GENETIC ALGORITHMS FOR NESTING OF RECTANGULAR SHAPES

Thesis submitted to

Thapar Institute of Engineering and Technology

in partial fulfillment of the requirement
for the award of the degree of

Master of Engineering
(Computer Science)

Supervised By
Dr. G.K. Sharma
Dr. B.B. Aggarwal

Submitted By
Manju Mehta

Department of Computer Science & Engineering
Thapar Institute of Engineering & Technology
(Deemed University)
Patiala-147 001

CERTIFICATE

This is to certify that the thesis entitled *"Genetic Algorithms for Nesting of Rectangular Shapes"* submitted by Ms. Manju Mehta in partial fulfillment of the requirement for the award of the degree of **Master of Engineering in Computer Science** in the **Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology (Deemed University), Patiala,** is a record of candidate's own work carried by her under our supervision and guidance.

Place : Patiala
Date : 6-1-1998.

(Mr. G. K. Sharma)
Professor and Head,
Department of Computer Sc. & Engg.,
Deptt. of Computing Services,
T.I.E.T, Patiala-147 001.

B. B. Aggarwal

(Dr. B. B. Aggarwal)
Head,
Department of Computing Services,
TCRDC, Patiala.

Head,
Deptt. of Computer Science & Engg.,
T.I.E.T., Patiala.

6/1/98

Dean, Academic Affairs
T.I.E.T., Patiala.

ACKNOWLEDGMENTS

At the outset I express my deep sense of gratitude to Mr. G. K. Sharma, Head Department of Computer Science & Engineering, T.I.E.T, whose vision, guidance and expertise motivated me to venture into the area of Genetic Algorithms.

I am thankful to Dr. Bharat B. Aggarwal whose motivation, patience and insight cleared my way to learn new concepts.

I also extend my thanks to Mr. Parveen Gupta for this untiring efforts in the preparation of this manuscript.

I also take this opportunity to express my thanks to my colleagues in DCS and CAED for their cooperation and timely help. My special thanks are due to my family and my friends for their moral support and understanding that helped me to work for long hours on weekends.

Manju
(Manju Mehta)

TABLE OF CONTENTS

Chapter 1

Introduction

The problem of placement of a set of 2-D shapes on a larger resource while producing a minimal waste is a common resource utilization problem. For example, in sheet metal industry, 2-D patterns are placed onto sheets with finite dimensions. In this case, stock sheets on which the pieces are placed are depleted resources and the material remaining after placement of pieces (scrap) can not be used for further allocation of pieces. This problem is encountered in every area of industry from design to distribution and sales through various aspects of manufacturing process. Due to high cost, any wastage of material is intolerable during manufacturing. Since the cost of high volume stamped cutout parts is largely dependent on material usage; scrap minimization is a primary goal of design process. Although many factors influence the scrap rate, one of the most critical is the positioning or layout of parts on the stock sheet. In other words, the problem of nesting is to select the optimum arrangement for a combination of 2-D regular/irregular shapes onto the large stock sheet so as to minimize material waste while taking into consideration the various practical constraints. The constraints of this problem vary with the application. These may be the raw material used, design requirements, number of shapes required, manufacturing constraints etc.

Nesting is a problem without having clear mathematical descriptors for the process where infinite solutions exist for most of the problems. Time needed to find the cutting pattern with the least amount of stock sheet wastage increases exponentially with the number of different shapes used in the problem. Another problem is that of determining an efficient arrangement of the pieces in a containing region without overlap. However, different industries work with different practical constraints and each industry has developed their own approach without input from other industries. This has also introduced the difference in terminology. Although the solution approach may be different in various applications, the underlying objective is same in all applications.

1.1 Application Areas

Traditionally, nesting problem has been solved by an expert manually in different industries. Human intelligence is well able to tackle such problems and many studies comparing the manual results with computer generated layouts show the automated layouts to be inferior in terms of trim loss. But as computers has become integrated part of the manufacturing process, a system that can take a computer generated design, produce the cutting patterns and then control the actual cutting process is a more desirable option. The nesting problem solutions are most sought after by industries where the raw materials form a large proportion of the finished product.

The most common application area is ship building. Here, the problem involves determining a cutting pattern for a given set of parts, each of which consists of a number of different pieces, from a series of metal stock sheets. The pieces from a single part must be kept together. As more than one sheet is required , the way in which the different parts are combined together has a significant effect on the efficiency of the overall cutting. Gaps or bridges are introduced between pieces to avoid the problems during cutting.

The layout problem in garment industry is totally different. Here there is no need of bridges but the grain and pattern of the fabric puts constraints on the orientation of the fabric. The pieces for a given layout will be for several copies of the same garment and will include some very large pieces corresponding to the larger sections of the garment and some very small pieces corresponding to the trim. The stock sheet is also longer and thinner than the one in sheet metal industry.

Nesting problem is applicable in all manufacturing industries where stamping of metal components is done. Here, the shapes are complex and large but the orientation may not be a restriction. In lumber and furniture industry it can be applied to cutting of boards. Here also the pattern of the wood is important. The problem varies for leather industry where pieces has to be cut from irregular hides for various kinds of objects such as furniture upholstery, footwear and other small goods. The leather hides are of irregular quality and bad patches in the hide can be considered as holes.

In spite of all the differences in the nature of nesting problem, all the applications has a common requirement of finding a feasible layout of the pieces on the stock sheet. The number of feasible positions and orientation may differ for each problem, the technique used for one application can be applied to another but the computational speed and quality may differ from one application to another.

1.2 Nesting of Rectangular Shapes

Nesting problem can be classified with respect to shape of patterns used in the problem. Nesting of rectangular shapes deals with the problem of 2-D shape nesting. Since 1950, this problem has received considerable attention as more powerful computers have become available and the optimization techniques have developed.

Gilmore and Gomory (1961, 1963) have solved rectangular cutting stock problem by using integer programming. They have expressed it as knapsack problem, in which the large number of variables involved generally makes the computation infeasible. The same problem occurs when approximate solutions are sought by linear programming.

Chrisofides and Whitlock (1977) have proposed a tree search algorithm for the solution of rectangular nesting problem. here, a dynamic programming procedure is used to produce upper bounds to limit the size of search.

Adamowicz and Albano (1976) also deal with this problem. In their algorithm, rectangular strips are generated by the patterns having one common dimension at least : then a subset of these strips is selected to lay out on the part of the stock sheet currently under examination. Another algorithm by Albano (1977) offers interactive methods to improve the 2-D layout.

Attempts have also been made to solve the nesting problem with the help of heuristics, rule based and expert systems based approaches. However, much work remains in the direction of the development of new methods for solving nesting problems of rectangular shapes. This thesis focuses on development of a new method based on Genetic Algorithms for solving the rectangular shapes nesting problem.

1.3 Outline

This thesis is divided into six chapters including this introductory chapter. The remaining chapters are outlined below :

- The second chapter gives an overview of various nesting solution techniques for regular and irregular shapes. These techniques are based on heuristic approach, rule based approach, combinatorial methods and simulated annealing.
- The third chapter discusses the theory behind working of Genetic Algorithms (GAs). Two methods for nesting solutions are also described briefly. First one discusses a hybrid approach using GAs and heuristics. The other one describes a Layout Determination Approach (LDA) with GAs. GAs are used to determine the order of the placement of shapes on sheet. LDA decides where the shapes will be placed on sheet in the order specified by GAs.
- The fourth chapter discusses the proposed approach in detail. This approach has been implemented and tried out for rectangular shapes. This algorithms is discussed in detail along with the results obtained.
- The fifth chapter gives the details behind implementation of GA based approach. The implementation has been done using object oriented techniques. The class structure of the software modules is explained in this chapter.
- The sixth chapter concludes the work with a discussion of possible enhancements to the algorithm used in this work.

Chapter 2

Automation of 2-D Nesting Problem

Research into the automation of 2-D nesting problem can be subdivided into two broad categories. The first one involves allocation of rectangular shapes onto a rectangular sheet. This problem has been studied extensively and numerous analytical and heuristic solution techniques exist. The second category addresses the nesting of irregular profiles onto resources of arbitrary shapes. This problem can be subcategorized into three basic groupings as given below:

1. Rectangular approximation methods simplify the complexity of irregular profiles by approximating the individual shapes with rectangular enclosures. With all pieces resolved to minimum enclosure, an algorithm for rectangular nesting is easily applied. In this approach, the amount of waste material depends upon the accuracy of rectangular enclosure.

2. The second approach makes use of probabilistic optimization techniques for minimizing waste. An objective function dependent upon part overlap and overall layout dimensions is constructed. To find the associated global minimum, either a multistart, genetic algorithm or simulated annealing optimization method is used. The scale of associated problem is directly related to number of shapes involved. Although computationally expensive, this technique have the advantage of finding a true minimum over the range of design variables. Unfortunately, the solution time associated with calculating overlaps between non convex objects restricts practical use of this problem.

3. The third category encompasses a broad set of techniques based on rule based, intelligent and expert or heuristic methods. The common feature in all of these methods, is an attempt to mimic the methods used in manual nesting. This generally consists of building final solutions by placing parts onto the resources one at a time. Methods of this type are capable of producing results similar to optimization techniques. However exhaustive placement and search routines would be required.

The following sections describe some of the popular approaches for nesting of regular and irregular patterns :

2.1 Heuristic Methods

Dagli and Tatoglu [9] has solved the two dimensional cutting stock problem using a heuristic approach. They use a construction type modeling approach in which the solution is achieved by locating the patterns sequentially. It finds only one solution under the guidance of priorities assigned to patterns and

other practical considerations. Manual modification can be done after the solution gives a proposed layout of the patterns.

This method sequentially places patterns on the sheet based on a set of priority rules. These priority rules are dependent on the application. All the patterns are assigned a priority value according to the adopted priority rule. Two patterns having the highest priority are selected from the candidate set of patterns and a new pattern is defined by unification of these two patterns. The relative location of these patterns are determined by pair wise matching of their sides. The best relative location is one which gives the minimum rectangular enclosure. Another pattern is chosen from input set and new unified pattern is found by following the above procedure. This process continues until no more pattern is left.

This technique gives the flexibility to rotate the sheet to move the unified pattern without extra computation efforts because starting point of allocation is not fixed. This approach gives satisfactory results for allocation of irregular patterns but is not adequate for rectangular patterns.

2.2 Combinatoric Methods

P.Y. Wang [10] has devised combinatoric methods that generate constrained cutting patterns by making successive horizontal and vertical builds of ordered rectangles. The addressed problem is related to glass and lumber industry so it considers only rectangular cutting stocks. The solution assumes that the acceptable cutting patterns are guillotine type which means the rectangles are to be obtained by successive edge-to-edge cuts of the stock sheet.

The combinationic algorithm finds guillotine cutting patterns by combining every two rectangles from the input set $R_1, R_2, \ldots, R_n$. The rectangles can be joined in horizontal or vertical build by placing them with other rectangles along the width or length of the stock sheet. The algorithm adds the resulting rectangles (combined) to each of the original rectangles R_i to form a set of larger guillotine rectangle. The process is repeated so that each successive horizontal or vertical build forms a larger guillotine rectangle from two smaller ones. The rectangles which do not follow the practical constraints are eliminated. This approach reduces the number of combinations.

2.3 Rule Based Approaches

Dietrich and Yakowitz [11] presented a heuristic algorithm for solution of the rectangular trim loss problem using rule based approach. In this case also, the input pieces are restricted to be rectangular in shape. This algorithm uses a problem reduction approach where the original problem is reduced to a new problem of the same kind but with a smaller domain.

According to this approach, if an input pattern has the same dimensions as of stock sheet, the unused region of the sheet will be reduced completely after placing it. If one side of input pattern matches stock sheet, it will cleanly reduce stock sheet. In other cases, the input pattern will divide the stock sheet into

two holes. Then each of the holes is sought as an individual stock sheet. So, there are essentially three ways in which a price can fit into a hole.

At any time, there will a set of holes and a set of pieces to be selected from to fill the hole. Rules are developed to select a hole and a piece from the prepared set to be placed on stock sheet. Dietrich and Yakowitz has specified a set of length rules and area rules for testing. The length rules prefer longer holes and longer pieces as compared to shorter holes and shorter pieces. Area rules prefer larger holes and larger pieces as compared to smaller ones in the testing results, the area rules were the ones with the best performance.

2.4 Expert Systems

Yuzu, Lujun, Wei and Jianwen [12] have given an expert system for automatic allocation of 2-D irregular shapes. The proposed system utilizes the heuristic knowledge about allocation that is based on the experience derived from clothing industry. This allocation problem in clothing industry is considered very difficult because the number of pieces is very large and the pieces are irregular.

In this algorithm, the hierarchical method of allocating the large pieces first and filling the small pieces in complementary area is adopted. Various rules guide the allocation of pieces. One of the rules select a parameter size CA. All the input pieces are grouped in two groups according to their relative sizes to CA. This parameter is different for different types of clothes. The rule base consists different set of rules for two groups.

The rules in the rule base are designed based on the experience of involved people but when the computer imitates the allocation, it needs a evaluation criteria to decide whether the adopted scheme is acceptable or not. Because the modifications are based on the scheme adopted so each scheme in the rule base should have some information about possible modifications. This step is known as feedback and it completes a rule based system.

2.5 Optimum Nesting Using Simulated Annealing

This section discusses a nesting approach based on simulated annealing [13]. It describes an automated system that optimizes blank nesting for continuous strip stamping process. Blank nesting is often done by hand, resulting in inefficient nesting with high scrap rate. This problem only addresses nesting of three or fewer blanks since typical sheet metal dies stamp only one or two parts at a time. The shape of the blank is described by an ordered set of points on the boundary of blank, which are connected together by straight line segments. The figure below shows a layout of two parts per blank. Part I is assume to be pivoted at (0,0) and has only one degree of freedom (q1). Part 2 has three degrees of freedom (x2 , y2) : the location of a reference point on the body and q2 : its orientation w.r.t. the fixed frame of reference. The fifth degree of freedom is the repeat distance l. Thus there are total five variables for the two part

problem. This formulation allows to squeeze the objects into a smaller area in a continuous fashion. The objective function evolves as follows

$$Obj = pressure * scrap + penalty * OVERLAP$$

The weight associated with scrap cost in the objective function is analogous to pressure and generally has a value of one. The OVERLAP cost is penalized because total overlap area at the end of the run should be zero.

The cooling schedule depends upon the number of parts per blank. In a tested study [9], it is reported that for nesting of one blank , the employed cooling schedule, using an initial temperature of 30.0 and a final temperature of 0.001 with 50 iterations at each temperature, required 840.8 seconds on the Sun Microsystems 3/280 workstation. In all the blank nesting problems, the temperature is reduced by a factor of 0.9. It was observed that if the algorithm is started at high enough temperature and if the number of moves attempted at each value of temperature is large (> 200) then the algorithm has a greater chance of converging to a local minimum.

As the number of blanks increases, the number of variables goes on increasing and the number of moves at each temperature value is also increased. The starting temperature is kept high enough to allow a majority of configurations to be accepted and the stopping temperatures are kept low enough to accept only those configurations with the smallest value of the objective function.

2.6 Problem Formulation

The rectangular cutting stock problem is a simplified version of general 2-D allocation problem in which one is provided with a supply of rectangular sheets and an order for a specified number of each of certain types of rectangles. The task is to find a way of cutting the shapes out of the sheet that minimizes the total waste. If computation is significant factor then one may not go up to optimum solution but satisfy with the near optimal solution that require less time to generate. The allocation problem may have some constraints like where the shape can be placed or not. A lot of research has been done for nesting of rectangular patterns. These solutions can also be applied to nesting of irregular patterns after the approximation methods.

With the advent of high speed digital computers even computation intensive optimization techniques like linear and Dynamic Programming are also being tried. A lot of work has also been done recently using Genetic Algorithms (GAs) to solve nesting problems. GAs are probabilistic optimization techniques similar to simulated annealing. It is, however, easier to map the problem using GAs compared to complex cooling schedules used in simulated annealing.

This thesis focuses on use of GAs for nesting of rectangular shapes. Rectangular shape nesting problem is selected because it is relatively simple and the algorithms have been studied extensively. This has

allowed the thesis to focus on the use of Genetic Algorithms rather than algorithmic details of nesting problem. The next chapter discusses two Genetic Algorithm based approaches.

Chapter 3

Genetic Algorithms for Nesting Solutions

GAs are very different from traditional search and optimization methods used in engineering problems. GAs are being used in every problem domain because of their simplicity, ease of operation, minimal requirement and global perspective. This chapter discusses about the basics of GAs, their working and theories behind working of GAs.

GAs are adaptive search and optimization algorithms that mimic the principles of natural genetics. These algorithms are motivated by the natural genetics and natural selection. The idea behind GAs is to do what nature does. In nature, most organisms evolve by means of two primary processes : natural selection and reproduction. The first one determines which members of a population survive to reproduce and the second ensures the mixing and recombination among the genes of the offspring. The mixing allows the creatures to evolve more rapidly than they would if each offspring contains a copy of the genes of a single parent modified by an occasional mutation. [1]

In natural world, the selection is very simple. If an organism fails some test of fitness, such as recognizing a predator and fleeing, it dies. Similarly, in algorithmic details, one can easily find a way to merit a solution based on its fitness. Crossbreeding of solutions is not difficult once the genetic code is constructed. Genetic code represents the structure of a program in the same way as DNA represents the structure of a person or a mouse.

Genetic Algorithms use vocabulary analogous to natural genetics. The commonly used terms are explained below :

- **Chromosomes** represent the component vector. It is made of genes where each gene represents a feature or character. Genes are arranged in linear order and every gene controls the inheritance of one or several features. Chromosomes may be referred as strings, vectors or solutions depending upon the context.
- **Allele** is the possible value of a gene.
- **Locus** is position of a gene in chromosome. Genes of certain characteristics are placed at certain positions of chromosome.
- **Genotype** is a chromosome or a collection of chromosomes. Each genotype is a potential solution to the problem.

- **Phenotype** is the physical meaning or value of a particular chromosome. The function to calculate this value is externally defined by the user and is called fitness function.
- **Genetic Operators** manipulate the present chromosomes to produce the new population of chromosomes. The most commonly used operators are crossover and mutation.

3.1 Working Principles of GAs

The working principles of GAs are very different from those of traditional optimization techniques. A genetic algorithm for a particular problem must have the following components :

3.1.1 Encoding Mechanism

In order to use GAs to solve a problem, genetic code has to be constructed from the decision variables. At the end, these variables will represent the solution. An encoding mechanism is adapted to map the problem's decision variables to a genetic code or chromosome. In most of the cases, binary strings are used. The length of the string is usually determined according to the accuracy of the desired solution. If a five bit binary string is used to code a variable x then the string (0 0 0 0 0) decodes to Xmin and string (1 1 1 1 1) decodes to Xmax. Here, Xmin and Xmax are minimum and maximum admissible values of x. Any other 5-bit string will decode to a value that is in the range (Xmin,Xmax). It is worthwhile to mention here that with 5-bit string there are only 2^5 or 32 different solutions. If the actual solution is not represented exactly by one of these strings, it will get mapped to the nearest one. So, if you want to increase the accuracy of the solution, increase the number of bits in the binary string. It is also noteworthy that as the string length increases, the minimum possible accuracy in the solution increases exponentially [4].

It is not necessary to encode the problem variables as binary strings in all cases. In some studies, GAs have been applied directly on the problem parameters themselves. In such case, the genetic code comprises of a set of real values for design variables rather than 0's and 1's [5].

3.1.2 Fitness Function

GAs mimic the survival-of-the-fittest principle of nature to make a search progress. Therefore GAs are naturally suitable for solving maximization problems. Minimization problems are generally transformed into maximization problems by some suitable transformations. Usually, a fitness function is derived from the objective function keeping in view the behavior of various problem variables. The fitness value of a genetic string or chromosome is known as the chromosome's fitness and it shows how good is the solution in comparison to other solutions. Because the fitness value of the solution varies from problem to problem, it should be normalized to maintain uniformity over various solutions.

The formulation of fitness function is very decisive in the case of GAs. Each problem parameter, given with correct weight in the fitness function, can lead to a faster convergence in accurate region.

3.1.3 Genetic Operators

GAs begin with a population of chromosomes created at random. Each chromosome in the population is evaluated and then the population is operated by three main operators - **Selection, Crossover, Mutation** - to create a better population. These three common genetic operators are explained below : .

Selection : Selection models nature's survival-of-the-fittest mechanism to produce next generation pool. Fitter solutions survive while weaker ones perish. Selection selects good strings in a population and forms a mating pool. Higher the fitness value of a string, higher are the chances of passing its features to the offspring in the next generation pool. There are a number of selection schemes but the essential idea behind each selection operator is that above average strings are picked from the current population and duplicates of them are inserted in the mating pool. Under selection, individual chromosomes can only do one of these three things : they may be born, they may die or they may live. If we consider these events to be moved along synchronously, time step by step, the following general equation solution can be written :

$$M_{i,t+1} = M_{i,t} + M_{i,t,b} - M_{i,t,d} \quad \text{..... (1)}$$

where M is the number of individuals, subscript i identifies the class of individuals with common objective function value f_i , subscript t is the time index, b signifies individuals being born, d signifies dying individuals and lack of b or d signifies living individuals [3]. In the light of above discussion, some selection schemes are discussed below :

Proportionate Selection : It describes a group of selection schemes that choose individuals for birth according to their objective function value. In these schemes, the probability of selection p of an individual in a generation is calculated as :

$$P_{i,t} = \frac{f_i}{\sum_{i=1}^{n} f_i} \quad \text{.....(2)}$$

where f_i is the fitness value of an individual and the total number of individuals is n. In the case of a non overlapping population of constant size n, n selections are made in each generation according to distribution of above equation, it is straightforward to calculate the expected number of copies of an

$$P_{i,t+1} = \frac{P_{i,t} * \bar{f}}{\sum_{i=1}^{n} f_i} \quad \text{.....(3)}$$

individual in next generation :

where $\bar{f}$ is the average fitness value of current generation.

Proportionate reproduction can be implemented in a number of ways. The simplest one is roulette wheel selection. This scheme simulates the spin of weighted roulette wheel. Here, probability of selection of a string can be easily calculated by equation (3). Therefore, the cumulative probability of each string in the population is calculated. The roulette wheel scheme can be simulated by realizing that i^{th} string in the population represents the cumulative probability values from Pi-1 to Pi. In order to choose n individuals for next population, n random numbers are generated from 0 to 1. Thus a string that represents the chosen random number in the cumulative probability range, is copied into the mating pool. This way, a string that covers a large range has more chances to go to the next mating pool.

Tournament Selection : The idea behind this scheme is to choose some number of individuals randomly from a population, select the best individual from this group for further processing and repeat as often as possible. Tournaments are often held between pairs of individuals.

In binary tournament selection, two individuals are chosen at random and the better of two is individuals is selected with fixed probability p, $0.5 <= p <= 1$.

Crossover : This operator is applied after formation of the mating pool to the strings. A number of crossover operators exist in literature. But in almost all the operators, two strings are picked from the mating pool at random and some portions of the strings are exchanged between the strings. In a single point crossover, a random cross site is chosen along the length of the string and all bits on the right side of the crossing site are exchanged. An example of single point crossover follows :

0 0 0 0 0 0 0 1 1 1
1 1 1 1 1 1 1 0 0 0

It is intuitive from this exchange that good substrings from either parent string can be combined to form a better child string if an appropriate site is chosen. Since the knowledge of appropriate site is usually not known, a random site is chosen. With a random site, the children strings produced may have or may not have the good substrings. But this fact is taken care of in the subsequent generations. If the good strings are created by crossover, there will be more copies of them in the next mating pool generated by the reproduction operator. But if the good strings are not created by crossover, they will not be reproduced in the next generation.

In a two point crossover, two random sites chosen along the length of chromosome and the contents bracketed by these sites is exchanged. Two point crossover eliminates the single point crossover bias towards bits at the ends of the string [15]. This idea can be extended to multiple point crossover where the string is treated as a ring of bits divided by k crossover points into k+1 segments. One set of alternate segments is exchanged between the pair of strings to be crossed.

The extreme of multiple point crossover is uniform crossover where bits of a string are exchanged rather than segments. At each string position, the bits are probabilistically exchanged with some fixed probability.

The purpose of crossover is twofold. The main purpose is to search a parameter space. Other aspect is that the search is needed to be performed in a way so that the information stored in the parent string are maximally preserved between parent and children strings. In order to preserve some good strings, not all strings in the population are used in the crossover. If the crossover probability is Pc then Pc*100% strings are used in the crossover and rest (1-Pc)*100% are simply copied to the new population. Crossover is mainly responsible for the search aspect of genetic algorithms.

Mutation : It plays the role of regenerating genetic material lost by reproduction and crossover. For example, if all the strings in the population has converged to zero at a particular position and the optimized solution has 1 at that position then neither reproduction nor crossover can recover it. Mutation of an individual string involves toggling its bits individually depending upon some probability called mutation probability. This operator is needed to maintain diversity in the population. Furthermore, for local improvement of a solution, mutation may be found very useful.

3.1.4 Control Parameters

Crossover probability P_C, mutation probability P_m and population size are the control parameters and should be specified before executing the algorithm. Values of control parameters do not change during the execution of SGA. P_C and P_m control the number of crossover and mutation in a new population generation and has a value between 0 to 1. In SGA, after choosing the strings from the pool, a crossover occurs only if a randomly generated number between 0 and 1 is less than P_C otherwise the strings remain unaltered. Increase in crossover probability increases the recombination of strings but it also increases the disruption of good strings. Similarly in the case of mutation, increase in P_m tends to transform the genetic search into random search but also reintroduces the lost genetic material. A large population size increases the diversity in strings. It also reduces the probability of convergence to a local optimum but increases the time required for the population to converge to optimal region in the search space.

Although the choice of optimal control parameters still remains an open issue, several researchers have proposed control parameter sets that guarantee good performance on carefully chosen test beds of objective functions. Two distinctive sets have emerged : One has a small population size and relative large mutation and crossover probability, while the other has a large population size but much smaller crossover and mutation probabilities. These categories are :

- P_C : 0.6, P_m : 0.001, Size : 100
- P_C : 0.9, P_m : 0.01, Size : 30

From second set, it is clear that a small population needs a high level of string disruption.

3.2 Adaptive Genetic Algorithms

During the recent developments in GAs, various components of GAs have been studied in depth. Apart from various schemes in above discussion, non traditional techniques that are different from the traditional setup of GAs have also been evolved.

In practical situations, the static configurations of control parameters and encoding in GAs have some drawbacks. Control parameter settings, that are optimal in the earlier stages of search, become inefficient during the later stages. Similarly, encoding becomes coarse as the search progresses and the fraction of search space that GA focuses on becomes progressively smaller.

To overcome these problems, adaptive strategies are chosen where the control parameters adapt themselves to the problem after every generation. For example, In one of the strategies, the mutation rate exponentially decreases with increasing number of generations to gradually decrease the search rate and the disruption of strings. Another approach dynamically modifies the rates of use of the genetic operators based on their performance. Each operator is evaluated for the fitness values of strings it generates in the subsequent generations [16].

Genetic algorithms are useful for their domain independence, robustness, ease of modification and parallel nature. These algorithms may be improved by hybridization with problem specific information. One of the most important aspects to be taken care of is that how the problem coding should ensure the building block hypothesis underlying the working of GAs. Nevertheless, across a wide spectrum it is true that even a SGA will probably perform as well as or better than most general techniques.

In spite of solving so many engineering problems, GAs are not considered to be cure all. In engineering design optimization, they will be most successful in comparison with other methods in problems where search space is huge, where number of design variables is large and where known methods fail to provide adequate results. An efficient way to use GAs is to use them in conjunction with traditional search and optimization methods.

3.3 Nesting Solutions with GAs

Solutions of nesting problems using traditional approaches and simulated annealing has already been discussed in chapter 2. For optimal cutting of free patterns, which are used in the cutting processes of clothes, metal sheets, there have not been so many studies because of the difficulty of theoretical treatment of two dimensional search spaces. The size of a two dimensional search space, given by the product of each dimension, becomes quite large as compared to one dimensional search space. Various algorithms differ in their handling of this large search space. While using GAs, the key to the solution

lies in the representation of the two dimensional patterns and their location in two dimensional search space. Two approaches, from literature, for solving the nesting problem using GAs are discussed below.

3.3.1 Genetic Algorithms for Optimal Cutting

According to Ono and Watanabe [8], it is difficult to get proper results with a usually adopted tactic to represent the two dimensional positions in genes. They adopt a method where this problem is solved as (one dimensional) ordering problem by incorporating layout determining algorithms into GAs to reduce the dimension of the search space. At the same time, LDAs are designed to get the solution as quickly as possible. This method is designed to meet the following requirements :

- Lay the shapes on the sheet without mutual overlapping
- Make the required sheet length minimal

According to Ono and Watanabe, regular way of representing the two dimensional positions in binary strings, increases the chromosome length in turn increasing the search space. Instead if it is possible to solve the problem as an ordering one, the size of the search space is N! where N is the number of shapes. To reduce the search space, they have solved the problem by an ordering GAs, introducing LDAs to change a two-dimensional search into one dimensional search. The GAs and LDAs work sequentially, communicating with each other. The GAs determine the order of patterns to arrange them on the sheet and send it to LDA. LDAs fix the position of each pattern according to the order given by GAs so that each pattern on the sheet doesn't have an overlap and an extra opening. After the layout of all the patterns, the sheet length is calculated and sent back to the GAs. With this data, GAs determine the order of pattern arrangement to make the sheet length minimal by applying genetic operators. Figure 3.1 gives the configuration of this system.

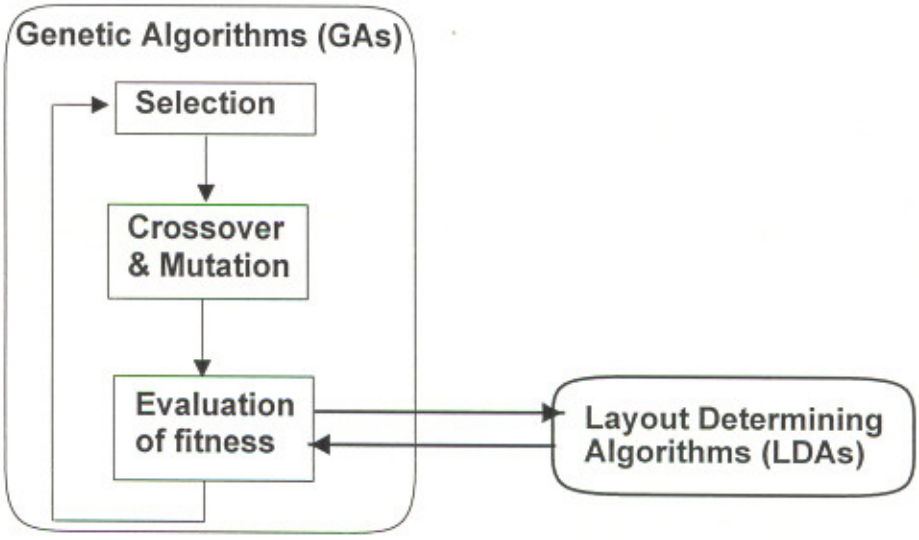


Figure 3.1 - Nesting solution with GAs and LDAs

Layout Determination Algorithms (LDAs) : LDAs search the allowable positions for

each pattern on the sheet, each starting from the origin of the sheet and walking in the direction of the width (Y-axis) while stepping in the longitudinal direction (X-axis). When the allowable position is found, the pattern is set there and a marking is given to the sheet data to store the occupied area by the pattern.

Various techniques for checking the overlap are used in LDAs to reduce the search time. For example, pattern A is the pattern to be searched for its position on the sheet. At first a check is done to see if there is an overlap between pattern A and the shapes already fixed on the sheet. If the pattern A overlaps, it jumps to the next search position by a calculated amount. Once the position is found where no overlap is there, the pattern A is set there.

Genitor [6] is used as a software workbench to solve the ordering problem in this case. The genes are represented by the path representation, with the rank method for the selection. For crossover, three methods have been compared among various methods applied. For mutation, the method exchanging one random element of genes with another random one is used.

Ono and Watanabe have been successful in solving a two-dimensional problem as an one dimensional ordering problem by combining GAs with LDAs and the required time is also considerably reduced.

3.3.2 Nesting of Two Dimensional Shapes using GAs

Ismail and Hon [7] use a combination of heuristic and GA based approach. Heuristics are used to transform the two dimensional shapes into a format that can be effectively used by GAs while GAs are used to arrive at the solution.

All of first, a discrete image of the shapes is produced by finding the minimum circumscribing rectangle for each of the shapes. The shapes are then rotated so that the axes of the minimum circumscribing rectangle are aligned with global X and Y axes. A grid consisting of identical square elements is superimposed on the minimum circumscribing rectangle of each shape. The size of the square element is uniform across all the grids. As a result some of the minimum circumscribing rectangles are increased in size in order to contain whole square elements. The grid is scanned horizontally and vertically to identify whether the elements lie partially or wholly inside the shape. Accordingly, the grid element is marked filled or empty.

To reduce the optimization search space, a further set of conditions and assumptions is introduced. As the shapes need not be rectangular, there are 8 possible orientations in which a shape can be placed on the grid. The working area in which the shapes will be placed and arranged is represented by a square grid. The size of the grid elements in the layout grid is the same as that used for the shapes. The number of elements on each side of working area is approximated by N_s as follows :

$$N_s = N_{max} * \sqrt{N_s}$$

Where N_{max} is the number of two-dimensional shapes. The size of the search space is given by :

$$L = (8 * N_s)^{2*N_s}$$

The layout problem is now reduced to that of moving and arranging a number of discretized two dimensional shapes on the layout grid. We have to identify the row and column locations (R_i,C_i) and the orientation of each shape which result in smallest overall occupied area.

Ismail and Hon [7] have carried the tests for randomly generated shapes using this algorithm. The results have also shown that the use of swap operator results in an improvement in the objective while the creep operator has no apparent effect on the results. One set of results is shown in Figure 3.2.

Figure 3.2 - Solutions after generation 1500, utilization 90.5%

3.4 Discussion

Ono and Watanabe approach uses GAs only to decide the order in which the shapes are to be placed on the sheet. The placement of the shapes is controlled by LDAs to ensure that there is no overlap. Ismail and Hon approach lets GAs decide the ordering as well as the placement of shapes. Here, the solution quality is driven completely by the fitness function. Because the goal of this thesis is to explore the use of GAs to solve nesting problem for rectangular shapes, the work done in thesis uses the approach taken by Ismail and Hon as starting point. The proposed solution is discussed in detail in the next chapter.

Chapter 4

Proposed Solution

In the problem of rectangular packing, the objective is to pack all the rectangles in a minimum enclosure. A fit and feasible solution is one that places all the rectangular shapes on the smallest piece of sheet without any overlap. Fitness of a solution will reflect how well or tightly the shapes are packed on the sheet. The objective function will consist of terms representing the utilization of the sheet on which the rectangles are placed and some penalty terms to discourage behavior that is not acceptable. Penalty represents a variable that is to be minimized such as number of overlaps or the total sheet area. The solution starts with the maximum estimated size of sheet that may be required to place all the shapes. As the initial solution is generated randomly, the shapes are placed in a total haphazard way on the sheet and occupy a large part of it. There may not be many shapes around the origin of the sheet, reducing utilization of the sheet because of large unutilized segments in the minimum enclosure of the solution. As the GAs start working, solutions evolve through successive generations and more fit solutions start appearing.

The approach taken by Ismail and Hon [7] is extended to include the following enhancements :

- The objective function has been modified to change the penalty terms. Now, there are three penalty terms to control the solution.
- Adaptive GAs have been used to improve the speed of convergence to the optimum solution.
- The sheet is shrunk throughout the generations to increase the utilization and the search space. Reducing the sheet size may increase number of overlaps but the balance between the shrinking factor of sheet and the number of overlaps is taken care of by the deterministic approach.

All elements of the proposed solution are discussed below.

4.1 Problem representation

Various steps are involved in the representation of the rectangular packing problem in a format that can be effectively used by GAs. Two dimensional rectangular shapes are represented as a discrete image as shown in Figure 4.1. This transformation is carried out as follows :

1. The rectangular shape is rotated so that the axes of the shape is aligned with global X and Y axes.
2. A grid consisting of identical square elements is superimposed on the aligned rectangular shapes. The size of the grid element is kept same for all the shapes and is decided by the user. As a result

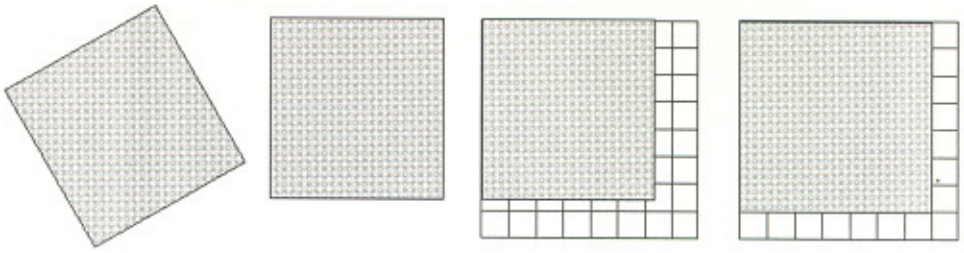


Figure 4.1 - The two dimensional rectangular shape conversion

of using uniform size grid elements, the size of some of the shapes may have to be increased in order to fit whole square element.

3. The sheet on which the shapes will be placed is also represented by a large square grid as shown in Figure 4.2. The size of the grid element will remain same as in the case of the shapes. The number of grid elements on each side of the square grid is approximated as follows :

$$N_g = N_{MAX}\sqrt{N_S}$$

where N_S is the number of two dimensional rectangular shapes and N_{MAX} the length of the longest edge among all the shapes

4. The discretization of shapes and the sheet makes the handling of overlaps manageable. Because the shapes are represented by grid elements, only those shape orientations can be considered where the shapes are aligned to the axes of the grid. For rectangular shapes, rotation of shapes is restricted to a 90° rotation only. The layout problem is now reduced to that of moving and arranging a number of digitized rectangular shapes (N_S) in discrete steps of element size on the layout grid.

As a result of the above transformation, the layout problem is now reduced to that of arranging N_S discretized rectangular shapes on the grid representing the sheet : the location (row and column) and

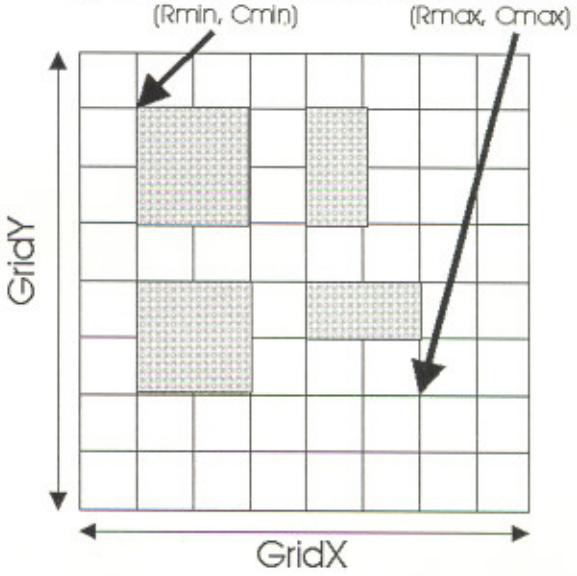


Figure 4.2 - Shape representation with layout grid

orientation of each of the rectangular shapes which results in the smallest overall occupied area without overlap. This is the area of the smallest rectangle enclosing the shapes arranged on the grid of the sheet.

4.2 Objective function

The main aim of the solution is to reduce the area occupied by the shapes without overlap. The closer the shapes are to the top left corner of the working area, higher is the utilization of the sheet. The objective function should, therefore, have terms that encourage high utilization and discourage overlaps. For our representation, the fitness of a solution may be calculated as follows :

$$Fitness = \frac{100 * \sum_{i=1}^{N_s} A_i}{R_{MAX} * C_{MAX} * A_g} - pA * \left(\frac{100 * N_{OVERLAP}}{N_{FILLED}} \right)^{pB} - pC * \left(\frac{R_{MIN}}{gridX} \right) - pC * \left(\frac{C_{MIN}}{gridY} \right) \quad \ldots\ldots\ (1)$$

Where

Ai	:	Area of shape i
R_{MAX}, C_{MAX}	:	LastFilledRow, LastFilledCol in the grid respectively
R_{MIN}, C_{MIN}	:	FirstFilledRow, FirstFilledCol of the grid respectively
gridX, gridY	:	Grid extents
A_g	:	Area of a grid element
pA, pB, pC	:	Penalty weights
$N_{OVERLAP}$	:	Number of overlapping elements
N_{FILLED}	:	Number of filled elements

The first term is a measure of the utilization of occupied area. The other three terms are introduced as penalties for violating the constraints of the problem. The effect of the penalty terms is to reduce the value of the objective function so that unfeasible solution or less preferred solutions would attain a lower fitness value. Penalty weights pA, pB, pC are fixed at the start of the solution.

The first penalty is based on the constraint that solutions in which the rectangular shapes overlap are not acceptable. The penalty for overlapping is based on ratio of the number of overlapping elements ($N_{OVERLAP}$) to the number of occupied elements in the grid. This term will discourage the inclusion of solutions containing overlapping elements in the population. Once $N_{OVERLAP}$ becomes zero, this penalty term will stop contributing to fitness value and this factor won't play any part in convergence of solutions.

The other two penalty terms are designed to force the solution to move towards the top left corner of the occupied area. As the values of R_{MIN} and C_{MIN} increases, these terms will contribute more negative influence to the solution, thus promoting solution with low R_{MIN}, C_{MIN} values and discouraging solutions with high R_{MIN}, C_{MIN} values i.e. the solutions which are far away from the origin.

The above approach may force the algorithm to reproduce a population with a small number of feasible solutions leading to false convergence. In some cases the fitness value may also become negative due to high penalty values. In such cases, negative fitness is assigned a zero value. To deal with these problems, the objective function has been modified to remove the negative penalty terms. New penalty functions were introduced to minimize $N_{OVERLAP}$, R_{max} and C_{max}. Additional weight terms were also introduced to normalize the fitness value over various solutions. This provides more control over the quality of the solution. Penalties pA, pB, pC decide how fast the fitness decreases when $N_{OVERLAP}$ or R_{max} or C_{max} increases. The weight terms wA, wB, wC decide the contribution of each factor towards the fitness value. The new objective function is as follows :

$$Fitness = wA * \left(pA * \frac{N_{FILLED}}{R_{MAX} * C_{MAX}} \right) + wB * \exp\left(-pB * \frac{N_{OVERLA}}{R_{MAX} * C_{M}} \right.$$
$$+ wC * \exp\left(pC * \frac{gridX}{gridX - RMax + 1} \right) + wC * \exp\left(pC * \frac{gridY}{gridY - CMax + 1} \right) \quad \ldots\ldots(2)$$

The four terms play the same part in penalizing the solutions as in the earlier objective function but the manner of their contribution towards the fitness has changed. The third and fourth terms try to move the solution towards the top left corner of the layout grid by minimizing R_{MAX} and C_{MAX}.

The overall penalty has been selected as a decreasing exponential function of $N_{OVERLAP}$. Its operating range is non-negative with a maximum value of 1 when $N_{OVERLAP}$ is 0. As the function is exponential, a small change in pB can drastically change the solutions. The weight term wB can be assigned an appropriate value to give required contribution to the fitness function. As the ratio of N_{ve} to grid area will increase, the contribution of this penalty term towards fitness will exponentially decrease and such solutions will not be preferred in the subsequent generations. The highest contribution of second term to fitness value is wB i.e. when $N_{OVERLAP}$ is zero.

The third and the fourth terms were also made exponentially decreasing functions of grid extent ratios to R_{max} and C_{max}. These terms help in selecting the solutions that are away from the bottom right hand corner of the layout grid. Solutions lying in upper regions are likely to have more fitness and thus get selected for inclusion in subsequent generations.

The new form of the objective function results in faster convergence and with placement of shapes in the right region. The key to fast convergence lies in selecting penalty terms pA, pB and pC. These vary the contribution and influence of each term on the solution. A slight change in any of these parameters may totally change the quality of solutions. For example, if you slightly decrease the value of pC, you may end up getting solutions which lie away from top left corner of sheet. A lot of effort went into the tuning of these parameters during the development phase and it has also helped in the understanding of how GAs work.

4.3 Components of Genetic Algorithms

There are five components of GAs and each is described below for the rectangular layout problem :

4.3.1 Genetic Code Representation

As discussed above, the objective is to code the position and orientation of two dimensional rectangular shapes on the layout grid. The position of each shape is represented by R_i , C_i which are the coordinates of the top left corner of the shape relative to the origin of the layout grid. The orientation of each shape is represented by T_i which corresponds to an anti clockwise rotation of the shape by 90°. The genetic code employed for this problem is a multiparameter binary based code where each positional parameter (R_i ,C_i) and T_i is represented by a binary string. The length of binary string for each parameter influences the performance of the program. As the number of bits in the binary string are decreased, the accuracy of the solution also decreases. For example, initially a 10-bit string represented each positional parameter but the convergence of the solutions turned out to be very slow due to the fact that the disruption in the solutions is very small when mapped to actual parameters. Changing the length of the binary string to use 5-bits for R_i ,C_i and 3-bits for rotation T_i improves the convergence speed. Presently, this 13-bit binary string is used to represent a shape. The software developed for this work gives the user a free hand to choose the binary string length. User can adjust it accordingly to the size of the layout grid.

The overall genetic string length will depend upon the total number of shapes in the problem. Mapping of each individual parameter is done to its actual value (within the specified extents) by the program using the standard mapping techniques [5].

4.3.2 Initial Population

Ismail and Hon [7] have investigated two methods for generating initial population. The first one follows random generation while second approach is based on deterministic technique to evolve feasible solutions. This software adopts the first one which is the classical method of generating populations randomly. Each positional parameter is generated randomly within the specified extents. The weak solutions automatically die out in the evolving generations.

4.3.3 Genetic Algorithm Operators

Two types of genetic operators were used for this problem. The first category is the classical genetic operators i.e. reproduction, crossover and mutation. Roulette wheel selection and Tournament selection with a size of 2 were tried for this problem. For crossover operator, multiple point crossover has been used with 2 , 3 and 4 crossover sites. For mutation, simple mutation operator is used.

The second category consists of two operators - swap and creep. These operators are specific to the problem and are derived using similar principles as those of crossover and mutation. The swap operator aims at testing new alternative shape arrangements by randomly exchanging the position and orientation

of two shapes in the solution. This operator is applied to two shapes that are randomly selected. The creep operator is applied to improve on existing solutions by shifting the shapes closer to the top left corner of the working area. The shapes are shifted by one row position or by one column position when this operator is applied. This operator is also applied on a randomly selected shape from the solution. The frequency of these operators varies from 0 to 1. These probabilities are usually set at the start and are changed depending upon the nature of the GAs.

4.3.4 Elitist

This approach saves the elitist member of each generation. It checks if the fittest member is better than that of previous population, if not, the fittest member of the previous population is reintroduced into the new population by replacing the weakest solution. This approach does not allow the genetic operators to disrupt the best schema so far. This operator is also applied at a controlled frequency so that the program does not converge into the first above average solution. Although a solution can be maintained throughout the generations. At the end, it can represent the best possible solution attained so far.

4.3.5 Control Parameters

According to Goldberg [1], Genetic Algorithms are more effective with moderately sized populations. Therefore, a population size of 50 is used for all the test cases in this thesis. It has produced good results without false convergence.

In the case of simple GAs, the crossover and mutation probabilities were kept 0.8 and 0.05 respectively. The swap and creep probabilities were kept low at 0.1 because of their disruptive nature. No appreciable improvements in solutions have been noticed with changes in swap factor. This is because of the rectangular nature of the shapes.

For adaptive GAs, a set of equations has been used to calculate the crossover and mutation probability factors. Another set of equations has been devised using a similar pattern to control swap and creep probability factors. Ismail and Hon [7] had used a static set of probabilities for swap and creep operators. The following equations guide the genetic operators if the adaptive GAs are used :

$$P_{cross} = \begin{cases} k_1\left(\frac{f_{max} - f'}{f_{max} - \bar{f}}\right), & f' \geq \bar{f} \\ k_2, & f' < \bar{f} \end{cases} \qquad k_1 = 1.0,\ k_2 = 0.5$$

$$P_{mutation} = \begin{cases} k_3\left(\frac{f_{max} - f}{f_{max} - \bar{f}}\right), & f \geq \bar{f} \\ k_4, & f < \bar{f} \end{cases} \qquad k_3 = 0.05,\ k_4 = 0.001$$

$$P_{swap} = \begin{cases} k_5\left(\frac{f_{max} - f}{f_{max} - \bar{f}}\right), & f > \bar{f} \\ k_6, & f \leq \bar{f} \end{cases} \qquad k_5 = 0.1,\ k_6 = 0.2$$

$$P_{creep} = \begin{cases} k_7\left(\frac{f_{max} - f}{f_{max} - \bar{f}}\right), & f \leq \bar{f} \\ k_8, & f > \bar{f} \end{cases} \qquad k_7 = 0.1,\ k_8 = 0.2$$

where

$\bar{f}$: average fitness value of population

f_{max} : maximum fitness value of population

f : fitness value of solution

f' : larger fitness value of solutions to be crossed

Crossover, mutation and swap probabilities are constant when the fitness value of solution is below the average fitness of population. These probabilities decrease as the fitness value of solution approaches the elitist fitness value. So, more the solution is near to elite solution, less is the disruption caused by these three genetic operators. This way, the good building blocks are saved for the next generations. The creep operator plays its part in a different way. When the disruption due to other operators is high, the creep probability is kept low i.e. when the fitness of solution is near average fitness value. When the solution approaches elite solution, the disruption in it due to other genetic operators is very small. Now, the creep operator tries to improve the solution by shifting it horizontally or vertically towards the top left corner of the layout grid. If the new solution is worse than the previous solution it gets rejected in the next population selection. This way, crossover, mutation and swap operators are used to converge the solution rapidly to a near optimum solution while creep operator is used to fine tune these solutions.

4.4 Discussion

A software package has been developed to implement the proposed solution discussed in this chapter. This software has been used to investigate the effect of control parameters on the convergence speed and the quality of nesting solutions. The design and implementation of the software along with the results is discussed in next chapter.

Chapter 5

Design and Implementation Issues

Generic softwares for Genetic Algorithms are available in literature and public domain. Most of these softwares are developed in C or Pascal. To use them for a specific application problem, user has to only redefine the fitness function and recompile the program. A software package has been developed for use of Genetic Algorithms using C++ and object oriented techniques. The main purpose behind this effort was to classify the objects related to GA problem and make a reusable software. The software consists of two modules. The first module finds the solution for nesting of rectangular shapes using Genetic Algorithms. This software module has been developed using IBM VisualAge for C++ under Windows 95 and is fully portable to any ANSI C compiler as it does not use any system dependent calls. The Second module is application dependent and is used to display the solutions. This module uses graphic functions to draw shapes and can be ported to any other compiler by rewriting one or two functions. This chapter discusses the design and implementation of these modules in detail.

The main task in design of an application is to identify the various objects related to problem and their relationships. Objects that are identical except for their state at execution time are grouped into a class of objects with identical characteristics and behavior. So, a class is a data type like int or float. The only difference is that now user can define its own data types specifically required for an application instead of using the predefined ones. Analysis, Design and Implementation are three essential steps in development of a software. These steps are described in the following sections.

5.1 Analysis

This step involves the understanding of the problem domain, system requirements and the user input and output. To summarize the working of Genetic Algorithms, a population of initial solutions called chromosomes is taken and various genetic operators are applied on this population to produce a better population. Two main objects related to this GA part are chromosomes and population. Now to specify the problem input parameters, a storage object consisting the information about number of input shapes, shape data and some grid specifications is also required.

5.2 Design Details

Here, we refine the output of analysis phase considering the constraints of the system. Three main classes identified in analysis phase are - Chromosome, Population and LayoutJob. These three classes interact with each other in the main program to produce the final solutions using GAs. Given below is the detailed description of each class :

5.2.1 Chromosome

This class encapsulates the whole functionality of an individual solution. The state of a chromosome varies for its various instances but some attributes - number of genes, length of each gene, minimum and maximum value of each gene - will remain the same for all instances of chromosome object. These attributes are known as class variables and are declared as static variables in the class. Apart from these static attributes, a chromosome also has a list of genes. There are some derived attributes which are not input but are calculated. For example, fitness value, probability of selection and expected count of a chromosome in next generation are calculated for every chromosome.

Each chromosome has a fitness value which is calculated by its fitness function. Now, the chromosome class has only information about its genes and their decoded values. It does not contain any problem specific information. To calculate the fitness value, it needs the problem specific information. To isolate the problem specific behavior, all the basic functionality of chromosome has been kept in a base class called Chromosome. To specify the problem dependent behavior, you can inherit a class from base class Chromosome and override the function to calculate fitness. This way, the base class will provide all the functionality of encoding and decoding of solutions and the derived class will carry the problem specific functions of evaluating the fitness of the solutions. Here, the user does not have to know the details about encoding and decoding mechanism.

5.2.2 Population

Population is a set of chromosomes. It may represent an initial population or a population after a number of generations depending upon the stage of the program execution. Each population will contain a chromosome that will represent the elitist solution obtained so far. The basic attributes of population are size of population and a set of chromosomes. Average fitness and total fitness of the population are calculated. The behavior of population includes the genetic operators that are applied on population to generate a new population. The probabilities of genetic operators like P_{cross}, $P_{mutation}$ is stored statically. For generation of initial population the solutions are generated randomly within the specified constraints at the time of construction of object. This class also stores the statistics of each generation like number of crosses, mutation etc.

The nesting problem defines a set of operators that are problem specific. To implement these operators, a new class has been derived from Population, which inherited the generic attributes and behavior from the base class and two new genetic operators Swap and Creep has also been added. The derived class also gives the choice between SGAs and AGAs.

5.2.3 LayoutJob

This class encapsulates the problem dependent data and functions for conversion of shapes in a format that can be recognized by Genetic Algorithms. It stores the number of shapes and coordinate data of each

shape. Input data can be read from a file or can be set by access functions. This class also contains the layout grid that is later used in Chromosome class to evaluate the solutions.

5.3 Implementation Details

For implementation of Genetic Algorithms for nesting of rectangular shapes, base classes Chromosome and Population have been used. LayoutChromosome is derived from Chromosome and LayoutPopulation is derived from Population to add some problem specific functions. Figures 5.1 shows the derivation of LayoutPopulation and LayoutChromosome class.

Figure 5.1 - Class derivation

In the main program, the input data for shapes is read in an object of type LayoutJob. After resetting the input shapes in their minimum enclosure orientation, an object of LayoutPopulation type is created with shape information from LayoutJob. Initial population is generated randomly when the LayoutPopulation object is created. The controlling parameters for objective function and for Genetic Algorithms are read from data files prepared by the user and the subsequent generations are generated by applying genetic operators to the initial population. This process is repeated until the number of generations specified by the user have been generated.

The program output contains three types of data : statistical information for each generation, shape data and solution details to enable display of output.

5.4 Experimental Results and Discussion

The software for this work has been designed in a way to facilitate experimentation with GAs. The software has been tested with several test cases for SGAs and AGAs. This section discusses the difference in the working of simple and adaptive GAs. Crossover operator has been experimented with 2, 3 and 4 crossover sites. For the selection operator, roulette wheel and tournament selection with tournament size of 2 are used.

Results obtained with roulette wheel selection were consistently better than tournament selection. So, only roulette wheel selection is used in the test cases discussed below.

For the first test case, population size is 50, P_{cross} is 0.8 and $P_{mutation}$ is 0.05. Both P_{swap} and P_{creep} are fixed at 0.1 for both SGA and AGA. Figure 5.2 shows the results for SGAs while figure 5.3 shows the results

using AGAs. It is evident from the results that 2-point crossover gives best results for both SGA and AGA. The fitness value increases very rapidly in the initial generations. This is due to a decrease in the number of overlaps. Initially, there are a large number of overlaps which decrease the fitness value. But in subsequent generations, the solutions with overlaps do not get selected because of low fitness. Thus the average number of overlaps decrease very fast in initial generations, increasing the fitness rapidly. When the number of overlaps becomes low, other penalty functions also start playing their part in convergence of solutions. The penalty and weight factors in the objective function also play a very important part. These factors have to be adjusted such that all the penalty functions contribute significantly. For example, if pB is kept very high then a small number of overlaps may decrease the fitness of solution by such a large value that the contribution of other penalty functions may not play any part in selection of this solutions for subsequent generations. For solutions in this case, the shapes are not packed adequately in the top left corner of the layout grid. This is because there is no way to detect the amount of packing of shapes in the objective function : All solutions with same R_{max} , C_{max} and no overlaps have the same fitness value.

For the second test case, all the control parameters are kept the same as in the first case but an additional term is added to objective function to measure and encourage packing of shapes. The layout grid is divided into four quadrants and the utilization is calculated for each quadrant. The objective function gives a higher weight to the solutions with high utilization of the first quadrant. Now the solutions with same R_{max} , C_{max} and no overlaps have different fitness values depending on the utilization in each quadrant. The results for SGA and AGA are shown in figure 5.4 and 5.5 respectively. There is a significant improvement in the quality of the solutions as compared to the first test case. There is a rapid convergence towards a well packed solution. However, it is noted that the solutions can be improved by applying a slight creep to some of the shapes.

In the third test case, adaptive creep and swap are introduced to remedy the above behavior. The results for AGA, shown in figure 5.6, demonstrate a clear increase in the fitness value of solutions. The quality of the solutions in terms of packing also improves dramatically.

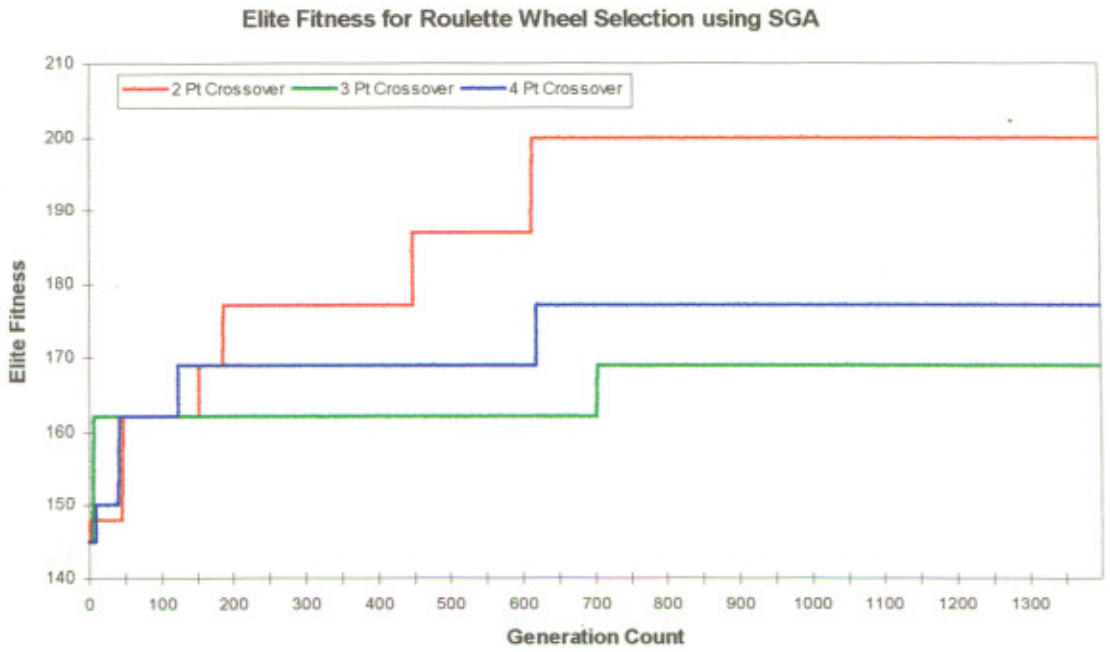


Figure 5.2 - Elite Fitness comparison for *SGA* (without packing)

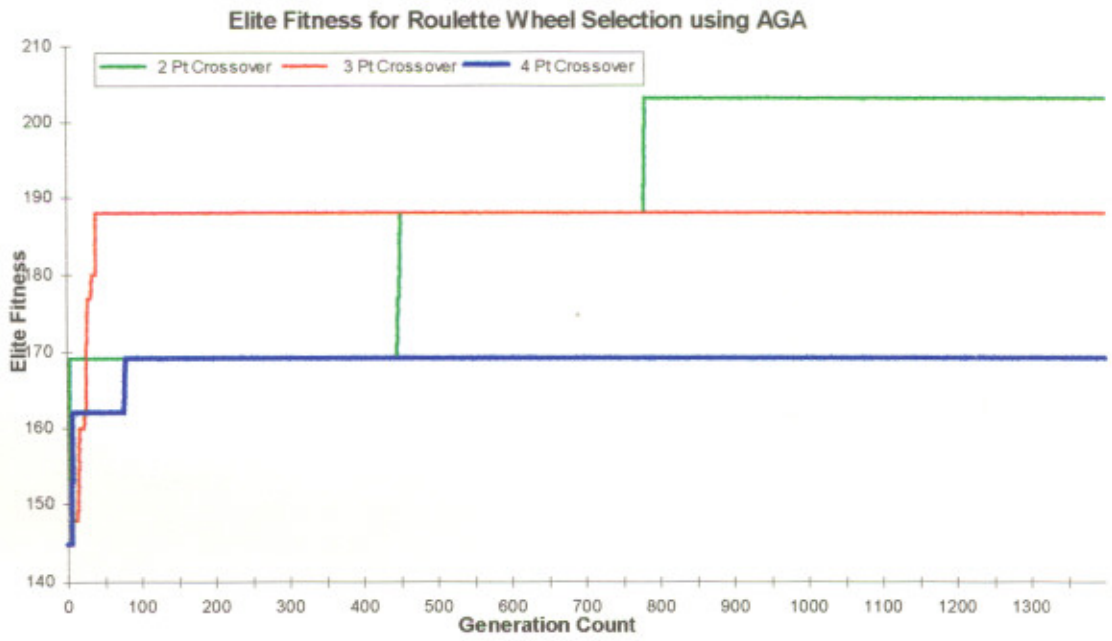


Figure 5.3 - Elite Fitness comparison for *AGA* (without packing)

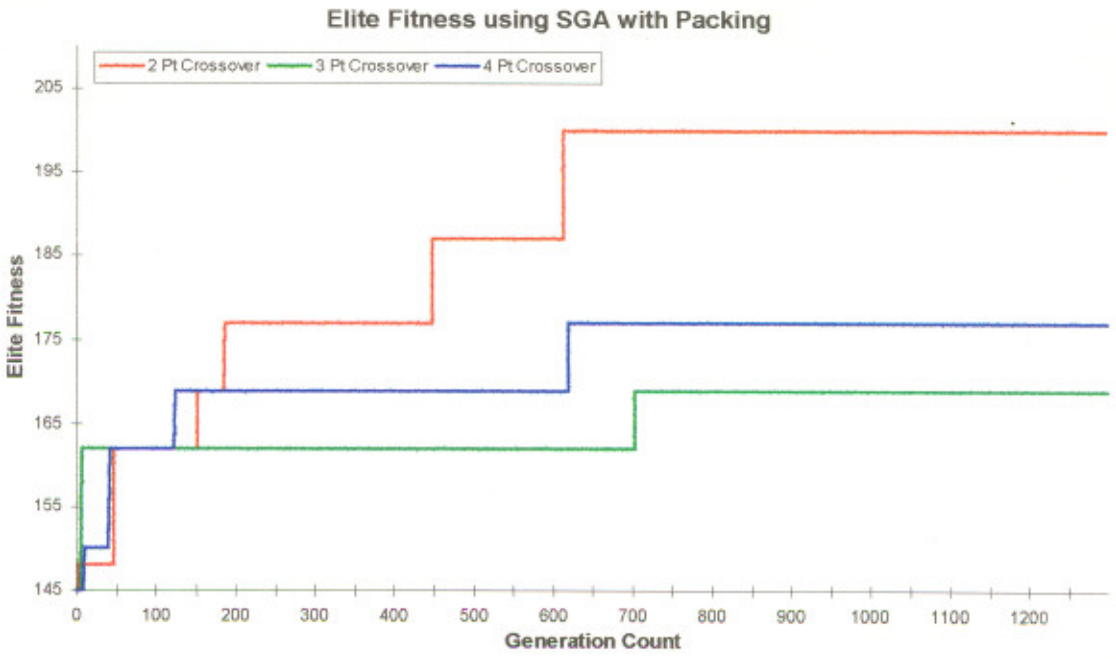


Figure 5.4 - Elite Fitness comparison for *SGA* (with packing)

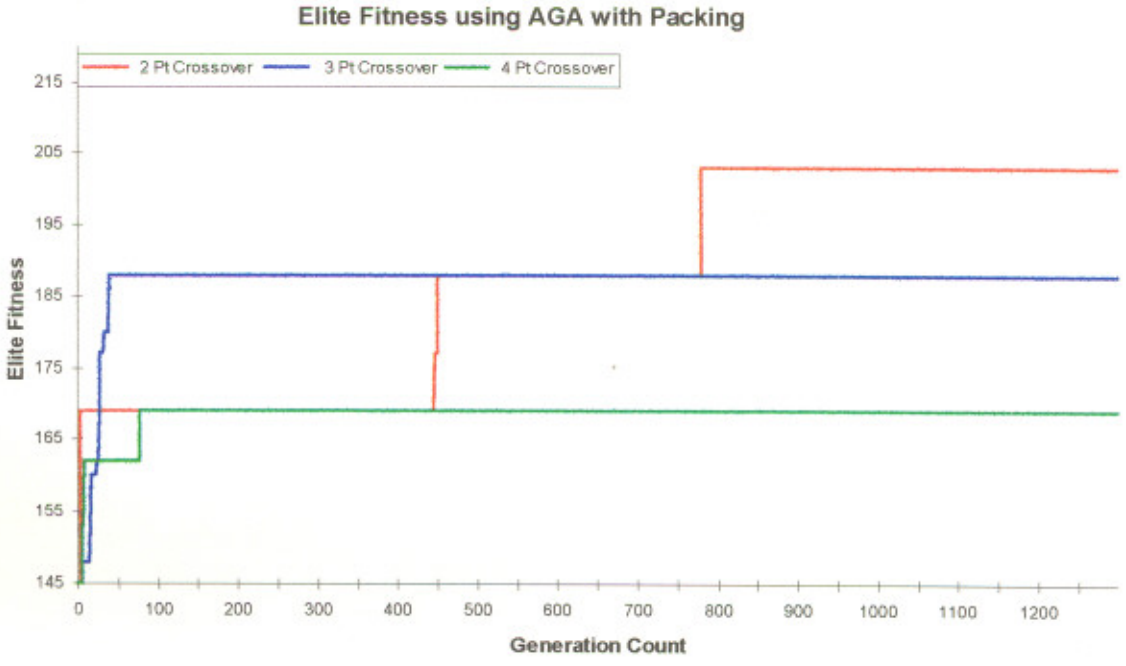


Figure 5.5 - Elite Fitness comparison for *AGA* (with packing)

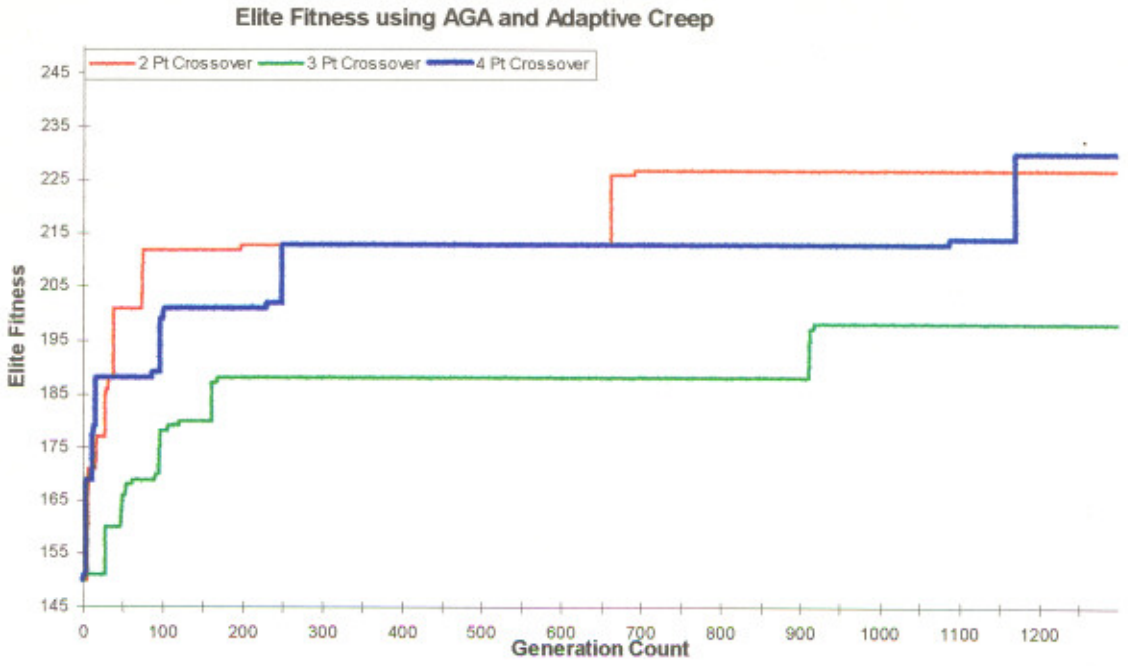
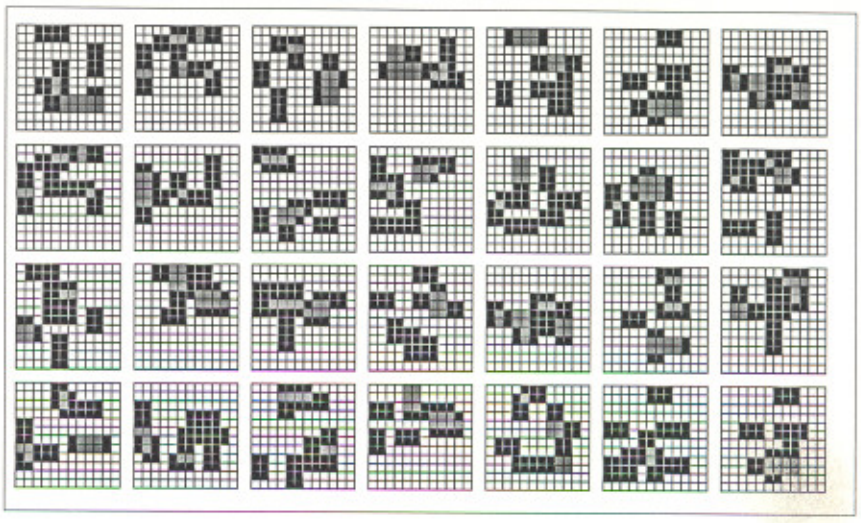


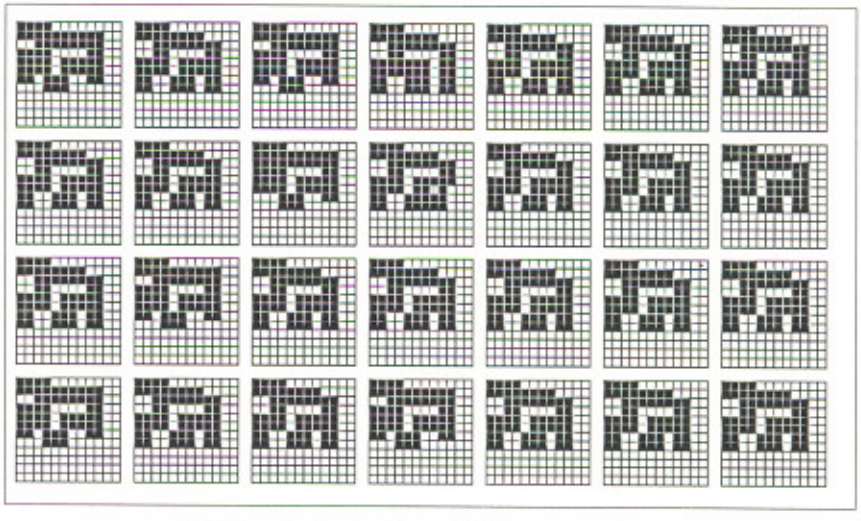
Figure 5.6 - Elite Fitness comparison for AGA (with packing and adaptive creep)

Some of the solutions for the three test cases discussed above are shown in figures 5.7 through 5.9. The filled elements in the layout grid are represented by the black, overlapping elements by gray and empty elements by white. These figures show the improvement in quality of the solutions that results in the fitness increase shown in the figures above. It is evident from the following data that emphasis is to reduce the number of overlaps in the initial generations while in the later solutions shapes tend to pack towards the top left corner of layout grid.

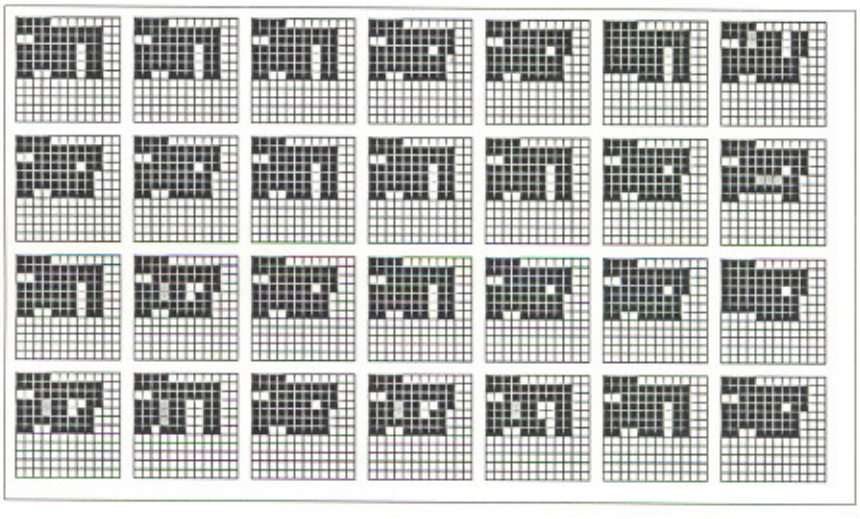
Comparison of Figure 5.7 and 5.8 shows the difference in packing of shapes due to introduction of packing terms in the objective function. The solutions after 1450 generations in Figure 5.8 shows that the solutions can still be improved further by a slight creep. As a result of this, adaptive creep has been introduced to increase the probability of creep in the later generations when the contribution of other three genetic operators is minimal. The results are shown in figure 5.9. After 1450 generations, the shapes are very closely packed. Further improvements in the solutions are possible if it is run for a larger number of generations.



(a) Initial Population

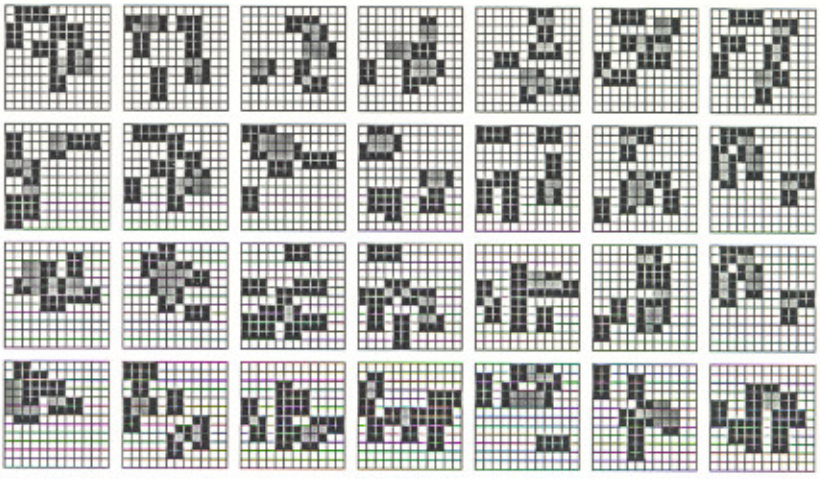


(b) Population after 500 generations

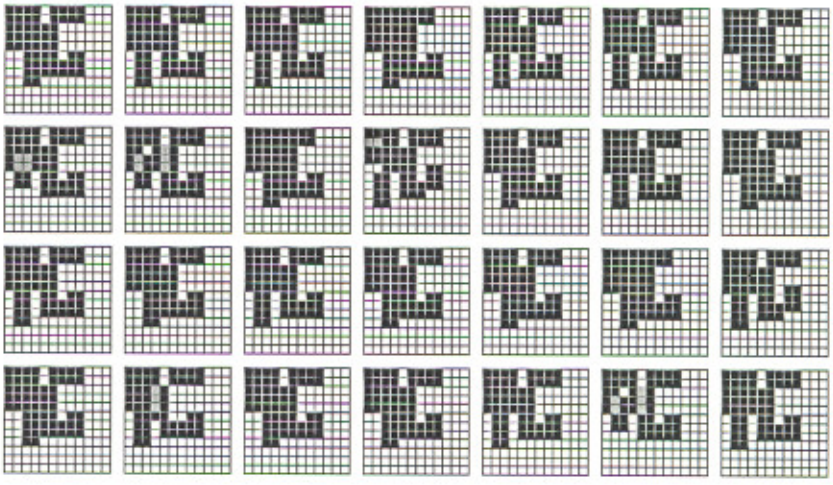


(c) Population after 1450 generations

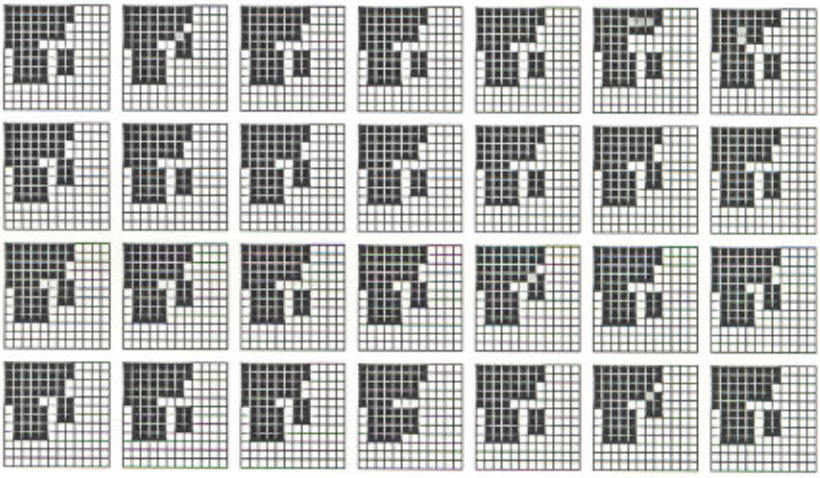
Figure 5.7 - Solutions without packing using *AGAs* with 2-pt crossover



(a) Initial Population

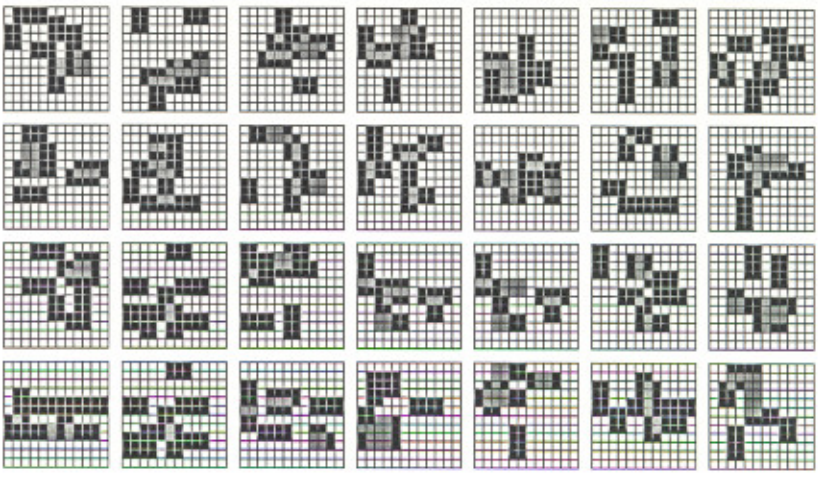


(b) Population after 500 generations

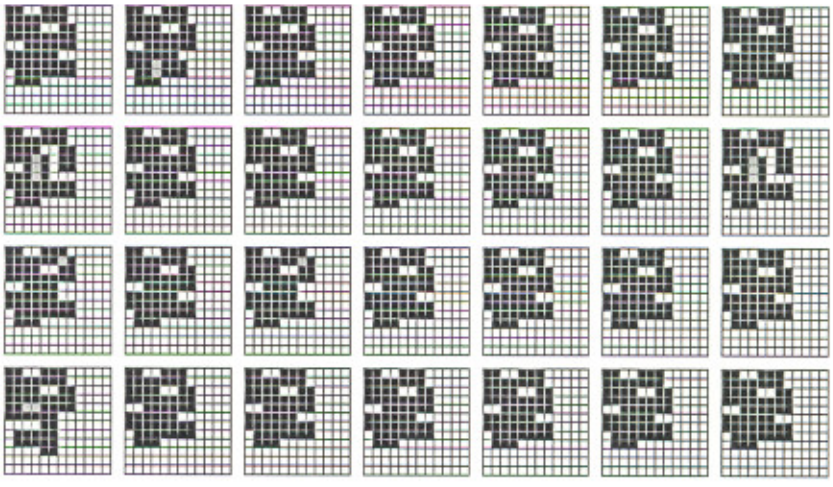


(c) Population after 1450 generations

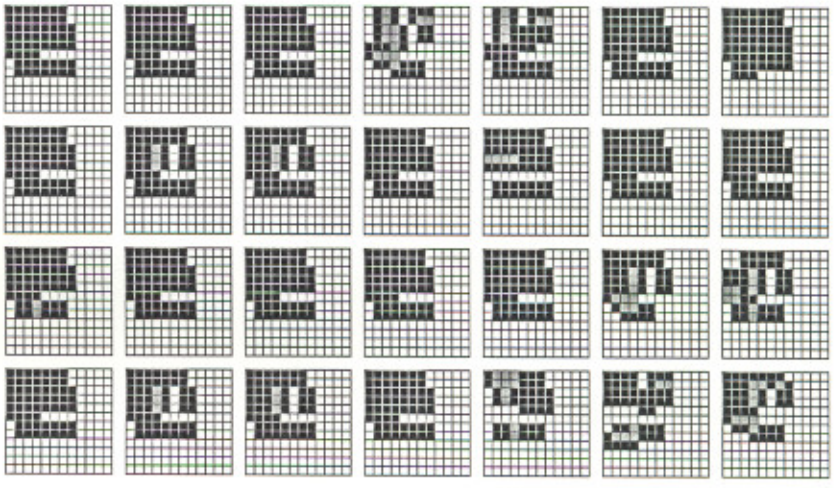
Figure 5.8 - Solutions with packing using AGAs with 2-pt Crossover



(a) Initial Population



(b) Population after 500 generations



(c) Population after 1450 generations

Figure 5.9 - Solutions with packing using AGAs with 4-pt Crossover and Adaptive Creep

Chapter 6

Conclusions and Future Scope

Various nesting solution techniques for regular and irregular shapes have been studied. Some techniques, employing different methods, have also been reported in this thesis. Genetic Algorithms based nesting solutions are also described briefly.

A new method based on Genetic Algorithms for solving nesting problem for rectangular shapes has been proposed in this thesis work. A software package for the proposed method has been developed using IBM VisualAge for C++ under Windows 95. The software has been tested for Simple Genetic Algorithms (SGA) and Adaptive Genetic Algorithms (AGA). A comparison of the results showed that AGA gives better elite fitness and faster convergence than SGA. The overall quality of AGA solution is also better with a larger number of solutions that are close to the elite solution. The results also confirm that special problem dependent genetic operators such as swap and creep help in improving the quality of the solutions. Without these operators, the solutions would have converged to a lower fitness.

Therefore, it is concluded that GAs have a lot of potential for the solutions of nesting of rectangular shapes. By properly deriving the objective function and adjusting control parameters, it is possible to obtain optimum or near optimum solutions for a diverse set of problems. For general applicability, this work may be extended in the following ways :

- *Handling irregular shapes :* The grid representation for shapes used in present work can be replaced by a quad tree representation. This representation is more efficient for representing irregular shapes on a rectangular sheet. The standard quad tree operations like intersection of two quad trees can be used to determine the amount of overlap.
- *Coding Schemes :* The binary coding scheme used in this work can be replaced by real number coding. This scheme will represent the solutions in a format closer to real world solution. However, all genetic operators will have to be modified appropriately.
- *Automated Grid Size :* Presently, the grid element size is user defined. To ensure accurate optimum solutions, it is desirable to have the software determine the grid element size based on the size of the shapes to be placed.
- *Enhanced Creep Operator :* Presently, creep operator moves the selected shape towards the origin. However, sometimes it may be beneficial to move the selected shape away from the origin to reduce overlap. The creep operator can be enhanced to include this functionality.

- *Software Enhancements :* Presently, the software consists of two modules. The first module solves the nesting problem by using GAs and generates output files. The second module displays the results using these output files. These two modules can be merged into one so that the software can dynamically display the solutions while solving the problem.

After incorporating the above modifications to the present work, it will be capable to solve the nesting problem for regular and irregular shapes. The software package with the above enhancements and a lucid user interface can also be marketed commercially.

References

1. Jean Goldberg D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Jan 1989.

2. Holland J. H., *Genetic Algorithms*, SCIENTIFIC AMERICAN, July 1992.

3. Jean Goldberg D. E., Deb *K., A Comparative Analysis of Selection Schemes used in Genetic Algorithms,* Foundations of Genetic Algorithms-I, 1991.

4. Deb K., *Genetic Algorithms for Engineering Design*, From Course material of Genetic Algorithms for Engineering Design organized by KanGAL, IIT Kanpur, pp. 8-20, November 1997.

5. Deb K., *Optimization for Engineering Design Algorithms and Examples*, Prentice Hall of India, 1996.

6. Michalewicz Z., *Genetic Algorithms + Data Structures = Evolutionary Programs*, Springer-Verlag, 1992.

7. Ismail H. S., Hon K. K. B., *The Nesting of two Dimensional Shapes using Genetic Algorithms*, Proceedings Instn Mechanical Engineers, pp. 115-124, Vol 209.

8. Ono T., Watanbe G., *Genetic Algorithms for Optimal Cutting*, From Course material of Genetic Algorithms for Engineering Design organised by KanGAL, IIT Kanpur, pp. 311-326, November 1997.

9. Chang H. D., Taatoglu M. Y., *An Approach to Two Dimensional Cutting Stock Problems*, INT. J. PROD. RES. , VOL. 25, NO. 2, PP. 175-190, 1987.

10. Wang P. Y., *Two Algorithms for Constrained Two Dimensional Cutting Stock Problems*, Operations Research, Vol. 31, No. 3, May-June 1983.

11. Dietrich R. D., Yakowitz S. J., *A Rule Based Approach to the Trim Loss Problem*, INT. J. PROD. Res. Vol. 29, No. 2, 401-415, 1991.

12. Yuzu C., Lujan L., Wang W., Jianwen S., *An Expert System for Automatic Allocation of 2-D Irregular Shapes*, Expert Systems in CAD, PROD. IFIP WORKING CONF., PP 407-422, FEBRUARY, 1987.

13. Jain P., Fenyes P., Richter R., *Optimal Blank Nesting using Simulated Annealing*, General Motors Research Laboratories, Warren, Michigan.

14. Dowsland K. A., Dowsland W. B., *Solution Approaches to Irregular Nesting Problems*, European Journal of Operational Research, pp. 506-521, 1995.

15. Srinivas M., Patnaik L. M., *Genetic Algorithms : A Survey*, COMPUTER, pp. 17-26, June 1994.

16. Adamowicz M., Albano A., *A Solution of the Rectangular Cutting Stock Problem*, IEEE Transactions on System, Man and Cybernatics, Vol. SMC-6, No. 4, April 1976.

17. Filho J. L. R., Treleaven P. C., Alippi C., *Genetic Algorithms Programming Environments*, COMPUTER, pp. 28-43, June 1994.

18. Kirkpatrick S., Gelatt C. D. Jr., Vecchi M. P., *Optimization by simulated annealing* , SCIENCE, Vol. 220, No. 4598, pp. 671-680, May 1983.

19. McLaughlin M. P., *Simulated Annealing*, Dr. Dobb's Journal, September 1989, pp. 26-37.

20. Nee A. Y. C., Seow K. W., Long S. L., *Designing Algorithms for Nesting Irregular Shapes with and without Boundary Constraints*, Annals of the CIRP, Vol. 35/1/1986, pp. 107-110.

21. Reeves C., *Modern Heuristic Technique for Combinatorial Problems*, Orient Longman.

22. Galante M., *Genetic Algorithms as an Approach to Optimize Real-World Trusses*, International Journal for Numerical Methods in Engineering, Vol. 39, 1996.

23. Rich E, Knight K, *Artificial Intelligence*, Tata McGraw-Hill.