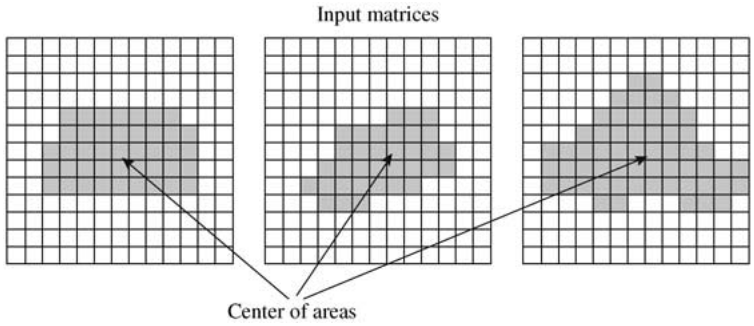

Fig. 12. Determination of scrap area centers.

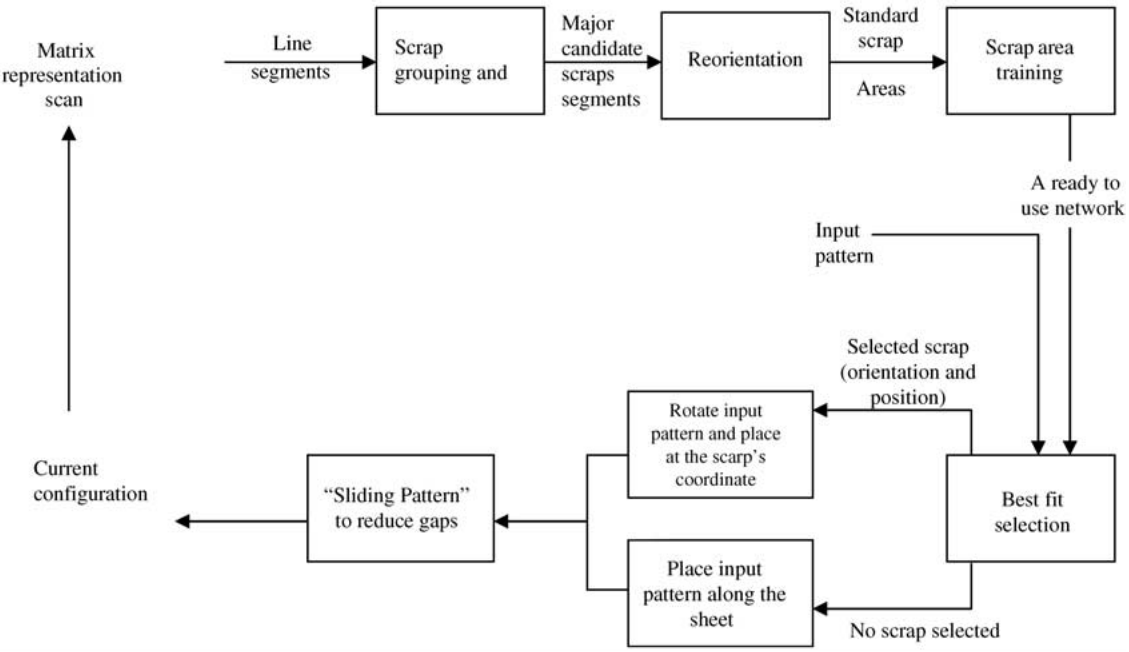

Fig. 13. The control flow of the detailed allocation module.

current pattern the ''sliding pattern'' is employed. Finally, the current configuration of the stock sheet is updated. This process is repeated for each of the required patterns based on the sequence given by the sequence generator. The system flow of the detailed allocation module is summarized in Fig. 13.

3.3. *Sequence generator*

This module uses genetic algorithms as a technique to improve performance of the system by generating new sequences to allow the detailed allocation module to perform a better layout. The number of individuals used in each problem is varied based on the size of the problem that is provided in an input configuration file. The first individual (sequence) of the first population is generated by a heuristic, the rest of the individuals are randomly generated. Elements of this first sequence are sorted in descendent order of pattern's area. This sorted order groups similar sized parts together. It also has smaller parts towards the end of the sequence which is important because those smaller parts that are allocated later may fit into gaps generated by previously allocated larger parts. This sequence may improve the performance of the system once it is crossed over with the other sequences in order to generate new offsprings.

As stated earlier in the model description, the selected genetic operators used in this research are reproduction, OX and inversion. The reproduction operator is applied to the old population to create their offsprings. The best individual within the current population is automatically carried to the next population, so the best solution found is kept within the population at all times. Crossover and mutation rate used by the OX and inversion operator are specified as input parameters to the system. The rates of 0.6–0.8 and 0.001–0.020 are used for the OX and inversion respectively to test the performance of the solution system.

4. Experimental results

Different problem sets are tested to evaluate the performance of the proposed system. Both rectangular and irregular-shaped parts are used in the test runs. Tests are performed for different crossover rates (0.6–0.8) and different mutation rates (0.001–0.020). Population sizes are set up to three times of the number of required parts.

4.1. *Rectangular problems*

The system is tested with 37 rectangular parts created from eight different pattern types. Their dimensions and corresponding demands are randomly generated using normal distributions (Table 1). These parts are used in two different problems. In the first problem, they are allocated on a stock sheet with a finite width of 30 inches but infinite in length. In the second problem, the same required parts are tested with stock sheets that are finite in both width and length with dimensions of 30×60 inch^2. A fixed population size

Table 1. Dimensions and demands corresponding to eight different rectangular patterns

Pattern	*Dimension (Width)*	*Demand*
1	12×10	3
2	8×11	4
3	9×13	6
4	4×5	4
5	9×10	3
6	6×8	6
7	10×12	9
8	15×7	2

Fig. 14. The best final packing pattern of 37 rectangular parts on a continuaous stock sheet.

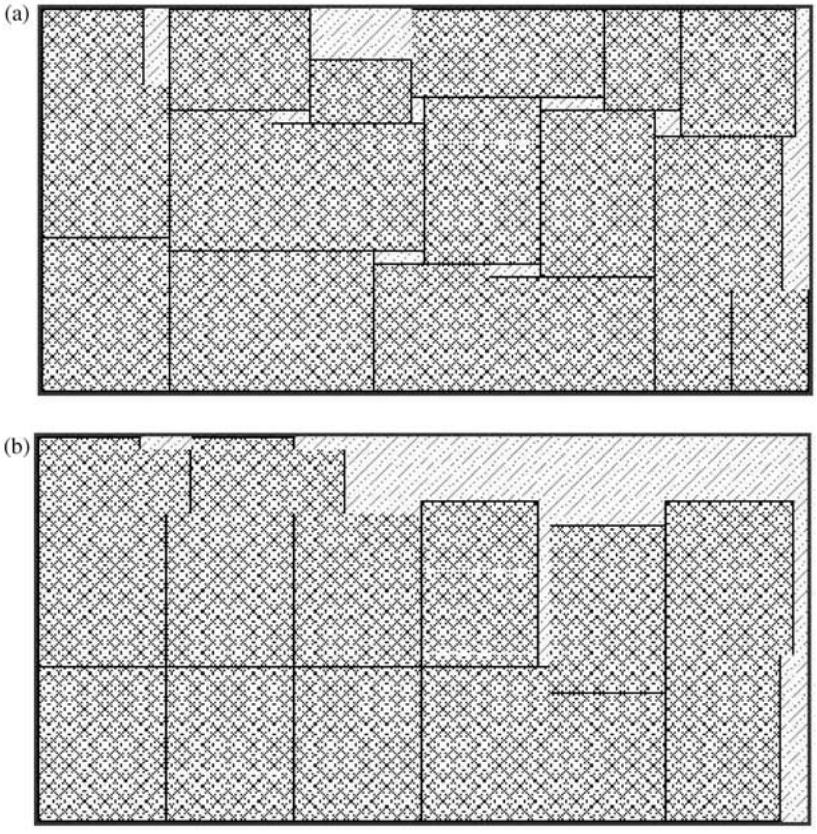

Fig. 15. The best packing pattern of 37 rectangular parts on finite size stock sheets.

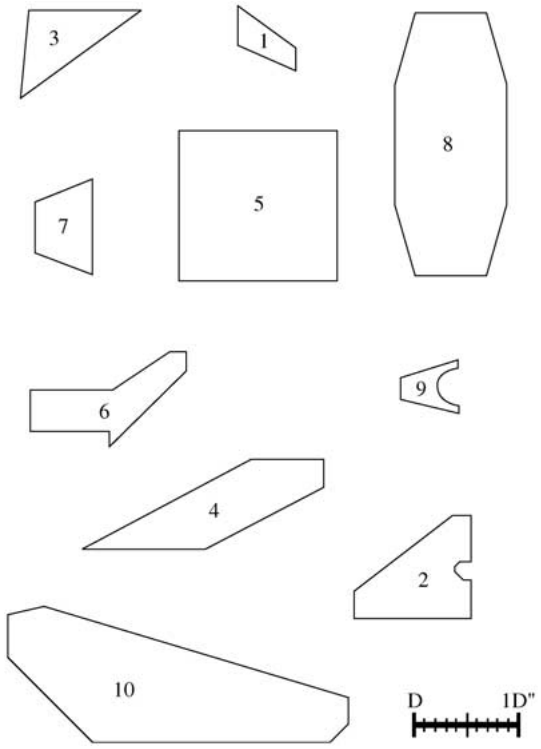

Fig. 16. Stock patterns used.

of 91 with different crossover and mutation rates are used in both problems. Five hundred generations are produced for each test run. Various results for these problems are obtained through different sets of parameters as shown in Tables 2 and 3, respectively. The best final packing patterns found for both problems are demonstrated in Figs. 14 and 15(a), (b), respectively.

4.2. *Irregular-shaped problems*

Ten different irregular-shaped patterns are provided by an outside company (see Fig. 16). Thirty required parts, three parts from each pattern, are first tested on the continuous stock sheet having a width of 60 inches to see the grouping characteristics among these parts. Different parameter sets are used to run these tests. Results obtained are shown in Table 4, and the best final packing pattern with the packing density of 80.61% is shown in Fig. 17.

Table 2. Results of allocating 37 rectangular parts on an infinite length with finite width of 30 inches stock sheet (packing density (%))

Crossover rates	*Mutation rate*		
	0.001	0.01	0.02
0.6	91.31	93.61	92.83
0.7	93.61	92.83	96.03*
0.8	92.06	93.61	95.21

*The best solution found.

Table 3. Results of allocating 37 rectangular parts on two 30×60 inch^2 stock sheet (packing density (%))

Crossover rates	*Mutation rate*		
	0.001	0.01	0.02
0.6	87.72	87.72	87.72
0.7	87.72	93.61	87.72
0.8	87.72	88.41	94.41*

*The best solution found.

Table 4. Results of allocating 30 irregular-shaped parts on an infinite length with finite width of 60 inches stock sheet (packing density (%))

Crossover rates	*Mutation rate*		
	0.001	0.01	0.02
0.6	80.54	84.35	81.61*
0.7	80.54	79.78	79.36
0.8	80.52	77.09	80.56

*The best solution found.

5. Conclusion

Automated irregular-shaped pattern cutting design is becoming a requirement in manufacturing due to large product variety, low product life, and high demands for manufacturing flexibility. This research work introduced recently emerging techniques to continue to meet these objectives. Many solution approaches to various stock cutting problems are reviewed and commented on, but as yet, there is no single powerful algorithms that can solve this problem effectively. New technologies in the early 1990s such as artificial neural networks and genetic algorithms are investigated and found to be useful tools. A new approach

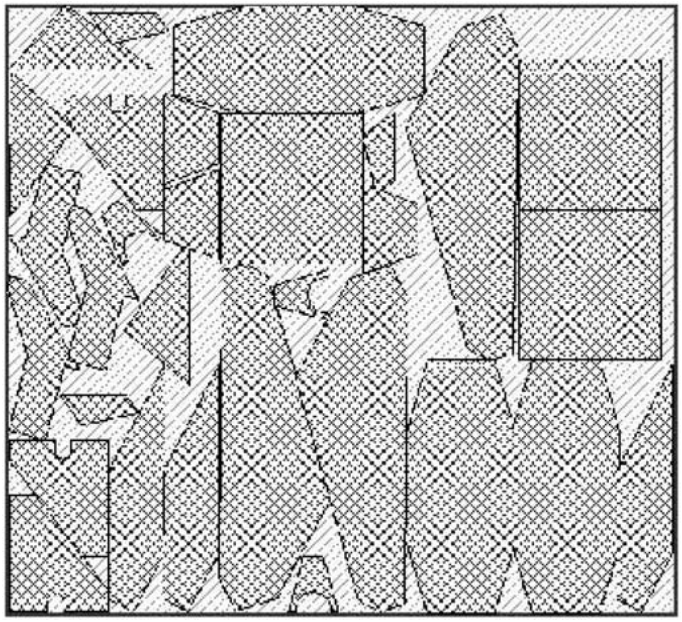

Fig. 17. Final allocation of patterns.

that combines the use of both techniques is proposed as a solution for both rectangular-shaped and irregular-shaped cutting stock problems.

Parts that are designed using a design tool such as a CAD system are able to be retrieved by the proposed automated system. The proposed system consists of three main components: a pre-processing module, a detailed allocation algorithm and a sequence generator. The pre-processing module functions as an interface between the system and a CAD software. The detailed allocation algorithm performs the part layout based on the sequences of parts given by the sequence generator. It sequentially allocates each part in such a way that the resulting scrap is minimized. An artificial neural network paradigm, ART1, is selected as a tool for the detailed allocation algorithm. The sequence generator, on the other hand, deals with another optimization problem. The objective of this module is to find an optimum part sequence to be used by the detailed allocation algorithm in which the minimum scrap is generated. Genetic algorithms are chosen as a solution technique to this problem.

The performance of the solution approach is tested using several nesting problems using varying parameter sets. Different rectangular and irregular patterns were used in both one-and-a-half and two dimensional problems and the results obtained are satisfactory.

References

Adamowicz, M. and Albano, A. (1976) A solution of the rectangular cutting-stock problem. *IEEE Transactions on Systems, Man, and Cybernatics*, **6**(4), 302–310.

Adamowicz, M. and Albano, A. (1976) Nesting two-dimensional shapes in rectangular modules. *Computer Aided Design*, **8**, 27–33.

Albano, A. (1977) A method to improve two-dimensional layout. *Computer Aided Design*, **9**, 48–52.

Albano, A. and Orsini, R. (1980) A heuristic solution of the rectangular cutting stock problem. *The Computer Journal*, **23**(4), 338–343.

Albano, A. and Sapuppo, G. (1980) Optimal allocation of two dimensional irregular shapes using heuristic search methods. *IEEE Transactions on Systems, Man, and Cybernatics*, **10**, 242–248.

Beasley, J. E. (1985) An algorithm for the two-dimensional assortment problem. *European Journal of Operation Research*, **19**(2), 253–261.

Bengtsson, B. E. (1982) Packing rectangular pieces—a heuristic approach. *The Computer Journal*, **25**(3), 353–357.

Carpenter, G. A. and Grossberg, S. (1987) A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, **37**, 54–115.

Christofides, N. and Whitlock, C. (1977) An algorithm for two-dimensional cutting problems. *Operations Research*, **25**(1), 30–44.

Chung, J., Scott, D. and Hillman, D. J. (1990) An intelligent nesting system on 2-D highly irregular resources. *Proceedings of Application of Artificial Intelligence VIII*, pp. 472–483.

Dagli, C. H. (1988) Cutting stock problem: combined use of heuristics and optimization methods, in *Recent Developments in Production Research*, Mital, A. (ed), pp. 500–506.

Dagli, C. H. (1988) Integrating AI and OR in solving cutting stock problem, in *Robotics and Manufacturing: Recent Trends in Research, Education and Applications*, Jamsidi, M., Luh, J. Y. S., Seraji, H. and Starr, G. P. (eds), pp. 911–918.

Dagli, C. H. (1990) Neural networks in manufacturing: possible impacts on cutting stock problems. *Second International Conference on Computer Integrated Manufacturing*, pp. 531–537.

Dagli, C. H. (1994) Intelligent manufacturing systems. *Artificial Neural Networks for Intelligent Manufacturing*. Chapman & Hall, London, pp. 3–16.

Dagli, C. H., Ashouri, M. R., Leininger, G. and McMillin, B. (1990) Composite stock cutting pattern classification through neocognitron. *International Joint Conference on Neural Networks*, **2**, pp. II-587–590.

Dagli, C. H. and Hajakbari, A. (1990) Simulated annealing approach for solving stock cutting problem. *Proceedings of IEEE International Conference on Systems, Man, and Cybernatics*, pp. 221–223.

Dagli, C. H. and Nisanci, I. H. (1981) A heuristic for cutting-stock problem. *Sixth International Conference on Production Research*, Vol. 1, pp. 9–16.

Dagli, C. H. and Tatoglu, M. Y. (1983) A computer package for solving cutting stock problems. *Seventh International Conference on Production Research*, Vol. 1, pp. 175–190.

Dagli, C. H. and Tatoglu, M. Y. (1987) An approach to two-dimensional cutting stock problems. *International Journal of Production Research*, **25**(2), 175–190.

Daniels, J. J. and Ghandforoush P. (1990) An improved algorithm for the non-guillotine cutting-stock problem. *Journal of Operational Research Society*, **41**(2), 141–149.

Dietrich, R. D. and Yakowitz, S. J. (1991) A rule-based approach to the trim-loss problem. *International Journal of Production Research*, **29**(2), 401–415.

Dowsland, K. A. (1992) Comments on an algorithm for the cutting stock problem. *Journal of Operational Research Society*, **43**(12), 1189–1190.

Duta, L. D. and Fabian, C. (1984) Solving cutting-stock problems through the Monte-Carlo method. *Economic Computation and Economic Cybernetics Studies and Research*, 35–54.

Dyckhoff, H. (1989) Approaches to cutting and packing problems. *Tenth International Conference on Production Research*, August.

Eisemann, K. (1957) the trim problem. *Management Science*, **3**(3), 279–284.

Farley, A. A. (1988) Mathematical programming models for cutting-stock problems in the clothing industry. *Journal of Operational Research Society*, **39**(1), 41–53.

Farley, A. A. (1990) The cutting stock problem in the canvas industry. *European Journal of Operational Reaearch*, **44**, 247–255.

Ferreira, J. S., Neves, M. A. and Fonseca e Castro, P. (1990) A two-phase roll cutting problem. *European Journal of Operational Research*, **44**, 185–196.

Foronda, S. U. and Carino, H. F. (1991) A heuristic approach to the lumber allocation problem in hardwood dimension and furniture manufacturing. *European Journal of Operational Research*, **54**(2), 151–162.

Garey, M. R. and Johnson, D. S. (1979) *Computers and Interactability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco.

Gemmill, D. D. (1992) Solution to the assortment problem via the genetic algorithm. *Mathematical and Computer Modeling*, **16**(1), 89–94.

Gemmill, D. D. and Sanders, J. L. (1991) A comparison of solution methods for the assortment problem. *International Journal of Production Research*, **29**(2), 2521–2527.

George, J. A. (1992) A correction of the improved bounds for the non-guillotine constrained cutting stock problem. *Journal of Operational Research Society*, **43**(12), 1187–1189.

Ghandforoush, P. and Daniels, J. J. (1992) A heuristic algorithm for the guillotine constrained cutting stock problem. *ORSA Journal on Computing*, **4**(3), 351–356.

Gilmore, P. C. and Gomory, R. E. (1961) A linear programming approach to the cutting stock problem. *Operations Research*, **9**(6), 849–859.

Gilmore, P. C. and Gomory, R. E. (1965) Multistage cutting stock problems of two and more dimensions. *Operations Research*, **13**(1), 94–120.

Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison Wesley, MA.

Golden, B. L. (1976) Approaches to cutting stock problem. *AIIE Transactions*, **8**(2), 256–274.

Goulimis, C. (1990) Optimal solutions for the cutting stock problem. *European Journal of Operational Research*, **44**, 197–208.

Haessler, R. W. (1980) Multimachine roll trim: problem and solutions. *TAPPI*, **63**(1), 71–74.

Haessler, R. W. and Sweeney, P. E. (1991) Cutting stock

problems and solution procedures. *European Journal of Operational Research*, **54**, 141–150.

Hahn, S. G. (1986) On the optimum cutting of defective sheets. *Operations Research*, **16**(6), 1100–1114.

Haims, M. J. and Freeman, H. (1970) A multistage solution of the template-layout problem. *IEEE Transactions on Systems Science and Cybernetics*, **6**(2), 145–151.

Harris, R. and Zinober, A. (1988) Practical microcomputer solutions of the cutting stock problem. *IEE Colloquium on Artificial Intelligence in Planning for Production Control*, **8**(1–4).

Hinxman, A. I. (1973) Two-dimensional trim-loss problem with sequencing constraints. *International Joint Conference on Artificial Intelligence*, pp. 859–864.

Hinxman, A. I. (1980) The trim loss and assortment problems: a survey. *European Journal of Operational Research*, **5**, 8–18.

Kampke, T. (1988) Simulated annealing: use of a new tool in bin packing. *Annals of Operations Research*, **16**, 327–332.

Kantorovich, L. V. (1960) Mathematical methods of organising and planning production. *Management Science*, **6**, 366–422.

Kirkpatrick, S., Gellatt, C. D. Jr. and Vecchi, M. P. (1982) Optimization by simulated annealing. IBM Research Report, RC 9355.

Lirov, Y. (1992) Knowledge based approach to the cutting stock problem. *Mathematical and Computer Modeling*, **16**(1), 107–125.

Lutfiyya, H., McMillin, B., Poshyanonda, P. and Dagli, C. (1992) Composite stock cutting through simulated annealing. *Mathematical and Computer Modeling*, **16**(1), 57–74.

Metzger, R. (1958) Stock slitting. *Elementary Mathematical Programming*, Wiley, New York.

Oliveira, J. F. and Ferreira, J. S. (1990) An improved version of Wang's algorithm for two-dimensional cutting problems. *European Journal of Operational Research*, **44**, 256–266.

Oliver, I. M., Smith, D. J. and Holland J. R. C. (1987) A study of permutation crossover operators on the traveling salesman problem. *Proceeding of the Second International Conference on Genetic Algorithms*, pp. 224–230.

Paull, A. E. (1956) Linear programming: a key to optimum newsprint production. *Pulp Paper Mag. Can.*, **57**, 145–150.

Pierce, J. F. (1966) On the solution of integer cutting stock problems by combinatorial programming, Part I. IBM Technical Report, 36.Y02, Cambridge Scientific Center, Cambridge, MA.

Poshyanonda, P., Bahrami, A. and Dagli, C. (1992) Two dimensional nesting problem: artificial neural network and optimization approach. *International Joint Conference of Neural Networks*, pp. IV-572–IV-577.

Poshyanonda, P., Bahrami, A. and Dagli, C. (1992) Artificial neural networks in stock cutting problem. *Neural Networks in Manufacturing and Robotics*, ASME Press, New York, p. 143.

Poshyanonda, P. and Dagli, C. (1992) A hybrid approach to composite stock cutting: artificial neural networks and genetic algorithms, in *Robotics and Manufacturing: Recent Trends in Research Education and Applications*, Vol. 4, Jamshidi, M., Lumia, R., Mulling, J. and Shahinpoor, M. (eds), ASME Press, New York, p. 775.

Qu, W. and Sanders, J. L. (1989) Sequence selection of stock sheets in two-dimensional layout problems. *International Journal of Production Research*, **27**(9), 1553–1571.

Richter, K. (1992) Solving sequential interval cutting problems via dynamic programming. *European Journal of Operational Research*, **57**, 332–338.

Schollmeyer, M. and Lin, J. Q. *et al*. (1991) Hybrid expert system and operations research for solving nesting problems. *The World Congress on Expert Systems Proceedings*, Vol. 2, pp. 1223–1231.

Vajda, S. (1958) Trim loss reduction. *Readings in Linear Programming*, Wiley, New York.

Viswanathan, K. V. and Bagchi, A. (1993) Best-first search methods for constrainted two-dimensional cutting stock problems. *Operations Research*, **41**(4), 768–776.

Viswanathan, K. V. and Bagchi, A. (1988) An exact best-first search procedure for the constrainted rectangular Guillotine Knapsack problem. *7th National Conference on Artificial Intelligence*, pp. 145–149.

Wang, P. Y. (1983) Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*, **31**(3), 573–586.