THESIS
/

LIBRARY
Michigan State
University

This is to certify that the

thesis entitled

NESTING OF IRREGULAR SHAPES USING A PARALLEL GENETIC ALGORITHM AND FEATURE MATCHING

presented by

Anand Uday

has been accepted towards fulfillment
of the requirements for

MS degree in Mechanical Engineering

Erik D Goodman

Major professor

Date December 13, 2001

O-7639 *MSU is an Affirmative Action/Equal Opportunity Institution*

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.
MAY BE RECALLED with earlier due date if requested.

DATE DUE	DATE DUE	DATE DUE
OCT 00'7 92 80'3		

6/01 c:/CIRC/DateDue.p65-p.15

Nesting of Irregular Shapes Using a Parallel Genetic Algorithm and Feature Matching

By

ANAND UDAY

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

2001

ABSTRACT

Nesting of Irregular Shapes Using a Parallel Genetic Algorithm and Feature Matching

By

ANAND UDAY

The problem of finding a dense packing of a set of two-dimensional polygonal shapes within another larger two-dimensional polygon is called nesting. This problem is widely encountered in companies fabricating metal parts, leather cutting industry and in the textile industry - in short, where the material is costly and scrap needs to be minimized. This thesis describes a new approach to nesting problems. It is a hybrid approach, which uses a parallel genetic algorithms and shape information in the form of feature matching. Here, the shape information has been used to do the local search and a parallel genetic algorithm has been employed for the global search. Various experiments were performed to determine a good set of parameters for use in feature matching and the parallel genetic algorithm. To reduce the chances of premature convergence of the parallel genetic algorithm, different topologies for communication among subpopulations and different migration schemes were tried. A good choice of communication patterns seems to maintain balance between the frequency of migration and the degree of interconnectivity among the subpopulations. The test problems show this approach to work well for the nesting problem, where the search domain is often very large.

This research is dedicated to the Genetic Algorithm Research and Applications Group

ACKNOWLEDGEMENTS

I wish to take this opportunity to express my gratitude to my advisor Dr. Erik D. Goodman. This research work would not have been possible without his support, encouragement, guidance and time. He was always patient in understanding the problems I encountered during the research work and helped me to solve those. The constructive suggestions he made during the reading of the manuscript of this thesis helped in improving it to a great deal.

My special thanks go to Dr. Ronald Rosenberg and Dr. Farhang Pourboghrat for serving on my committee. I would like to appreciate the help given by my fellow researchers at MSU GARAGe. They always provided me with useful technical tips and encouragement. I also want to acknowledge Rakesh Kumar's help in providing me with the data from ship building industry.

I would like to recognize the support and encouragement provided by my friends. Finally, a very special thanks to my parents, sisters and brothers for their love and moral support.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Introduction

Layout and cutting problems are important in many industries, as they involve the optimal use of valuable raw material. Problems of optimal arrangement of 2-D pieces to be cut from an initial piece of stock material are called nesting problems. They are also called by other names such as cutting stock problems, layout problems and packing problems. There are many varieties of problems, depending on the shapes of the pieces, constraints on their orientations, etc. The problem to be addressed in this report can be stated as follows: given a rectangular piece of stock of a specified width and indefinite length, find the optimal arrangement of a given set of polygonal part shapes onto that stock such that:

- None of the part overlaps any other.
- All are contained within the boundary of the stock piece.
- The length of the stock piece used is minimized (optimality condition for this problem)

In this case, there is no constraint on the orientation of the part shapes, but they may not be *turned over*. It should be also be noted that there is no constraint on the shape (for example convexity) of the parts to be nested. Figure 1.1 illustrates the problem to be addressed in this report.

1.2 Application of Nesting

The nesting problem is encountered in various industries. Some of them are listed below:

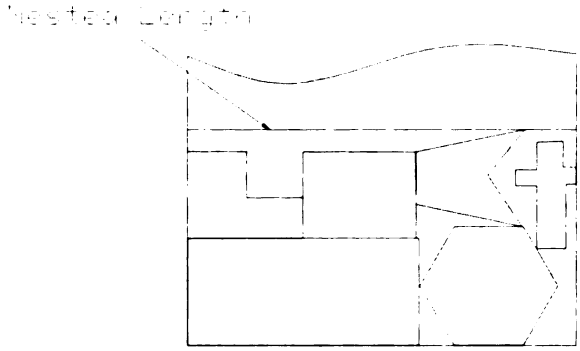

Figure 1.1: An example showing the different shapes placed onto a bigger sheet

1. Ship Body Building Industry
2. Automobile Body Building Industry
3. Aircraft Body Building Industry
4. Textile Industry
5. Leather Industry
6. Glass Cutting Industry
7. Wood Industry
8. Heavy Equipment Manufacturing Industry

1.3 Types of Nesting

In recent years, a number of researchers have investigated the problem of nesting of irregular shapes, and the approaches can be grouped in three categories:

manual, semiautomatic and automatic. In the first case, the draftsman makes use of a computer with graphics input-output devices, capable of manipulating (rotating, translating, etc.) pieces on a working display. In the second category, the system proposes a tentative solution and then interactive improvements are allowed by a conversational display unit, using a special set a of commands to operate on the solution process. The third one is totally automatic, where given the input, the system gives the final output. Automation improves material requirement estimates and substantially reduces man hours required for part layout. This leads to increased productivity, reduced material handling and tracking, and thereby helps to reduce the overall project costs. Research into the automation of the 2-D nesting problem can be divided into three broad categories. The first involves allocation of rectangular shapes onto rectangular stock material. The second, more general, problem, entails nesting irregular profiles onto a sheet of rectangular shape. The third one is the most generalized form, which is nesting of irregular profiles onto resources of arbitrary shapes.

1.3.1 Approaches to Automated Nesting

The problem of automated nesting has been tackled in various ways. They can be broadly classified as follows:

- Rule-based heuristic approach: In a rule-based heuristic approach, a set of rules is designed to try to take advantage of some characteristics of the shapes of the parts, placing earliest those with certain characteristics, packing together parts with certain matching features, etc. They basically try to mimic the manual nesting.

- Stochastic approaches: Stochastic approaches such as genetic algorithms or simulated annealing typically use little information about part shapes, instead

use only simple packing rules and rely on the stochastic algorithm to vary the order in which these rules are applied to the parts to be nested.

The approach used in this thesis is a hybrid one. It relies on a powerful feature-matching heuristic capable of generating fairly good packing even without a genetic algorithm. It uses part shape features to determine the exact placement and orientation of the parts, here augmented by a genetic algorithm that determines the sequence in which they are nested (now sometimes together called a *memetic algorithm*). In addition a parallel GA has been employed to make the search both more global and more efficient.

1.4 Organization of Report

This report is divided into five chapters. The next chapter briefly reviews the literature on this problem. The third chapter describes in detail the algorithm implemented. The fourth chapter presents the results obtained. And finally the fifth chapter discusses the scope of further work and concludes the report.

Chapter 2: Literature Review

2.1 Introduction

As mentioned in the previous chapter, most of the proposed nesting methods can be broadly categorized into two categories. One of them predominantly relies on exploiting the shape information of the parts to be nested, and the other one tries to arrive at the solution using stochastic evolution of potential solutions. In this chapter an attempt is made to briefly cover the literature available on both the approaches.

2.2 Review of Algorithms Using Shape Approximation and Heuristics

One of the simplest and earliest methods of nesting irregular profiles is approximating either a part or a group of parts using a rectangular enclosure. In this case, Minimum Enclosing Rectangles (MER's) of each part is constructed and then they are nested by applying algorithms available for nesting rectangular shapes. A considerable amount of research work has been published using this approach.

One of them is being described in brief below. Nee *et al.* in 1986 suggested an algorithm for approximating the given shapes by means of lines, essentially trying to find the Minimum Enclosing Polygon(MEP). After the MEP has been found, the MER for that MEP is constructed. Depending on the shapes of the MEP's, pair wise clustering is also seeked. Once all the shapes have been considered by the algorithm described above, a rectangular nesting algorithm is invoked. Here, all the rectangles are first arranged in descending order of areas. The largest rectangle is placed in the lower left corner of the stock sheet and the pivot points are created. The next

rectangle's are then placed at each of the pivot points generated, both length and breadth-wise, to check if they intersect with any of the existing rectangles. Only the non-intersecting rectangles are considered, and the minimum enclosing area of both the rectangles is computed. After selecting the best position, the shape inside the rectangular module is further shifted in both horizontal and vertical directions. New pivot points are now defined and the old ones are deleted. This process is repeated until every rectangle has been packed. Going in steps like the above algorithm proves to be simple and natural, but the wastage associated with approximation of highly irregular shape to MER might lead to an inefficient final layout.

Earlier in 1975, Freeman and Shapiro, and in 1976, Adamowicz and Albano had approached the nesting algorithm using the same technique of approximating the given parts with MER's and then packing the MER's. Their solutions also suffered from the above-mentioned problem.

Several attempts have also been made to solve this problem using local greedy heuristics. The local greedy heuristics were essentially based on the use of a directional placement policy, which involved placing the parts at hand on the lowest and left-most available vertex.

In 1993, Dowsand and Dowsland used a random left-most placement policy and allowed for pieces to jump over pieces to fill gaps. This was done in order to allow the smaller pieces to fill in the gaps created by placement of bigger pieces. In 1980, Albano and Sapuppo proposed another algorithm using lowest leftmost policy, which decided the next part to be placed by evaluating the potential waste due to the placement of the part at hand.

In 1996, Lamousin et al. proposed an algorithm that modifies Albano and Sapuppo's algorithm, using the concept of No Fit Polygon (NFP) for part placement. Lamousin and Waggenspack (1996) also proposed another algorithm that uses features of the profile. This algorithm tries to find and match

complementary features of the profile and the remaining area of stock.

Some attempts have been also made to solve this problem using Monte Carlo algorithms. Bohme and Graham (1979) tried this method and suggested that approximately 2000 such random trials are usually required to get satisfactory results. The best solution was then fine tuned by fine random perturbation. In the next section the stochastic approaches for solving this problem are described.

2.3 Review of Algorithms Using Stochastic Approaches

Besides the above-mentioned deterministic approaches, the probabilistic and evolutionary techniques have also been successfully employed to solve the nesting problem. G-C Han and S-J Na (1996) used a two-stage method with a neural-network-based heuristic for generating an acceptable initial layout, and a simulated annealing algorithm for fine-tuning the solution.

In 1995, Ismail and Han proposed a genetic-algorithm based solution to this problem. They generated a set of initial random layouts as their first generation chromosomes. The layouts were allowed to have parts overlapping each other. They constructed an objective function that tried to minimize the total area needed to place all the parts. This objective function included a penalty term which took into account the overlaps between two parts.

In 1998, Jain and Gea proposed a solution based on a genetic algorithm by using a 2-D representation for a chromosome. The 2-D representation was arrived by dividing the stock into finite equal sized square cells. The value of cell was either 1 or 0 depending on whether it was occupied by any part or not. The genetic operator's were modified accordingly to perform on the 2-D chromosome.

In 1999, Babu and Babu came up with a genetic algorithm approach which

aimed at finding an optimum sequence of placing the parts on the sheet. In 2000, Sha and Kumar came up with a representation that encoded the sequence and orientation of the parts on a 2-D chromosome and modified the genetic operator's to deal with that form.

2.4 Conclusion

The approaches using the shape information were unable to perform very well, when the shapes to be nested turned out to be highly irregular. And on the other hand the probabilistic approaches, due to their evolutionary nature and lack of any in-built heuristic, proved to be of high time complexity. The approach proposed in this work uses a parallel genetic algorithm wedded with a powerful heuristic to solve the problem more effectively. The next chapter describes the algorithm used in the present work. It builds the background knowledge required to understand the approach as well as integrating everything to present the algorithm in totality.

Chapter 3: Nesting Using a Parallel Genetic Algorithm and Feature Matching

3.1 Introduction

This chapter describes the approach taken to address the nesting problem. As mentioned before, the approach is a hybrid. The idea comes from the observation that in a typical packing problem the order in which the parts are placed plays a very determining role in the quality of the final solution. Figure 3.1, which illustrates this concept, we can see that using the same set of parts, and same set of rules to pack them requires different sheet lengths if the order of parts are different. The algorithm utilizes shape information in the form of feature matching, and a genetic algorithms is used to generate the sequence in which the parts are to be placed on the sheet.

The sequence generated by the GA module is fed to the feature matching module, which in turn places the parts according to a predefined set of rules and returns the length of sheet used to the GA module for further iterations. The iterations are carried on until a good solution is obtained or a predetermined amount of effort has been put in. Figure 3.2 illustrates this concept graphically.

Before going any further, I will briefly introduce the concept of a genetic algorithm.

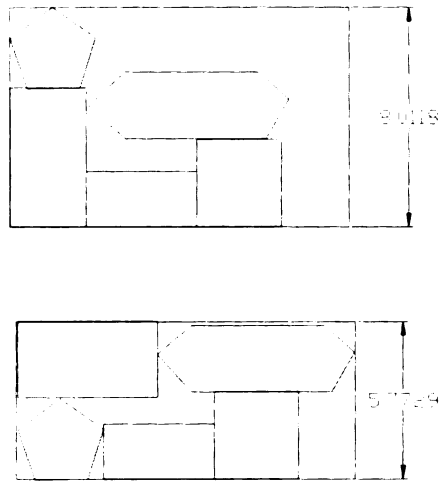

Figure 3.1: Effect of Part Sequence in packing problem

3.2 Genetic Algorithms (GA's)

Genetic algorithms are optimization procedures that operate by mimicking nature's evolutionary processes. The principle of a genetic algorithm is based on the Darwinian notion of "survival of the fittest". GA's were first introduced by John Holland in 1960 and described in his landmark book called *Adaptation in Natural and Artificial Systems"* (Holland 1975). In GA's, an initial population of solutions is randomly generated. Each member of this population (called a chromosome), is a coded representation of the design attributes that represents a potential solution for the problem to be solved. The initial group of solutions (zeroth generation) undergoes operations like mutation and crossover (similar to those of biological processes) to generate a new set of solutions (next generation). The idea behind mutation is to randomly alter one or more attribute of the individual in search of a better solution. Crossover is done in order to combine the attributes of two individuals and carry them to the next generation. The process of creating a new generation also involves selection, which is biased to give the fitter individuals more chances to contribute in the formation of the next generation. This process is

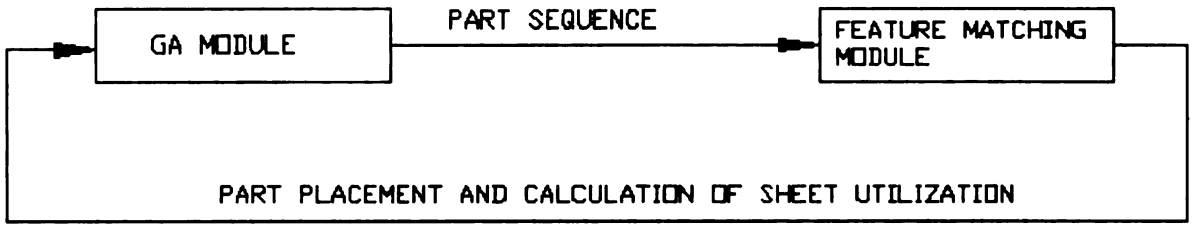

Figure 3.2: Flow of the Algorithm

continued until a 'good' solution is achieved, or a predetermined number of generations has been completed. Figure 3.3 illustrates steps involved in GA's graphically.

3.2.1 Parallel Genetic Algorithms (PGA)

Genetic algorithms run can be run on a single processors, but they have been proven to be highly parallelizable, capable of being run in a cluster of computers. Besides this, although GAs can be made resistant to premature convergence, they are not immune. One technique to reduce the likelihood of premature convergence and overcome the problem of speed is parallelization of the GA, by the use of

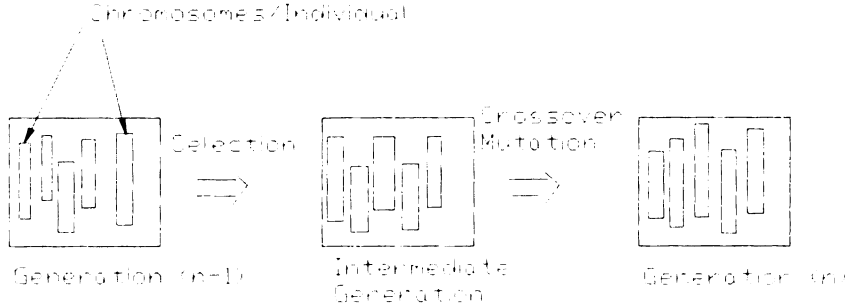

Figure 3.3: The generational evolution of Genetic Algorithms

multiple subpopulations on multiple processors. The two most commonly used kinds of parallel GA's are: fine-grain GA's, and coarse-grain GA's. In fine-grain GA's (fgGA's), individuals are arranged in some tessellation with an individual on each processor. In this model, the individuals are allowed to interact only with their immediate neighbors. Implementation of this model requires the processor topology to be of a specified manner and high connectivity among the processors. In coarse-grain GA's (cgGA's), each node is assigned a particular subpopulation performing a single population GA. At certain intervals, some individuals might migrate from one subpopulation to another. The migration rule is usually predetermined. The next few sections describe the various components involved in the implementation of a parallel genetic algorithm for the nesting problem.

Population Representation

In a *classical GA*, a binary string representation is often used. However, for sequencing and other combinatorial problems, chromosomes often represent an ordering of entities, specified simply as a permutation of the integers 0, 1, .., N-1). In this problem the chromosome has been represented as an array of integers, where each element of the array corresponds to a particular part index,
for example: 3 4 5 9 7 10 8 0 2 1

In the above example, the chromosome would be used for solving a problem having 10 parts to be nested. The chromosome would be interpreted as the sequence in which the parts are to be placed on the sheet. In the above case, this means that part number 3 would be placed first, then part number 4, then part number 5 and so on.

Crossover Operator

For this representation, many different crossover operators are suitable: partially matched crossover (PMX), uniform order based crossover (UOBX), order based crossover (OBX), and cycle crossover (CX) (Davis, 1991), (Goldberg , 1988). Each of the crossover operators are described in detail in appendix A.

Mutation Operator

The mutation operators that are designed to operate on permutation problem are called the swap and the scramble mutation operators (Davis, 1991; Goldberg, 1988). Appendix B describes both the mutation operators.

Crowding Factor and Incest Reduction

To reduce the chances of premature convergence, a DeJong-style crowding factor was used (Goldberg, 1988). It helps in allowing several distinct groups of individuals to develop and persist in the population. This technique is useful in exploring multi-modal problems. In addition, the mechanism of incest reduction (Goodman, 1994) reduces the proportion of crossovers performed between very similar chromosomes. Further, it also helps to maintain genetic diversity, thus helping avoid premature convergence.

Elitism

An Elitism mechanism was used to insure that at least one copy of the current generations best individual appears in the next generation.

Migration

The migration policy used between the subpopulation is critical in balancing the co-evolution of the sub populations. It affects the selection pressure and thus

convergence time (cantu-paz, 2000). The following are different migration policies which are commonly used:

1. The sending subpopulation sends a random individual, and that individual replaces a random individual in the receiving subpopulation.

2. The sending subpopulation sends a random individual, and that individual replaces the best individual in the receiving subpopulation.

3. The sending subpopulation sends its best individual, and that replaces a random individual in the receiving subpopulation.

4. The sending subpopulation sends its best individual, and that individual replaces the best individual in the receiving subpopulation.

In this research the sending subpopulation sends the best individual, which in turn replaces a random individual in the receiving subpopulation. This policy was used in order to ensure that all the subpopulations benefit from the discovery of a good solution in a limited fashion. In addition, the migration rate and the number of migrants are also very important. Here, depending on the complexity of the problem, one tenth of the individuals were migrated every five to ten generation.

Subpopulation Layout (Topology)

The spatial arrangement of subpopulation and their interconnectivity is the key to successful implementation of cgGA's. The topology of a cgGA should be designed keeping in mind that, a discovery of a good solution should benefit the entire search process, and also, at the same time, it should not lead the search towards premature convergence because of one subpopulation influencing the others to the extent of overshadowing their own exploration. The four different layouts that were considered in this research are as follows:

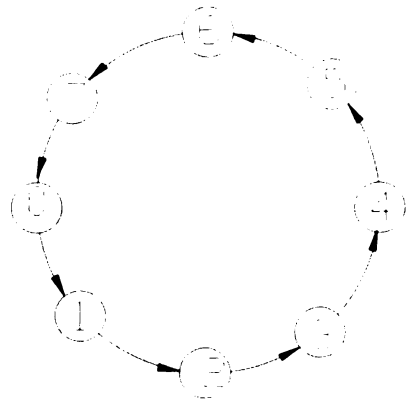

Figure 3.4: Eight subpopulation with one way communication

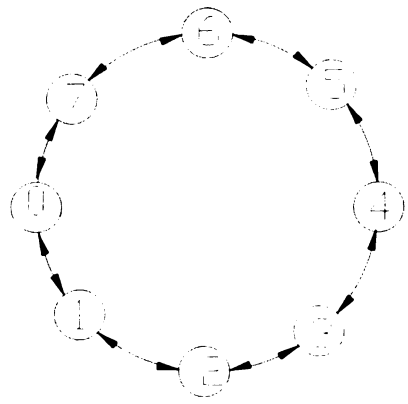

Figure 3.5: Eight subpopulation each having two neighbor

1. Ring layout with one neighbor: Eight different subpopulation laid out in a circular manner with one neighbor each (Fig 3.4).

2. Ring layout with two neighbor Eight different subpopulations laid out in a circular manner with two neighbors each. (fig 3.5).

3. Grid Layout with directional connectivity: Twenty different subpopulations laid out in a grid with partially directional communication (Fig 3.6).

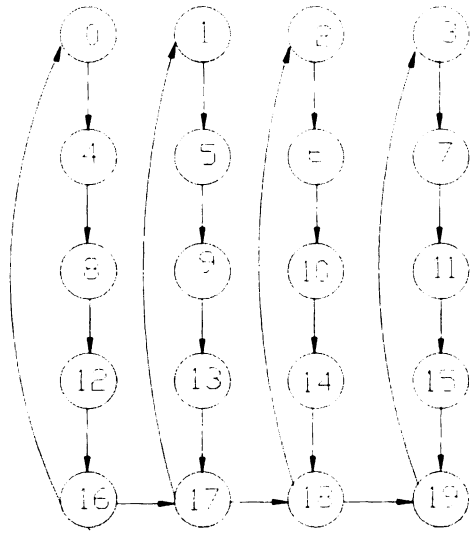

Figure 3.6: Grid Topology with partially directional communication

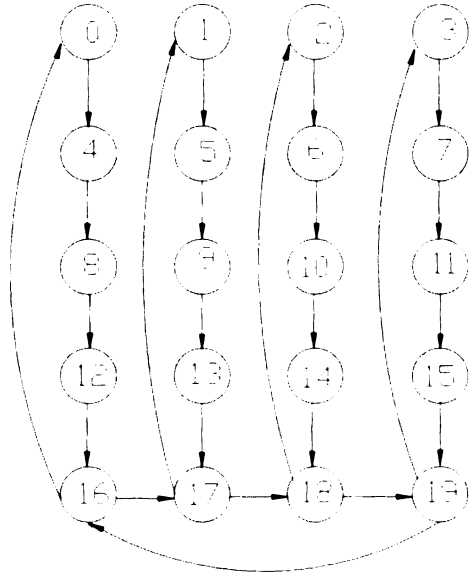

Figure 3.7: Grid topology with full communication

4. Grid layout with full connectivity: Twenty different subpopulations laid out in a grid with all subpopulations indirectly connected to each other (Fig 3.7).

The ring topologies are the standard topologies that have been used in previous research work with PGA's. But the two grid topologies are the non-standard

topologies, and were designed because of the inability of ring topologies to perform well on the complicated nesting problems. The effect of these topologies is discussed in detail in the next chapter.

Evaluation Function

The problem aims at minimizing the length used of a fixed-width piece of rectangular stock. However, in preliminary studies, it was observed that nesting larger parts first often yields a better solution. In order to speed the search and exploit this, in some of our runs, a bias term was added to the fitness function that slightly favored nesting in which larger (area) parts were placed first. But, the effect of this term was not found to be large, and its use was abandoned in the later runs.

Selection Method

Selection is an important component of GAs. According to some schemes, the fitter members of the population are chosen to be taken to the intermediate generation . Selection makes a large contribution in determining the final quality of solution and the time taken to achieve it. For this problem the *stochastic universal sampling* and *tournament selection* were considered for selection. Appendix C explains these methods in detail. The selection was done using linear scaling of fitness value (Goldberg, 1988) except when using the tournament selection.

3.3 Feature Matching and Placement Policy

In this hybrid (or memetic) algorithm, shape information is used to effectively match complementary features on the parts and the stock. In this case, building on the earlier work of another graduate student (Debnath, 1997), a feature was defined to be an instance of two adjacent edges on a polygon. The data defining this type of

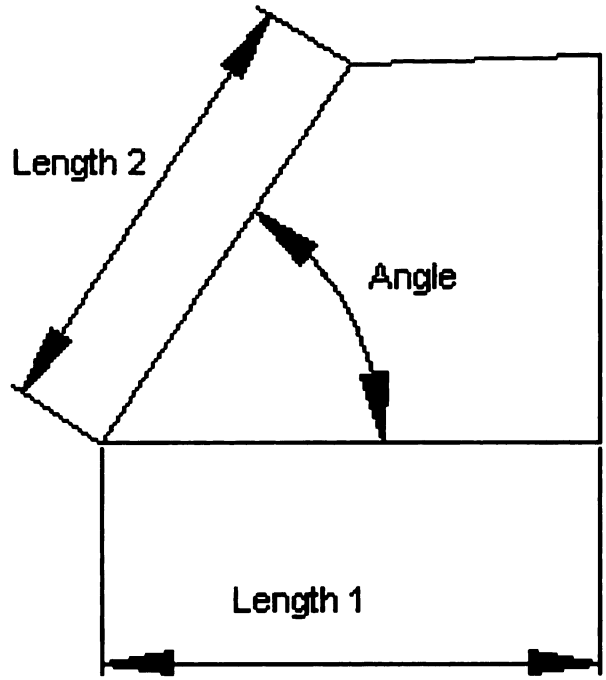

Figure 3.8: Feature Information

feature are the lengths of the two adjacent edges and the internal angle between these edges. Figure 3.8 shows an example of such a feature.

3.3.1 Placement Policy

Given the next part to be nested, the algorithm determines what position and orientation is best for the part vis-a-vis the current state of the stock. At any point, the system tracks a stock profile, a polyline comprised of portions of edges of stock and parts, and that includes all currently open area in the stock as nested to date. Candidates for features at which to nest subsequent parts are located on this profile. The profile will grow to include many small, closed areas in which no additional parts can be nested, and the algorithm, after numerous attempts to nest subsequent parts in such an area, will eventually mark the points in that area as *bad points*, and will refrain from trying to nest more parts there. Figure 3.9 shows typical packing in progress. The packing heuristic first selects the vertex on the stock profile (ignoring bad interior points) with the lowest y-co-ordinate. If more than one vertex

has the same y-co-ordinate, the vertex with the smallest x-co-ordinate is selected. This selection is based on the placement policy of lowest and if necessary leftmost. The heuristic forms a target feature on the stock, and iterates through all the features of the part at hand. To each of the iterations through the part features it assigns a particular score. The orientation yielding the highest score is retained for the final placement. This score is calculated using the following parameters:

3.3.2 Left Shadow Area

Since the placement policy was selected to fill up the stock from left to right, any closed-off areas to the left of the stock would be unfavorable for the final solution. See Figure 3.10

3.3.3 Bottom Shadow Area

Since the placement policy was also set to fill up the stock from bottom to top, any closed-off areas towards the bottom of the part would again be unfavorable for the solution. See Figure 3.10

3.3.4 Contact Length of the Feature

To effectively exploit a corner feature, it is necessary to calculate the contribution due to the feature itself. To ensure good local packing, we wanted the contact length between the part at hand and existing stock profile to be maximized. Further, in order to yield comparable units in the scoring function, the value of this measure is squared. See Figure 3.10

The scoring function used is thus:

$$= a * (Left\ Shadow\ Area) + b * (Bottom\ Shadow\ Area) + c * (Contact\ Length)^2$$

$$where\ a, b < 0\ and\ c > 0$$

The next chapter discusses the experimentation conducted and results obtained.

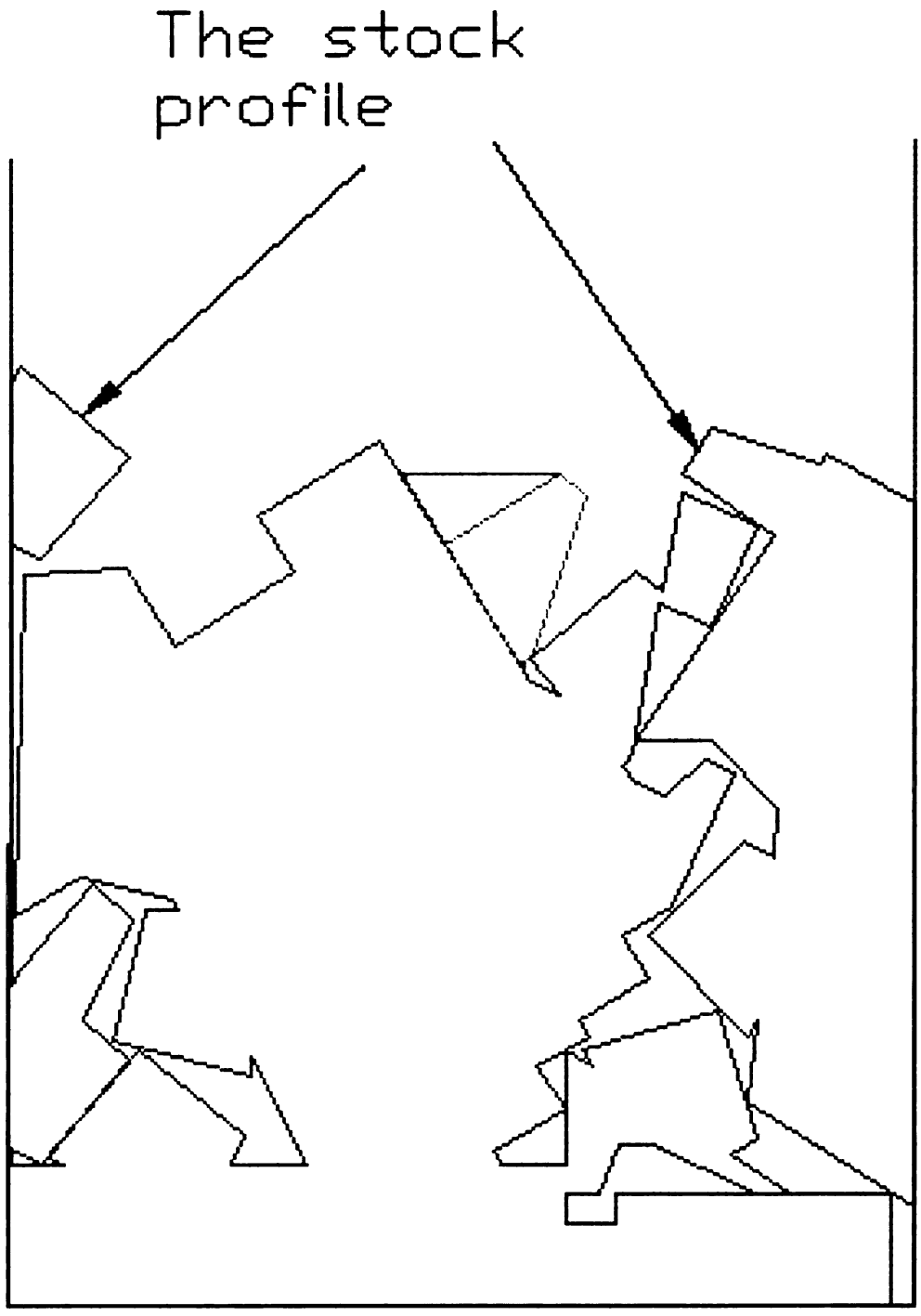

Figure 3.9: Developing Stock Profile

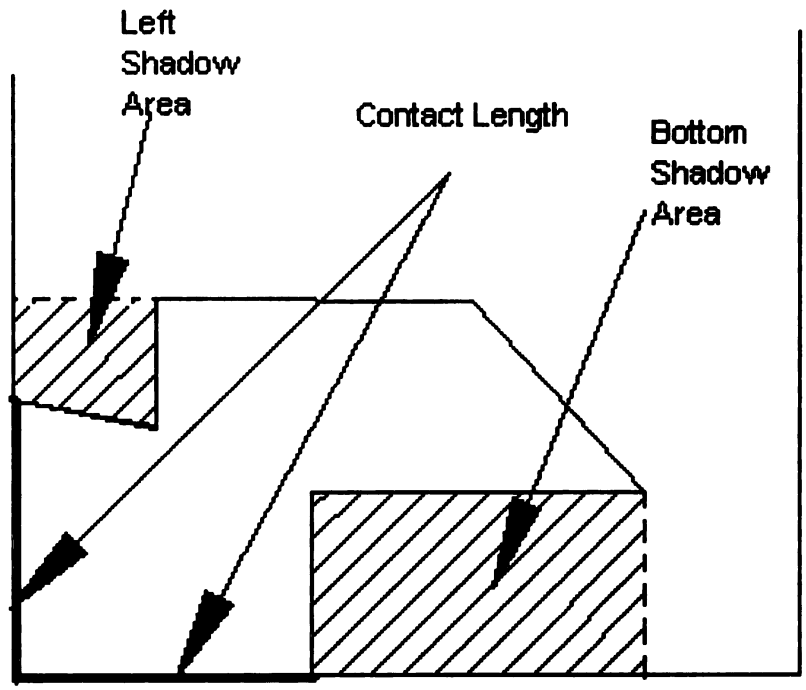

Figure 3.10: Sample part and stock illustrating some packing measures

Chapter 4: Results and Discussions

The algorithm was tried on various problems set with varying parameters of GAs and the feature matching coefficients. In the feature matching module, the coefficients of the scoring function were varied such that:

$$a, b \epsilon \{-1, -2, -3, -4\} \; and \; c \epsilon \{1, 2, 3, 4\}$$

After initial experimentation, it was found that a good set of values for (a, b, c) is (-1,-1,3). Here the shadow coefficients are given equal negative values and the contact length co-efficient takes a high positive value. This made the local search more biased towards placing parts in orientations that maximize their contact with the ongoing stock profile.

For the GA module, the following are the scope of variations which were considered or tested:

- Crossover operators : Among the four different crossover operators suitable for permutation type problems in GALOPPS the PMX and UOBX were the most logical candidates for the nesting problem. Both of them were tested and it was found that UOBX performed slightly better than the PMX operator. Considering the nature of this problem and the manner in which UOBX works, the UOBX operator was chosen for further use. There seems to be a higher probability that meaningful building blocks are preserved, by UOBX, further supporting the test results.

- Mutation operators : Between swap and scramble mutation, the swap mutation makes a better choice in this case, considering the fact that it does not have as destructive an effect as the scramble mutation. It was used for all the experiments.

- Selection methods : Both tournament and stochastic selection methods performed approximately equally well for this problem. In the experiments where it was desired to increase the rate of convergence, the tournament selection was used with size of the tournament between 3 and 5.

- Topology in which the subpopulations are arranged : It was observed that the grid layouts performed better than the ring layouts. In the case of ring layouts, the improvement in solution used to stop earlier than with the grid layouts. This may be due to the fact that the grid topology helped in maintaining genetic diversity among individuals for more generations and thus gave the subpopulations more time to evolve. In addition, the fully connected grid layout was found to yield more consistent results than the partially directional grid layout. This was expected, as in the case of a fully connected layout the finding of a good individual benefits all the subpopulations, which is not true with the directional connectivity.

The GA module was implemented using GALOPPS version 3.2.2 (Goodman, 1996), a GNU-licensed freeware developed at MSU's GARAGe.

The sample problems for, which the results are shown have been derived from the existing literature. However, since in most of the literature, the part geometries are not published, a definitive comparison cannot be made.

4.1 Problem 1

This problem involves nesting of rectangular parts on a rectangular stock sheet. It is a artificial problem published in (Burke and Kendall 1999). Figure 4.1 shows the part and the Figure 4.2 shows the solution obtained.

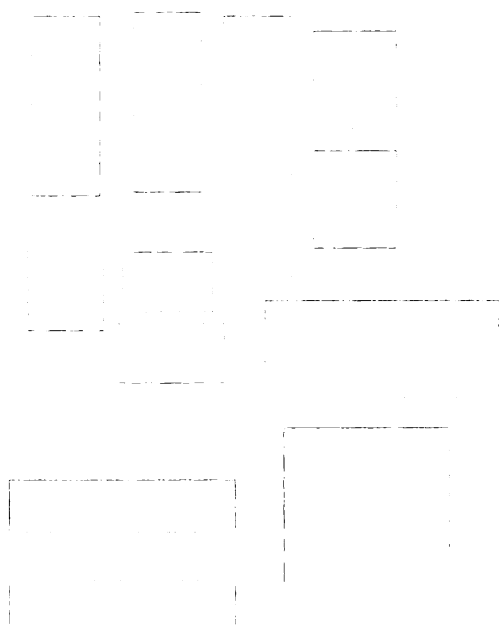

Figure 4.1: Parts for Problem 1 (Total Part Area = 11112 sq. units)

Number of Parts	13
Total Area of Parts	11112.00 sq. units
Total Rectangular Area Required	11200 sq units
Width of Stock Sheet	80 units
Percentage Utilization	99.21
Percentage Utilization as Reported	99.21
Generations Required	425
Topology Used	Fully Connected
Crossover Operator	UOBX
Mutation Operator	Swap
Selection	tournament selection
Stopping Criterion	500 generations

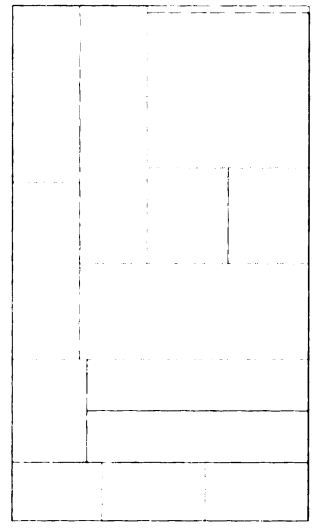

Figure 4.2: Solution obtained for Problem 1 (Percentage Utilization = 99.21)

4.2 Problem 2

This problem involves nesting of irregular shapes onto the rectangular stock material. This is an artificial problem published in (Jakobs, 1996). Figure 4.3 shows the parts and the Figure 4.4 shows the results obtained.

Figure 4.3: Parts for Problem 2 (Total Part Area = 392 sq units)

Number of Parts	25
Total Area of Parts	392 sq. units
Total Rectangular Area Required	502.37 q units
Width of Stock Sheet	40 units
Percentage Utilization	78.03
Percentage Utilization as Reported	65.33
Generations Required	192
Topology Used	Fully Connected
Crossover Operator	UOBX
Mutation Operator	Swap
Selection	Stochastic Universal Sampling
Stopping Criterion	500 generations

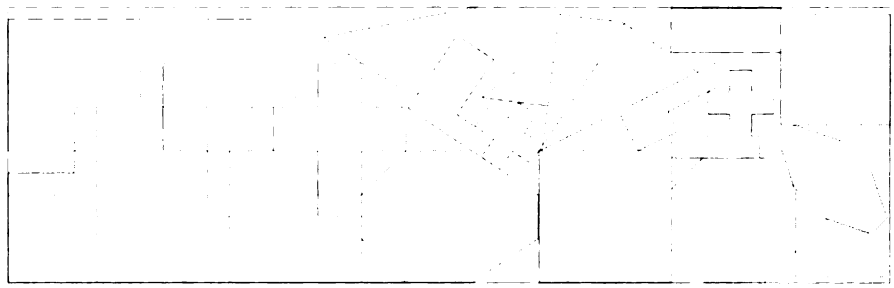

Figure 4.4: Solution Obtained for Problem 2 (Percentage Utilization = 78.03)

4.3 Problem 3

This is a jigsaw problem and was constructed from the results published in (Dighe and Jakiela, 1996). This problem was chosen to test the effectiveness of the feature matching algorithm. Figure 4.5 shows the parts used in this problem; Figure 4.6 shows the solution obtained. The details of the solution are as follows:

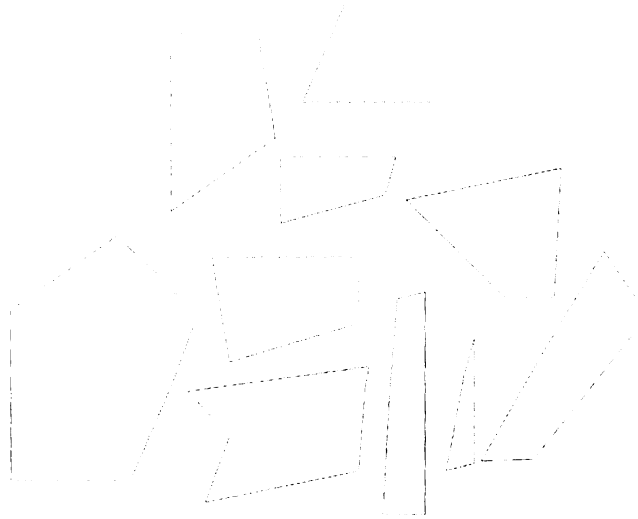

Figure 4.5: Parts for Problem 3 (Total Parts Area = 10000 sq. units)

Number of Parts	10
Total Area of Parts	10000 sq. units
Total Rectangular Area Required	10000 sq units
Width of Stock Sheet	100 units
Percentage Utilization	100.00
Percentage Utilization as Reported	100.00
Generations Required	94
Topology Used	Fully Connected
CrossOver Operator	UOBX
Mutation Operator	Swap
Selection	Tournament Selection
Stopping Criterion	500 generations

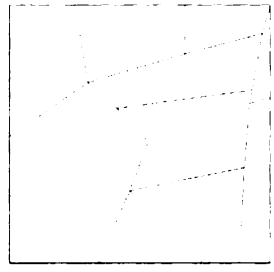

Figure 4.6: Solution Obtained for problem 3 (Percentage Utilization = 100.00)

4.4 Effect of variations in the PGA module

4.4.1 Effect of different topologies

In this section a comparison has been made between the various parallel genetic algorithm topologies. The parts to be nested have been taken from the shipbuilding industry. Figure 4.7 shows the parts used for the experimentation. These parts are characterized by their complex geometries and the large variation in their sizes. Figure 4.8 to fig 4.11 shows the result obtained using different topologies. The details of the solution obtained are as follows:

Number of Parts	10
Total Area of Parts	3748.75 sq. units
Width of Stock Sheet	65 units
Crossover Operator	UOBX
Mutation Operator	Swap
Selection	Stochastic Universal Sampling
Stopping Criterion	No improvement for about 100 generation or max of 500 generations

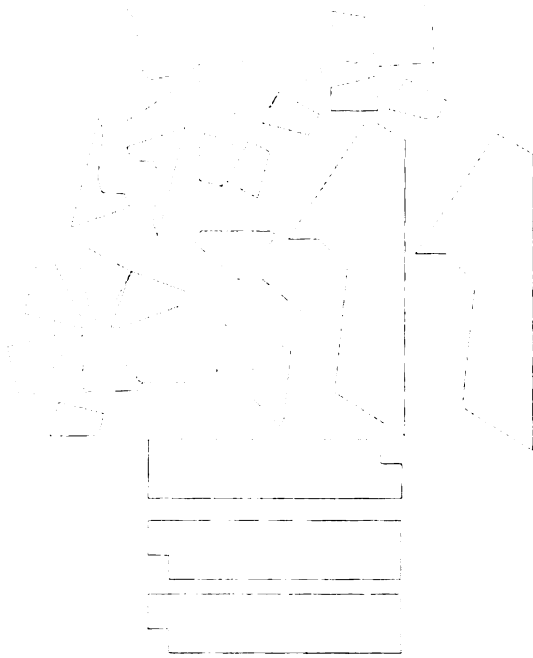

Figure 4.7: Parts of Problem 4 (Total Part area = 3748.75 sq. units)

Table 4.1: Results Obtained for Various Topologies (best of all the runs performed)

Topology Used	Percentage Utilization	Generations Needed
Ring Topology One Neighbor	82.07	252
Ring Topology Two Neighbor	82.31	310
Grid Topology Partial Connection	84.77	394
Grid Topology Full Connection	85.33	442

Table 4.2: Results Obtained for Various Topologies (average of the three test runs)

Topology Used	Percentage Utilization	Generations Needed
Ring Topology One Neighbor	81.42	270
Ring Topology Two Neighbor	82.00	290
Grid Topology Partial Connection	83.07	370
Grid Topology Full Connection	84.50	389

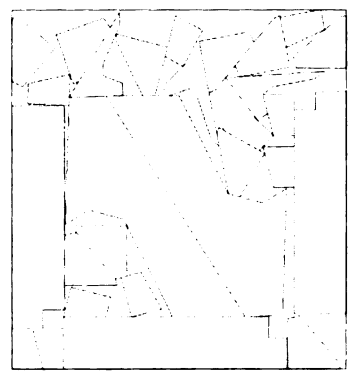

Figure 4.8: Solution to Problem 4 obtained using ring topology with one neighbor (Percentage Utilization = 82.07)

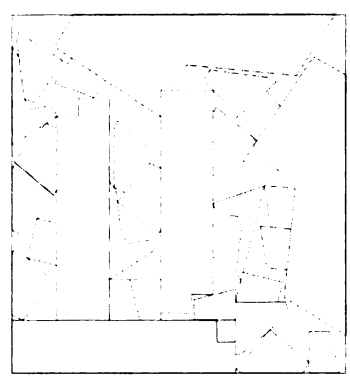

Figure 4.9: Solution to Problem 4 obtained using ring topology with two neighbors each (Percentage Utilization = 82.31)

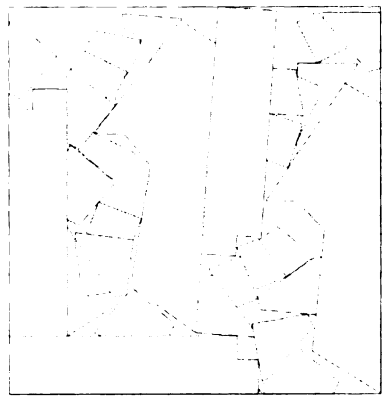

Figure 4.10: Solution to Problem 4 obtained using the grid topology having partially directional communication (Percentage Utilization = 84.77)

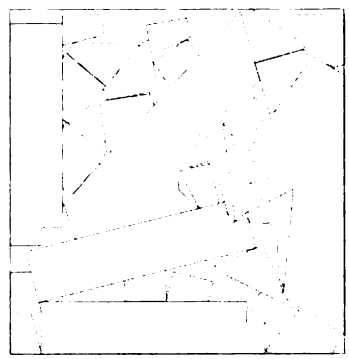

Figure 4.11: Solution to Problem 4 obtained using the grid topology having full directional communication (Percentage Utilization = 85.33)

Table 4.3: Results obtained for PMX and UOBX operator (best of the three test runs)

Operator	Percentage Utilization	Generations Needed
PMX	83.54	245
UOBX	83.67	190

Table 4.4: Results obtained for PMX and UOBX operator(average of the three test runs)

Operator	Percentage Utilization	Generations Needed
PMX	82.88	272
UOBX	83.03	210

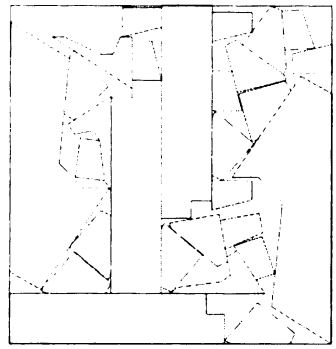

Figure 4.12: Solution to Problem 4 obtained using the UOBX operator (Percentage Utilization = 83.67)

4.4.2 Effect of PMX and UOBX operators

In this case, problem 4 was solved three times, with maximum generations set at 300. It was found that both the operators were able to generate sequences giving almost equally good packings but in the case of UOBX, the number of generations required was smaller. Figure 4.12 and Figure 4.13 show the results obtained in each case for the best result out of the three runs performed.

To draw a conclusion about the performance of the various topologies and the

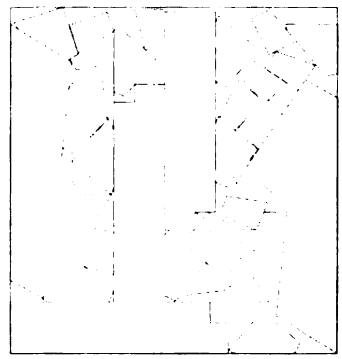

Figure 4.13: Solution to Problem 4 obtained using the PMX operator (Percentage Utilization = 83.54)

crossover operators this problem was run repetitively for three to four times for each experiment. The results shown here are the best obtained from each of the test cases. The multiple runs performed gave fairly good indications about the capabilities of the the various topologies, and helped in ultimately recommending the fully connected grid topology and the UOBX crossover operator. The sizeable amount of computer resources required for each run (generally at least overnight on a cluster of 20 processors) precluded making enough runs to draw a firm, statistially significant conclusion about each comparision. But there was sufficient consistency to strongly support the recommended choices given here.

4.5 Problem 5

Each of the parts of problem 4 was duplicated to yield a more challenging problem. Figure 4.14 shows the result obtained from best of the two runs. It can be seen that the algorithm is fairly scalable and produces sheet utilization comparable

Figure 4.14: Solution for Problem 5 (Percentage Utilization = 83.65)

to problem 4.

Number of Parts	56
Total Area of Parts	7497.50 sq. units
Total Rectangular Area Required	8962.94 sq. units
Width of Stock Sheet	65 units
Percentage Utilization	83.65
Generations Required	182
Topology Used	Fully Connected
Crossover Operator	UOBX
Mutation Operator	Swap
Selection	tournament selection
Stopping Criterion	200 generations

4.6 Conclusion

The above set of experiments shows the ability of the algorithm to solve both simple and complex nesting problems with good consistency. The time involved in

finding the final solution is dependent on the computational time needed for one evaluation of the objective function, which in turn is dependent on the number and complexity of parts constituting the problem. For example, the time required to calculate the objective function for problem 4 was 17 seconds, on a one processor of a dual processor machine in which each processor has a speed of 850 MHz, with 256 Mbytes of shared RAM. Thus the approximate run time required for this problem was 65 hours. However, for problem 1, the time required for the evaluation of one objective function was 1 second and thus the approximate run time for this problem was 4 hours. All the experiments were performed on a cluster of ten computers each running one or two subpopulations at a time (maximum of one subpopulation per processor).

Chapter 5: Discussion and Recommendations

This research work investigated the application of a shape-feature-matching heuristic and a coarse-grain PGA, to the irregular shape nesting problem.

The use of shape information and feature matching helped in finding feasible solutions very effectively. It made the search more efficient by doing local search within each evaluation of the GA.

An unusual grid topology, and migration scheme was designed and tested. The results suggest that, this led to the improvement in performance of the PGAs.

Nesting is essentially finding the right permutation of sequence of the part placement over the stock sheet. Even if we enumerate all the possible permutations for the case where we have just 10 parts, then it is equal to !10, which in turn is a large number, 3628800. Assuming one evaluation of the enumerated solution takes 1 seconds and we want to try all the possible combinations, it would require 42 days to solve the problem involving 10 parts. The approach presented here, helps in reducing the time involved to get a good solution considerably. But, it would still be inappropriate to use this approach for solving a problem, which requires real-time decision making, or which does not follows any particular template. For example, in the leather industry, the shape of the stock keeps changing every time, because it comes from animal skin. And requires generation of a fresh nesting pattern for each stock sheet.

However, in industries such as shipbuilding, where the material is quite costly and we need to cut same shapes repeatedly; even a half-percent improvement in packing density is sufficient to justify a fairly intensive search process, such as represented by the method described here.

5.1 Recommendations

- Once the layout has been generated, the next step is to generate the optimal tool path. This research does not address this problem, but it can be extended to provide the functionality of generating optimal tool path.

- In this case, the objective function is inversely proportional to the length of sheet used. But, this design would not work in the case where the stocks have irregular shape. So, there is a need to design the objective function that can represent the sheet utilization for the irregular stock sheets.

- The algorithm should be extended to solve problems involving multiple stock sheets.

- The feature matching heuristic should be extended to also handle curve-linear features.

- It is necessary to improve the current geometric algorithm in terms of computational speed. An intelligent method should be developed to distinguish between the features which are important and which are not, and while calculating the score function the unimportant features should not be used. This would save some computational time.

- The packing problem also finds application in the cargo industry, where its needed place the 3-dimensional shapes in optimal space. This algorithm can be extended to handle this kind of packing problems by adding routines capable of classifying and matching 3-dimensional features.

BIBLIOGRAPHY

Bibliography

[1] M. Adamovicz, and A. Albano (1976). "Nesting Two-Dimensional Shapes in Rectangular Modules," *Computer-Aided Design*, bf 8(1): 27-33.

[2] A. Albano, and G. Sapuppo (1980). "Optimal Location of Two-Dimensional Irregular Shapes Using Heuristic Search Methods," *IEEE Transactions on Systems, Man, and Cybernetics*, **10**(5): 242-248.

[3] A. R. Babu and N. R. Babu (1999). "Effective Nesting of Rectangular Parts in Multiple Rectangular Sheets Using Genetic and Heuristic Algorithms," *International Journal of Production Research*, **37**(7): 1625-1643.

[4] E. Burke and G. Kendall (1999). "Applying Simulated Annealing and No Fit Polygon to the Nesting Problem," *Proceedings of World Manufacturing Congress*, Durham UK 27-30.

[5] Cantu-Paz, E. (2000). *Efficient and Accurate Parallel Genetic Algorithms.* Kluwer Academic Publisher, Boston, MA.

[6] L. Davis (1991). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.

[7] Ananda A. Debnath (1997). *A New Approach to Solving 2-Dimensional Part Nesting and Layout Problems*,"M.S. Thesis, Dept. of Mechanical Engineering, Michigan State University.

[8] R. Dighe and M. J. Jakiela (1996). "Solving Pattern Nesting Problems with Genetic Algorithms Employing Task Decomposition and Contact Detection" *Evolutionary Computation* **3**. 239-266

[9] K. A. Dowsland and W. B. Dowsland (1995), "Solution Approaches to Irregular Nesting Problems", *European Journal of Operation Research* **84**: 506-521.

[10] H. Freeman and R. Shapira (1975). "Determining the Minimum Area Encasing Rectangle for an Arbitrary Closed Curve," *Communications of the ACM* bf 81(7): 409-413.

[11] D. E. Goldberg (1988).*Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

[12] Erik D. Goodman (1994). *An Introduction to GALOPPS*, Release 2.35, Technical Report GARAGe94-5, Genetic Algorithms Research and Applications Group (GARAGe), Michigan State University.

[13] Erik D. Goodman (1996) *An Introduction to GALOPPS*, Release 3.2, Technical Report GARAGe96-07-01, Genetic Algorithms Research and Applications Group (GARAGe), Michigan State University.

[14] G. C. Han and S. J. Na (1996). "Two-Stage Approach for Nesting in Two-Dimensional Cutting Problems Using Neural Network and Simulated Annealing," *Proceedings of Institution of Mechanical Engineers, Part:B Journal of Engineering Manufacture*, **210**: 509-519.

[15] J. H. Holland (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press.

[16] H. S. Ismail and K. K. B. Hon (1995). "The Nesting of Two-Dimensional Shapes Using Genetic Algorithms," *Proceedings of Institution of Mechanical Engineers, Part:B Journal of Engineering Manufacture*, **209**: 115-124.

[17] S. Jain and C. H. Gea (1998). "Two-Dimensional Packing Problem Using Genetic Algorithm," *Engineering with Computers,* **14**: 177-190.

[18] S. Jakobs (1996) "On genetic algorithms for packing of polygons", *European Journal of Operation Research*, **88**: 165-181.

[19] H. J. Lamousin, W. N. Waggenspack, and G. T. Dobson (1996). "Nesting of Complex 2-D Parts within Irregular Boundaries," *Transactions of ASME: Journal of Manufacturing Science and Engineering*, **118**: 615-622.

[20] H. J. Lamousin and W. N. Waggenspack (1996). "Nesting of Two-Dimensional Irregular Parts Using a Shape Reasoning Heuristic," *Computer-Aided Design*, **25**: 221-237.

[21] A. Y. C. Nee, K. W. Seow, and V. C. Long (1986). "Designing Algorithm for Nesting Irregular Shapes With and Without Boundary Constraints," *Annals of CIRP*, **35** (1): 107-110.

[22] O. P. Sha and R. Kumar (2000). "Nesting of Two Dimensional Irregular Parts Within an Irregular Boundary Using Genetic Algorithms," *Journal of Ship Production*, **1680**: 232-242.

APPENDICES

Appendix A: Crossover Operators

A.1 Partial Matched Crossover Operator (PMX)

The PMX operator (Davis, 1991; Goldberg, 1988) is carried out in following steps:

1. The two candidate strings are first aligned and two crossing sites are picked uniformly at random along the strings.

   A = 9 8 4 — 3 2 5 — 10 7 6 1

   B = 1 2 5 — 7 8 9 — 10 6 4 3

2. PMX proceeds by position wise exchanges. First, mapping string B to string A, the 7 and the 3, the 8 and 2, and the 9 and 5. Similarly also mapping string A to string B, the 3 and the 7, the 2 and the 8 and the 5 and the 9.

3. Under PMX we obtained following two offspring, containing ordering information partially determined by each parent.

   Child A = 5 2 4 — 7 8 9 — 10 3 6 1

   Child B = 1 8 9 — 3 2 5 — 10 6 4 7

A.2 Order Based Crossover

The order based crossover operator (Davis, 1991; Goldberg 1988) is carried out in following steps:

1. The order based crossover operators also starts off by aligning the two candidate strings and picking two crossing sites uniformly at random along the strings.

A: 9 2 3 — 5 6 8 — 1 4 8 10

B: 8 5 7 — 2 3 9 — 1 6 10 4

2. Like, PMX each strings maps to constituents of matching section of other parent. But instead of using point-by-point exchanges as in PMX, order based crossover uses a sliding motion to fill holes left by transferring the mapped positions. In this case when string B maps to string A, the positions of 5, 6 and 8 will leave holes (marked by H) in the string:

   A' = H H H— 5 6 8 — 1 4 8 10

   B' = H H 7 — 2 3 9 — 1 H 10 4

3. These holes are filled with a sliding motion that starts following the second crossover site:

   A' = 5 6 8 — H H H — 1 4 8 10

   B' = 2 3 9 — H H H — 1 10 4 7

4. These holes are then filled with the matching section from the other string. Performing this operation we obtain the two offspring's as follows:

   Child A: 5 6 8 — 2 3 9 — 1 4 8 10

   Child B: 2 3 9 — 5 6 8 — 1 10 4 7

A.3 Uniform Order Based Crossover Operator (UOBX)

The UOBX (Davis 1991) is carried out in following steps.

1. Selection of parents

   A: 1 2 3 4 5 6 7 8 9

   B: 2 4 5 7 9 6 3 1 8

2. In the second step a uniformly random binary template is generated.

   0 1 1 0 0 1 0 1 1

3. The 1's specify loci to be filled by the corresponding alleles of the first parent in the first child, and the 0's are filled on the second child with corresponding alleles of the second parent.

   Child A (partial): - 2 3 - - 6 - 8 9

   Child B (partial): 2 - - 7 9 - 3 - -

4. The void '-' space of the first child is filled by the genes of parent 2 in their order of appearance (without duplication, since the result must be a permutation) and the second child is handled correspondingly.

   Child A: 4 2 3 5 7 6 1 8 9

   Child B: 2 1 4 7 9 5 3 6 8

A.4 Cycle Crossover Operator (CX)

The cycle crossover (Davis, 1991; Goldberg, 1988) is carried out as follows:

1. The two candidates are randomly chosen.

   A = 9 8 2 1 7 4 5 10 6 3

   B = 1 2 3 4 5 6 7 8 9 10

2. In this case we start from the left by picking the first parent.

   A = 9 - - - - - - - - -

3. Since, we want each position to be taken from one of the parent, in this case the choice of 9 from string A means that we must get 1 from the string A because of the position of 1 in string B.

   A = 9 - - 1 - - - - - -

4. This selection in turn requires that we select position 4 from string C. This process continues until we are left with the following pattern:

$$A = 9 - - 1 - 4 - - 6 -$$

5. The selection of 6 means that we should now choose a 9 from string A: however this is not possible: a 9 having selected at the first position. So, we eventually return to the position of origin and this complete the cycle. Following the first cycle, the remaining positions are filled from the other string. Thus the final offspring's which are obtained are as follows:

Child A: 9 2 3 1 5 4 7 8 6 10

Child B: 1 8 2 4 7 6 5 10 9 3

Appendix B: Mutation Operator

B.1 Swap Mutation

In swap mutation (Davis, 1991; Goldberg, 1988) two positions are picked up and their alley's are exchanged. For example

2 4 5 7 8 0 1 3 6 9 2 1 5 7 8 0 4 3 6 9

B.2 Scramble Mutation

In scramble mutation (Davis, 1991; Goldberg, 1988) a subset of positions are picked at random and are reordered. For example

2 4 5 7 8 0 1 3 6 9 2 1 4 7 8 0 5 3 6 9

Appendix C: Selection Methods

C.1 Tournament Selection

In tournament selection a 'n' (where n ¿ 2 and less then population size) number of individuals are chosen at random from the population and the best individual among them is selected as parent. This implies that the bigger the 'n' more the selection pressure.

C.2 Stochastic Universal Sampling Method

Stochastic universal sampling method provides a bias free way of selecting individuals. In this methods the individual are mapped onto a roulette wheel with the sector size equals to their relative fitness. Next, N equally spaced pointers are drawn from the center, (where N equals to the number of individuals desired to be selected). The individuals which are pointed by the pointers are selected as parents. For example in the Figure C.1, there are 7 individual mapped onto the roulette wheel according to their relative fitness, and four individuals are desired to be selected. Therefore, four equally spaced pointers are drawn from the center. These pointers point to 6, 4, 1 and 7. Thus these individuals are selected as parents.

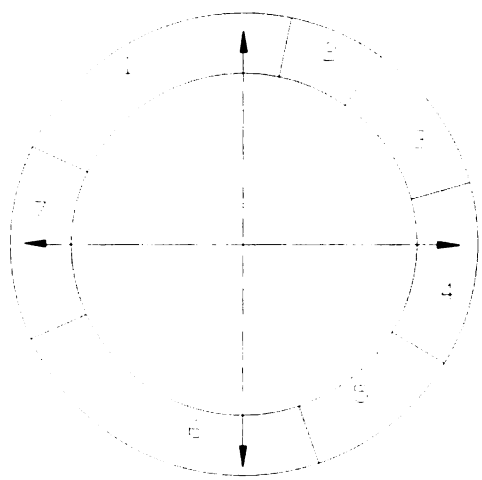

Figure C.1: Stochastic Universal Sampling Method

Appendix D: Geometry of the Example Problem

D.1 Data for Test Problem 1

```
NEWSTOCK
STOCKVERTEX 0 0
STOCKVERTEX 80 0
STOCKVERTEX 80 300
STOCKVERTEX 0 300
STOCKEND
PART
VERTEX 0 0
VERTEX 24 0
VERTEX 24 16
VERTEX 0 16
PARTEND
PART
VERTEX 0 0
VERTEX 28 0
VERTEX 28 16
VERTEX 0 16
PARTEND
PART
VERTEX 0 0
VERTEX 28 0
VERTEX 28 16
VERTEX 0 16
PARTEND
PART
VERTEX 0 0
VERTEX 60 0
VERTEX 60 14
VERTEX 0 14
PARTEND
PART
VERTEX 0 0
VERTEX 60 0
VERTEX 60 14
VERTEX 0 14
PARTEND
PART
VERTEX 0 0
VERTEX 20 0
VERTEX 20 28
VERTEX 0 28
PARTEND
PART
VERTEX 0 0
VERTEX 22 0
VERTEX 22 26
VERTEX 0 26
PARTEND
PART
VERTEX 0 0
VERTEX 22 0
VERTEX 22 26
VERTEX 0 26
PARTEND
```

```
PART
VERTEX 0 0
VERTEX 42 0
VERTEX 42 44
VERTEX 0 44
PARTEND
PART
VERTEX 0 0
VERTEX 18 0
VERTEX 18 70
VERTEX 0 70
PARTEND
PART
VERTEX 0 0
VERTEX 62 0
VERTEX 62 26
VERTEX 0 26
PARTEND
PART
VERTEX 0 0
VERTEX 18 0
VERTEX 18 48
VERTEX 0 48
PARTEND
PART
VERTEX 0 0
VERTEX 18 0
VERTEX 18 48
VERTEX 0 48
PARTEND
```

D.2 Data for Test Problem 2

```
NEWSTOCK
STOCKVERTEX 0 0
STOCKVERTEX 40 0
STOCKVERTEX 40 120
STOCKVERTEX 0 120
STOCKEND
PART
VERTEX 0 0
VERTEX 2 0
VERTEX 0 2
PARTEND
PART
VERTEX 0 0
VERTEX 3 0
VERTEX 0 3
PARTEND
PART
VERTEX 0 0
VERTEX 4 0
VERTEX 0 4
PARTEND
PART
VERTEX 0 0
VERTEX 5 0
VERTEX 0 5
PARTEND
PART
VERTEX 0 3
VERTEX 6 0
VERTEX 6 3
PARTEND
PART
VERTEX 0 4
```

```
VERTEX 7 0
VERTEX 7 4
PARTEND
PART
VERTEX 0 0
VERTEX 5 0
VERTEX 5 3
VERTEX 3 3
VERTEX 3 5
VERTEX 0 5
PARTEND
PART
VERTEX 0 0
VERTEX 4 0
VERTEX 4 1
VERTEX 2 1
VERTEX 2 4
VERTEX 0 4
PARTEND
PART
VERTEX 0 0
VERTEX 6 0
VERTEX 6 3
VERTEX 4 3
VERTEX 4 6
VERTEX 0 6
PARTEND
PART
VERTEX 0 0
VERTEX 5 0
VERTEX 5 2
VERTEX 3 2
VERTEX 3 1
VERTEX 0 1
PARTEND
PART
VERTEX 0 0
VERTEX 4 0
VERTEX 4 2
VERTEX 3 2
VERTEX 3 1
VERTEX 0 1
PARTEND
PART
VERTEX 0 0
VERTEX 6 0
VERTEX 6 3
VERTEX 4 3
VERTEX 4 6
VERTEX 0 6
PARTEND
PART
VERTEX 0 0
VERTEX 6 0
VERTEX 6 6
VERTEX 0 6
PARTEND
PART
VERTEX 0 0
VERTEX 5 0
VERTEX 5 5
VERTEX 0 5
PARTEND
PART
VERTEX 0 0
VERTEX 4 0
VERTEX 4 4
VERTEX 0 4
```

```
PARTEND
PART
VERTEX 2 0
VERTEX 4 0
VERTEX 4 2
VERTEX 6 2
VERTEX 6 4
VERTEX 4 4
VERTEX 4 6
VERTEX 2 6
VERTEX 2 4
VERTEX 0 4
VERTEX 0 2
VERTEX 2 2
PARTEND
PART
VERTEX 1 0
VERTEX 2 0
VERTEX 2 1
VERTEX 3 1
VERTEX 3 2
VERTEX 2 2
VERTEX 2 3
VERTEX 1 3
VERTEX 1 2
VERTEX 0 2
VERTEX 0 1
VERTEX 1 1
PARTEND
PART
VERTEX 2 0
VERTEX 4 0
VERTEX 4 2
VERTEX 6 2
VERTEX 6 4
VERTEX 4 4
VERTEX 4 6
VERTEX 2 6
VERTEX 2 4
VERTEX 0 4
VERTEX 0 2
VERTEX 2 2
PARTEND
PART
VERTEX 1 0
VERTEX 2 0
VERTEX 2 1
VERTEX 3 1
VERTEX 3 2
VERTEX 2 2
VERTEX 2 3
VERTEX 1 3
VERTEX 1 2
VERTEX 0 2
VERTEX 0 1
VERTEX 1 1
PARTEND
PART
VERTEX 0 0
VERTEX 6 0
VERTEX 6 3
VERTEX 0 3
PARTEND
PART
VERTEX 0 0
VERTEX 1 0
VERTEX 1 4
VERTEX 0 4
```

```
PARTEND
PART
VERTEX 0 0
VERTEX 5 0
VERTEX 5 2
VERTEX 0 2
PARTEND
PART
VERTEX 2 0
VERTEX 4 0
VERTEX 6 2
VERTEX 6 4
VERTEX 4 6
VERTEX 2 6
VERTEX 0 4
VERTEX 0 2
PARTEND
PART
VERTEX 3 0
VERTEX 6 0
VERTEX 8 2
VERTEX 8 4
VERTEX 6 6
VERTEX 3 6
VERTEX 0 4
VERTEX 0 2
PARTEND
PART
VERTEX 0 1
VERTEX 2 0
VERTEX 4 0
VERTEX 6 1
VERTEX 6 2
VERTEX 4 3
VERTEX 2 3
VERTEX 0 2
PARTEND
```

D.3 Data for Test Problem 3

```
NEWSTOCK
STOCKVERTEX 0 0
STOCKVERTEX 100 0
STOCKVERTEX 100 200
STOCKVERTEX 0 200
STOCKEND
PART
VERTEX 0 0
VERTEX 33 0
VERTEX 33 19
VERTEX 3 11
PARTEND
PART
VERTEX 0 0
VERTEX 42 0
VERTEX 37 30
VERTEX 0 19
PARTEND
PART
VERTEX 5 0
VERTEX 30 0
VERTEX 30 51
VERTEX 0 30
PARTEND
PART
VERTEX 0 0
```

```
VERTEX 3 11
VERTEX 7 33
VERTEX 8 38
VERTEX 0 36
PARTEND
PART
VERTEX 0 0
VERTEX 30 8
VERTEX 67 19
VERTEX 56 29
VERTEX 4 22
PARTEND
PART
VERTEX 23 0
VERTEX 53 21
VERTEX 53 70
VERTEX 19 70
VERTEX 7 42
VERTEX 0 23
VERTEX 12 10
PARTEND
PART
VERTEX 0 0
VERTEX 52 7
VERTEX 40 20
VERTEX 47 39
VERTEX 3 30
VERTEX 1 5
PARTEND
PART
VERTEX 0 0
VERTEX 8 2
VERTEX 10 27
VERTEX 12 64
VERTEX 0 64
PARTEND
PART
VERTEX 0 0
VERTEX 44 9
VERTEX 16 37
VERTEX 2 37
PARTEND
PART
VERTEX 0 28
VERTEX 28 0
VERTEX 40 28
PARTEND
```

D.4 Data for Test Problem 4

```
NEWSTOCK
STOCKVERTEX 0 0
STOCKVERTEX 65 0
STOCKVERTEX 65 120
STOCKVERTEX 0 120
STOCKEND
PART
VERTEX 2.1 2.1
VERTEX 8.4 9.6
VERTEX 0.6 16.2
VERTEX 0.0 15.0
VERTEX 0.0 3.1
PARTEND
PART
VERTEX 2.1 2.1
VERTEX 8.4 9.6
```

```
VERTEX 0.6 16.2
VERTEX 0.0 15.0
VERTEX 0.0 3.1
PARTEND
PART
VERTEX 2.1 2.1
VERTEX 8.4 9.6
VERTEX 0.6 16.2
VERTEX 0.0 15.0
VERTEX 0.0 3.1
PARTEND
PART
VERTEX 2.1 2.1
VERTEX 8.4 9.6
VERTEX 0.6 16.2
VERTEX 0.0 15.0
VERTEX 0.0 3.1
PARTEND
PART
VERTEX 2.1 2.1
VERTEX 8.4 9.6
VERTEX 0.6 16.2
VERTEX 0.0 15.0
VERTEX 0.0 3.1
PARTEND
PART
VERTEX 2.1 2.1
VERTEX 8.4 9.6
VERTEX 0.6 16.2
VERTEX 0.0 15.0
VERTEX 0.0 3.1
PARTEND
PART
VERTEX 2.1 2.1
VERTEX 8.4 9.6
VERTEX 0.6 16.2
VERTEX 0.0 15.0
VERTEX 0.0 3.1
PARTEND
PART
VERTEX 2.1 2.1
VERTEX 8.4 9.6
VERTEX 0.6 16.2
VERTEX 0.0 15.0
VERTEX 0.0 3.1
PARTEND
PART
VERTEX 2.1 2.1
VERTEX 8.4 9.6
VERTEX 0.6 16.2
VERTEX 0.0 15.0
VERTEX 0.0 3.1
PARTEND
PART
VERTEX 2.1 2.1
VERTEX 8.4 9.6
VERTEX 0.6 16.2
VERTEX 0.0 15.0
VERTEX 0.0 3.1
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 8.0 2.0
VERTEX 8.0 6.0
VERTEX 0.0 6.0
PARTEND
PART
VERTEX 0.0 0.0
```

```
VERTEX 8.0 2.0
VERTEX 8.0 6.0
VERTEX 0.0 6.0
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 8.0 2.0
VERTEX 8.0 6.0
VERTEX 0.0 6.0
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 8.0 2.0
VERTEX 8.0 6.0
VERTEX 0.0 6.0
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 8.0 2.0
VERTEX 8.0 6.0
VERTEX 0.0 6.0
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 8.0 2.0
VERTEX 8.0 6.0
VERTEX 0.0 6.0
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 8.0 2.0
VERTEX 8.0 6.0
VERTEX 0.0 6.0
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 8.0 2.0
VERTEX 8.0 6.0
VERTEX 0.0 6.0
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 8.0 2.0
VERTEX 8.0 6.0
VERTEX 0.0 6.0
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 8.0 2.0
VERTEX 8.0 6.0
VERTEX 0.0 6.0
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 10.5 0.0
VERTEX 10.5 2.4
VERTEX 6.9 3.9
VERTEX 6.9 6.3
VERTEX 10.5 13.8
VERTEX 10.5 16.2
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 10.5 0.0
VERTEX 10.5 2.4
VERTEX 6.9 3.9
VERTEX 6.9 6.3
VERTEX 10.5 13.8
```

```
VERTEX 10.5 16.2
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 10.5 0.0
VERTEX 10.5 2.4
VERTEX 6.9 3.9
VERTEX 6.9 6.3
VERTEX 10.5 13.8
VERTEX 10.5 16.2
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 43.5 0.0
VERTEX 43.5 6.0
VERTEX 39.9 6.0
VERTEX 39.9 10.2
VERTEX 0.0 10.2
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 43.5 0.0
VERTEX 43.5 6.0
VERTEX 39.9 6.0
VERTEX 39.9 10.2
VERTEX 0.0 10.2
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 43.5 0.0
VERTEX 43.5 6.0
VERTEX 39.9 6.0
VERTEX 39.9 10.2
VERTEX 0.0 10.2
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 51.0 0.0
VERTEX 54.6 6.3
VERTEX 33.9 20.1
VERTEX 33.9 14.7
VERTEX 28.8 9.9
VERTEX 7.5 12.0
PARTEND
PART
VERTEX 0.0 0.0
VERTEX 51.0 0.0
VERTEX 54.6 6.3
VERTEX 33.9 20.1
VERTEX 33.9 14.7
VERTEX 28.8 9.9
VERTEX 7.5 12.0
PARTEND
```

MICHIGAN STATE LIBRARIES
3 1293 02316 3342