Application of Genetic Algorithms to Group Technology

Lee Wai Hung

submitted to the Department of Systems Engineering and Engineering Management

in partial fulfillment of the requirements for the degree of

Master of Philosophy

at

The Chinese University of Hong Kong

June 1996

香港中文大學
大學
圖書館系統
12 MAY 1997
UNIVERSITY
LIBRARY SYSTEM
THE CHINESE UNIVERSITY OF HONG KONG

Abstract

Group technology (GT) is a management philosophy which capitalizes on similarity in manufacturing design and processing. One specific application of GT is cellular manufacturing which involves processing similar parts on a dedicated cluster of machines. Cellular manufacturing has been proposed to improve manufacturing efficiency and productivity. To implement cellular manufacturing design, parts must be grouped into part families and machines are grouped into machine cells.

GT problem requires heuristic method and there are lots of algorithms proposed in literature. However, most methods can not be applied to real size problems and only address one or a few aspects of the problems. In this research, we develop heuristic approaches for solving GT problems. Our consideration is comprehensive, we have investigated different GT problems, from simple models to complicated ones. Further, our approaches are applicable to solve large scale problem.

Three models of the problem that operate under different environments are the basis of the research and three genetic algorithm (GA) approaches are proposed to solve the problem. For standard model, a GA designed for traveling salesman problem is used. For generalized and integrated model, mutlichromosome GAs are utilized. Experimental results indicated that our approaches outperform the techniques suggested in literature. Especially, significant improvements can be identified for large size problems.

Acknowledgments

I would like to thank all the people who have helped and guided me through my graduate work.

In particular, I would like to express my deepest gratitude to my supervisor, Dr. C. H. Cheng, for his support, his helpful suggestions and advise. In addition, I would like to thank Professor Yash Gupta who gave helpful suggestions to this research and Dr. K. F. Wong who provided valuable comments to this thesis.

I would also like to thank all my colleagues who gave me an enjoyable and a memorable graduate life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Group Technology

Group technology (GT) is a management philosophy based on the idea of similarity. This approach was originally introduced by Mitrofanov in 1966. Later, Burbidge [8] developed a manual procedure — production flow analysis (PFA) — which uses part routing information to form machine groups. In PFA, similar parts are grouped and processed together. Actually, the idea of GT is not only applicable to process, design, production control, and part assembly, it can also be applied to other activities including administrative functions [2]. In this research, however, we focus the application of GT to manufacturing systems.

Traditionally, machine layout in a factory is process-oriented. Each department or section of a factory is composed of machines possessing similar capabilities and performing similar functions. This layout also referred to as

functional layout. If a part requires more than one process, it will travel from one department to other to have its processing requirements completed. The primary disadvantage of this layout is long and uncertain throughput time which leads to high work-in-process inventory, untimely product delivery, and increasing loss of sales [18].

One application of GT in the manufacturing environment is cellular manufacturing. Cellular manufacturing systems offer numerous benefits over functional layouts [2]. The main benefits include reduced lead time, reduced material handling costs, decreased work-in-process, reduced finished good inventories, and reduced setup time. Other benefits also include better production planning and control, improved job satisfaction, morale and communication. These features are essential for a firm to remain competitive in the current manufacturing environment. In Hyer and Wemmerlov's survey on the use of GT in the US manufacturing companies [34], respondents confirmed GT's usefulness and the opportunities to improve manufacturing productivity.

1.2 Cell design

Cell formation involves grouping functionally dissimilar machines together to process a group of parts. A cluster of machines is referred to as a machine cell and a group of parts is referred to as a part family (Figure 1.1). In an ideal situation, a part family can be completely processed within a machine cell. This cell formation procedure is a major step in designing a cellular manufacturing

system.

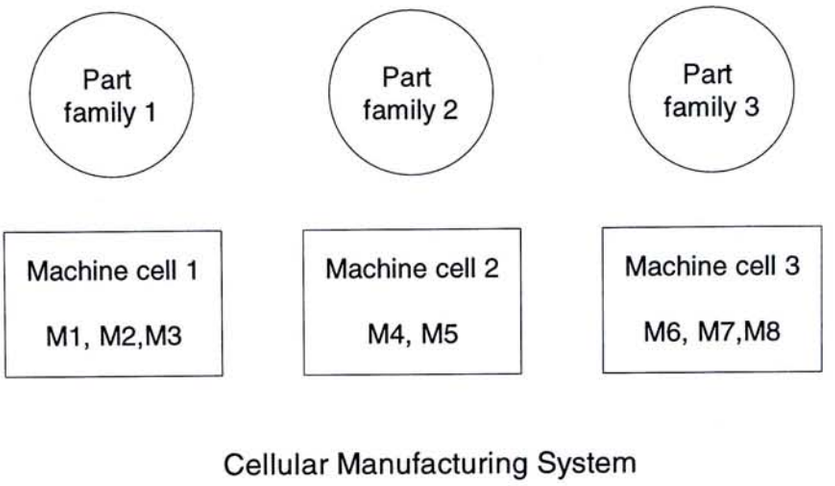

Figure 1.1: A manufacturing system with three machine cells and three part families

The cell formation problem is very difficult to solve [38, 51]. A large number of heuristic algorithms have been proposed in the literature. All these algorithms use some models. Broadly speaking, these models can be classified into two groups: standard models and generalized models. Standard models deal with a machine-part incidence matrix. Generalized models incorporate different design objectives and constraints to give a more realistic representation of manufacturing systems.

1.3 Objectives of the research

This problem can be solved using heuristic methods. There are extensive algorithms of such kind available in literature. However, they are limited to tackling small scale applications. In addition, most methods only consider one or a few aspects of the problem. For example, the rank order clustering proposed by King [37] ignored most manufacturing factors and considered only the operational requirements. This technique could not be applied to complicated models.

The objectives of this research are to develop heuristic approaches to the cell formation problem and to tackle the problem comprehensively, from simple models to complicated models. In particular, the approaches developed can handle large size problems.

1.4 Organization of thesis

In this thesis, we propose the application of genetic algorithms to solve the cell formation problem based on both standard models and generalized models. A literature review on designing cellular manufacturing systems is presented in Chapter 2. Chapter 3 proposes a genetic algorithm-based heuristic to deal with a standard model. We apply a genetic algorithm originally designed for the traveling salesman problem to group machines into cells and parts into families. The results are compared very favorably to a well-known algorithm available in the literature.

In solving generalized models, a different representation is required. A multi-chromosome representation for this problem is suggested in Chapter 4. Our representation generates better results when compared to an existing representation for the same workload model. Chapter 5 considers a more practical aspects of the clustering problem and demonstrates the extensibility of our GA in solving the problem. The algorithm takes into account the presence of alternative process plans. Plan selection and cell formation are solved simultaneously through an integrated model. Our conclusion will be presented in the final chapter.

Chapter 2

Literature review

2.1 Introduction

Methodologies for cell design use two types of models: standard models and generalized models [18]. A standard model ignores many manufacturing factors and only considers machinery operations of parts. In a standard model, a binary machine-part incidence matrix $[a_{ij}]$ is used to represent a manufacturing system. A matrix entry "1" ("0") indicates that machine i is used (not used) to process part j. Techniques dealing with this matrix formulation can be further categorized according to the type of algorithms employed to cluster the data, e.g., array-based clustering, cluster identification, graph-based, integer programming, seed-based, similarity coefficient, and artificial intelligence techniques.

Generalized models deal with the cellular manufacturing problem more comprehensively and incorporate different design objectives and constraints.

There are three types of models: machine assignment models, part family models, and cell formation models. A machine assignment model assigns machines to machine cells to process part families. A part family model groups n parts into p families based on similarity of part design and (or) manufacture. A cell formation model generates the grouping of parts and clustering of machines simultaneously.

2.2 Standard models

When a machine-part incidence matrix $[a_{ij}]$ is constructed, it does not display clusters of machines and parts. For example, matrix 1 (see Figure 2.1) does not show any identifiable clusters.

		Parts				
		P1	P2	P3	P4	P5
	M1		1		1	1
Machines	M2	1		1		
	M3		1			1
	M4	1		1		

Figure 2.1: Matrix 1

A clustering algorithm transforms the initial incidence matrix to one with a diagonal structure by rearranging rows and columns. A diagonal block structure is desirable because the number of clusters and the components of clusters are easily identified through visual analysis. Matrix 2 (see Figure 2.2) shows two diagonal clusters.

In this example, machines M2 and M4 form a cell that processes parts P1

		Parts				
		P1	P3	P2	P4	P5
	M2	1	1			
Machines	M4	1	1			
	M1			1	1	1
	M3			1		1

Figure 2.2: Matrix 2

and P3. Machines M1 and M3, on the other hand, process the part family that consists of parts P2, P4, and P5. This is an ideal example because mutually separable clusters can be formed.

		Parts				
		P1	P2	P3	P4	P5
	M1	1	1			1
Machines	M2	1	1			
	M3			1	1	1
	M4			1	1	

Figure 2.3: Matrix 3

In real cases, mutually separable clusters rarely occur. For example, matrix 3 (see Figure 2.3) cannot decompose into mutually separable clusters. Part 5 requires an operation in the other machine cell. This intercell move increases material handling cost. Clustering algorithms are needed to produce machine cells and part families with minimum number of intercellular moves. Some clustering algorithms are briefly reviewed in the following sections according to the categories mentioned in Section 2.1.

2.2.1 Array-based methods

Array-based methods involve manipulation of rows and columns to produce machine cells and part families. Algorithms include the bond energy algorithm developed by McCormick et al. [60], the rank order clustering algorithm by King [37], the direct clustering algorithm described by Chan and Milner [10]. The bond energy algorithm assumes that a bond exists between machines and parts. The bond energy is the strength of the bond. The optimal solution under this algorithm is a matrix that maximizes the bond energy. The rank order clustering operates by treating the 0,1 of the incidence matrix as binary number and assigning a value to each row and column according to the position of '1'. After assigning the values, the rows are arranged in decreasing order from top to bottom and the columns are arranged in similar manner from left to right. The direct clustering algorithm is similar to the rank order clustering method but it is not sensitive to the initial configuration of a machine-part incidence matrix. Chu and Tsai [20] compared the three methods and showed that the bond energy algorithm outperformed the other two array-based methods.

2.2.2 Cluster identification

The cluster identification (CI) algorithm first draws a horizontal line through any row of an incidence matrix. For each single-crossed entry of "1", vertical lines are drawn through the corresponding columns. The drawing of horizontal and vertical lines continues until there are no more single-crossed entries of "1"

in the matrix. The double-crossed entries of "1" in the matrix correspond to a machine cell and part family. This class of algorithms include the cluster identification algorithm by Kusiak and the Chow [50], the branch-and-bound algorithm by Kusiak and Cheng [48], and the branching algorithm by Kusiak [46].

2.2.3 Graph-based methods

A graph consists of a set of nodes or vertices and arcs or edges. Each arc connects two nodes. In applying the graph-based method to solve the cell formation problem, we treat machines as nodes and material flows as arcs. Cost c_{ij} is associated with the amount of material flows on arc $i\ j$. The cells can be formed if the graph is partitioned into subgraphs. Each subgraph is a cluster of machines. Askin and Chiu [1] suggested a graph partitioning procedure to deal with the cell formation problem. Their algorithm attempts to minimize the total costs associated with the arcs between subgraphs. Arcs between subgraphs represent intercell moves. Therefore, this approach aims at minimizing intercell material handling. Other graph approaches include the minimum spanning tree (MST) method by Srinivasan [75].

2.2.4 Integer programming

Clustering can be viewed as an optimization problem and therefore the problem can be modeled using an integer programming technique. These methods allow

the number of cells and the size of each cell be constrained. The close neighbor algorithm by Boe and Cheng [6], the A*-based algorithm proposed by Kusiak et al. [47], and the algorithm developed by Boctor [5] are some examples.

2.2.5 Seed-based

This type of clustering algorithms involves the generation of seed machines or parts. After the generation of seeds, other machines and parts are assigned to machine cells and part families based on some grouping measures. Examples of seed-based algorithms include the ideal seed algorithm [11], the zero-one data - ideal seed clustering (ZODIAC) by Chandrasekharan and Rajagopalan [13], and GRAFICS proposed by Srinivasan and Narendran [76].

2.2.6 Similarity coefficient

The algorithms of this category measure the similarity of a pair of machines and parts. A similarity matrix is often generated. The final output of the algorithms is a permutation of machines and parts with the maximum value of total similarity. These clustering techniques include single linkage clustering by McAuley [59], a method suggested by De Witte [22], similarity coefficient heuristic developed by Waghodekar and Sahu [84], and average linkage clustering by Seifoddini and Wolfe [71].

2.2.7 Artificial intelligence methods

Recently, several researchers employed artificial intelligence (AI) techniques in designing cellular manufacturing systems. Neural network approach is used by Karparthi and Suresk [36], and Chen and Cheng [16]. The Adaptive Resonance Theory (ART) neural network is used to form cells. One weakness of this approach is that the quality of a solution highly depends on the initial disposition of the incidence matrix. Chen et al. [15] proposed a simulated annealing solution to the cell formation problem. Venugopal and Narendran [83] proposed a genetic algorithm approach to the clustering problem. However, they employ a simple representation that may produce illegal solutions.

2.3 Generalized models

Standard models ignore many manufacturing factors such as part demand, the sequence of operations, machine utilizations. These models can only be used when a rough cut design is needed or when the detailed parameters are not available. On the other hand, generalized models consider more aspects of the cell formation problem. These models incorporate different design objectives, parameters, and constraints. A list of such objectives, parameters, and constraints can be long, for example, minimization of the cost of machines, set up cost, intercellular moves, material handling cost, work-in-process cost, intracell load imbalances, intercell load imbalances, maximization of the cell utilization, compatibility between machines and parts, and restriction of number

of cells. Generalized models can be further categorized into three classes, machine assignment models, part family models, and cell formation models.

2.3.1 Machine assignment models

Machine assignment models operate by forming machine cells based on some objective function measures. Parts are then allocated to appropriate machine cells according to the processing requirements or cell utilization rates. Gunasingh and Lashkari [28] suggested two 0-1 integer programming models to address the machine allocation problem. One model attempts to maximize the compatibility between machines and parts while the other one considers the trade-off between the cost of machine allocation and intercellular moves. These formulations restrict the size of a machine cell and the number of copies of machines. Harhalakis et al. [31] proposed a simulated annealing approach to allocate machines. The objective of their study is to minimize the intercellular moves. Cheng et al. [19] formulated machine allocation problem as a 0-1 integer programming model and used a branch-and-bound algorithm to find a solution.

2.3.2 Part family models

Part family models focus on formation of part families. The required machines are often duplicated in each machine cell. Kusiak [44] suggested a heuristic based on the part similarities to group parts into families. Suresh et al. [80] proposed a hierarchical methodology to solve a multi-objective model.

2.3.3 Cell formation models

In a cell formation model, the groups of machines and parts are determined simultaneously. Askin and Subramanian [3] proposed a cost-based heuristic for group technology configuration. The procedure considers costs of work-in-process inventory, material handling, and machine setups. A three stage procedure was used to solve the problem. Wei and Gaither [85] developed a cell formation model with four objective functions. The four objectives are minimization of bottleneck cost, maximization of the average cell utilization, minimization of intracell load imbalances, and minimization of intercell load imbalances. A linear integer programming enumeration scheme was used to solve the model. Sule [79] suggested to consider the cost and machine capacity in grouping. This approach was suggested to be more economical. Heragu and Gupta [32] presented a heuristic for machine cell-part family identification that addresses several design constraints; machine capacity, technological requirements, and number of cells. While generating a solution, the heuristic minimizes the intercellular moves of parts.

Chapter 3

Genetic cell formation algorithm

3.1 Introduction

In designing a cellular manufacturing system, parts requiring similar operations are grouped into a part family. Machines are identified to form a machine cell to process the part family. This design problem is made complicated by exceptional parts and/or exceptional machines [17]. An exceptional part is a part that requires processing in another machine cell. An exceptional machine is a machine that processes parts from a different part family. Both exceptional parts and exceptional machines cause intercellular movement of parts.

A 0-1 machine-part incidence matrix is used to model the manufacturing system. It is easy for practitioners to understand and can provide a rough cut design. A system designer may modify the rough cut design to derive a final cellular layout. When the final cellular layout is determined, the machine utilization cost, the utilization of machines, and the cost of intercellular moves

must be closely scrutinized.

In this chapter, the grouping problem is formulated as the traveling salesman problem [52, 73]. A grouping approach based on a genetic algorithm is proposed. The proposed algorithm is compared to a well-known algorithm using many test problems drawn from the literature. The comparative study shows that the proposed algorithm is very reliable and produces improved solutions.

3.2 TSP formulation for a permutation of machines

Given a machine-part incidence matrix $[a_{ij}]$, clustering involves rearrangement of rows and columns to create machine cells (i.e. blocks) that contain parts using similar machines and reduce intercellular moves among machine cells. In a solution matrix, a block diagonal form is often desirable because the blocks can be easily identified to facilitate subsequent cell design decisions. To illustrate the clustering concept, we consider an input matrix given in Figure 3.1.

		Parts				
		1	2	3	4	5
	1	0	1	0	1	1
Machines	2	1	0	1	0	1
	3	0	1	0	1	1
	4	1	0	1	0	0

Figure 3.1: An initial matrix

An initial matrix does not display any blocks (clusters). After rearrangement

of rows and columns, we obtain two blocks along the diagonal of a solution matrix (Figure 3.2).

		Parts				
		1	3	2	4	5
	2	1	1	0	0	1
Machines	4	1	1	0	0	0
	1	0	0	1	1	1
	3	0	0	1	1	1

Figure 3.2: A solution matrix

The problem of arranging rows and columns is similar to a permutation problem. In order to determine the desirable permutation for rows and columns in a solution matrix, we define a distance measure between a pair of rows (machines). Many such measures for cellular manufacturing were suggested in Shafer and Rogers [72]. In this thesis, we use the following distance measure for machines i and j:

$$d_{ij} = \sum_{k=1}^{n} |a_{ik} - a_{jk}| .$$

This measure belongs to a family of Minkowski metrics. The Minkowski metric for machines is given by $M_{ij}(p)$, the distance between machines i and j as a function of p, and is defined as:

$$M_{ij}(p) = (\sum_{k=1}^{n} |x_{ik} - x_{jk}|^p)^{\frac{1}{p}}$$

where $p > 0$ determines the particular metric used. For $p = 1$, the measure is known as rectilinear, city block, or absolute metric. When $p = 2$, it is

squared Euclidean metrics while chebychev (or infinity) metric is obtained when $p = \infty$. Note that for a 0-1 data, the rectilinear and the squared Euclidean metric produce the same result. After several experiments, this measure is proved to produce better results.

A small distance value between two machines implies that both machines process a number of common parts. Should two machines with a small distance value be placed in different machine cells, parts requiring the two machines must be transported between the cells, which results in increased material handling. Therefore, a cellular manufacturing clustering algorithm must place machines processing similar parts (and parts requiring similar machines) close to one another in a final permutation. This in turn attempts to minimize the total distance between pairs of machines.

The cellular manufacturing clustering problem can be formulated as a traveling salesman problem (TSP)[52, 73]. Lenstra and Rinnooy [52] showed that the clustering problem can be solved if we solve the associated TSP. Cities in a TSP correspond to machines.

Various approaches are proposed to solve TSPs, such as cutting planes [62], branch and bound [68], neural networks [74], 2-opt [54], simulated annealing [39], Markov chain [58], tabu search [25] and genetic algorithm [87].

Our motivation is to base on the TSP formulation for the cell formation problem and apply GA in solving this problem. To implement a genetic algorithm, several aspects are required to consider, for example initialization, chromosome representation, crossover operator, fitness function, replacement

strategy, and termination conditions.

3.3 Genetic algorithms

Genetic algorithms (GAs) were introduced by John Holland [33]. They are applied to a number of fields like mathematics, engineering, biology and social sciences [26]. A precise definition of genetic algorithms can be obtained from Goldberg [26]:

> Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search.

The concept of GAs is based on the evolution process that occurs in natural biology. An initial population of possible solutions (individuals) is generated. Some individuals are selected to be parents to produce offsprings via a crossover operator. All the individuals are then evaluated and selected based on the concept of survival of the fittest introduced by Darwin. The process of reproduction, evaluation and selection is repeated until a termination criterion is reached. Besides, a mutation operator with certain probability is applied to individuals to change their genetic makeup. The purpose of mutation are to increase the diversity of the population and to enable every point in the searching space be reachable.

The universe of all possible strings can be considered as an imaginary landscape; valleys mark the location of strings that encode poor solutions, and the landscape's highest point corresponds to the best possible string. The evolving population make genetic algorithms exploring such a landscape simultaneously. This characteristic is called implicit parallelism which enables GAs to be a powerful searching technique. A pseudo-code of GAs is as follows:

Algorithm 3.1 GA()

```
    ▷ A simple genetic algorithm
1   initialize (population)
2   while (the termination condition is not met) do
3       parent1 ← selection (population)
4       parent2 ← selection (population)
5       offspring ← crossover (parent1, parent2)
6       if mutate then
7           mutation (offspring)
8       evaluation (offspring)
9       population ← replacement (population, offspring)
10  end
```

3.3.1 Representation and basic crossover operators

Classically, an individual is represented by a binary string called chromosome, e.g. x_1=(1011001) and x_2=(0111011). Offsprings (another solutions) are generated by crossover. A crossover point will be selected randomly along the chromosome. The parent chromosomes will be split at that point and the segments of those chromosomes will be exchanged. For example, assume the parents are x_1 and x_2 as described above. If the crossover point is 3,

then the offsprings will be y_1=(1011011) and y_2=(0111001). This type of information exchange combines strings containing partial solutions. Two fit individuals (with higher fitness value) may combine their traits and make a superfit offspring.

For TSP, three vector representations were used [61]: *adjacency*, *ordinal*, and *path*. Each representation has its own genetic operators. Among the three representations, the path representation is perhaps the most natural one of a tour. For example, a tour

$$3-4-1-6-5-2-7$$

is represented simply as (3 4 1 6 5 2 7).

3.3.2 Fitness function

Fitness function is used to evaluate (see line 1, Algorithm 3.1) the value of the individuals within the population. According to the fitness value scored, the individual is selected as a parent to produce offsprings in the next generation or is selected to disappear in the next generation. The fitness function we used is the total distance for a TSP tour.

Let ρ be a permutation of machines and σ be a permutation of parts. For a permutation of machine:

$$3-1-2-8-7-4-6-9-5$$

$\rho(1)$ is 3, and $\rho(4)$ is 8. In the first phase, the proposed approach converts

the initial permutation of machines (specified by the initial matrix) to a new permutation that minimize the following fitness function:

$$Total\ distance = \sum_{i=1}^{m-1}\sum_{k=1}^{n}\left|a_{\rho(i)\sigma(k)} - a_{\rho(i+1)\sigma(k)}\right| + \sum_{k=1}^{n}\left|a_{\rho(m)\sigma(k)} - a_{\rho(1)\sigma(k)}\right| \quad (3.1)$$

where

m = number of machines

n = number of parts

$$a_{\rho(i)\sigma(k)} = \begin{cases} 1 \text{ if machine } \rho(i) \text{ processess part } \sigma(k) \\ 0 \text{ otherwise} \end{cases}$$

By minimizing the total distance, machines that process similar parts are grouped together. After rearranging machines in the initial matrix according to the permutation of machines obtained in the first phase, we obtain an intermediate matrix in which the positions of parts have not been changed.

3.3.3 Initialization

Initialization (see line 1, Algorithm 3.1) involves generating of possible solutions to the problem. It can be generated randomly or with some heuristic. Suitable heuristic can reduce the number of generation required in finding the solutions and let GAs start the search in a more favorable region of the search space. Certainly, this requires additional overhead. In our implementation, the initial population is generated randomly.

3.3.4 Parent selection strategies

Parent selection (see lines 3 & 4, Algorithm 3.1) is a process that allocates reproductive opportunities to individuals. In principle, individuals with higher fitness values are more likely to be selected into the mating pool and individuals with lower fitness values will receive lower or even no opportunity to act as parents. The probability of an individual being selected with P individuals in the population:

$$P_i = \frac{f(i)}{\sum_{j=1}^{P} f(j)}$$

This biased selection enables the convergence of population. There are several schemes for determining and assigning the selection probability, e.g. roulette wheel selection, scaling techniques, and ranking. In addition, non-probabilistic selection strategies may be used such as tournament selection, elitist models [26, 61], etc. As the process continues, the variation in fitness range will be reduced. But this often leads to the problem of premature convergence, a classical problem of GAs. This problem occurs because a few super-fit individuals receive high reproductive trials and rapidly dominate the population. If such individuals correspond to local optimum, GAs will be trapped like hill climbing.

In our implementation, fitness ranking [86] is employed to solve this problem. Individuals are sorted according to their fitness values, the number of reproductive trials are then allocated according to their rank. Several experiments have shown ranking to be superior to fitness scaling [86], in dealing

with premature convergence.

3.3.5 Crossover

Standard crossover operator cannot be used because illegal offspring may be generated. Illegal offspring is the one the contains either a machine occurs more than once or not all the machines appear in the tour. This violates the constraints of TSP. Several crossover operators (see line 5, Algorithm 3.1) are defined for this representation: partially-mapped (PMX) [27], order (OX) [21], cycle (CX) [67], and edge recombination (ER) [87] crossovers. ER was suggested to be most efficient for TSP [86]. In our clustering problem, we will use the path representation. Each gene in a chromosome corresponds to a machine (in the first phase).

Sequencing tasks involve permutation of objects of the problem domain. Applying genetic algorithms to sequencing problem requires specialized crossover operator, if path representation is used. The effectiveness of the operators can affect the performance of GAs and the quality of the solutions. Starkweather [78] conducted a study to compare six genetic sequencing operators for a 30 city "Blind" traveling salesman problem and a real world warehouse/shipping scheduling application. The results indicated that the effectiveness of different operators is dependent on the problem domain. Operators which work well in the traveling salesman problem may not be effective for other types of sequencing problems, while operators which perform

poorly on the blind traveling salesman problem work extremely well for the warehouse scheduling task. Syswedra [81] discussed the relative importance of position, order, and adjacency for different sequencing tasks. These studies contradict to the concepts assumed by some researchers that all sequencing tasks are similar and that one genetic operator suffices for all types of sequencing problems. Edge recombination operators perform well on the traveling salesman problem stressing adjacency but they perform poorly for sequencing tasks where relative order is critical. We compared six different operators for our clustering problem. The operators involved are enhanced edge recombination operator [78], order crossover #1 [21], order crossover #2 [81], partially mapped crossover [27], cycle crossover [67], and position based crossover [81].

Our experiment is conducted based on GENITOR [86]. The initialization, selection, replacement, and population size are kept constant. We intend to provide a common test-bed for the different sequencing operators. All the genetic algorithms are allowed to run for a predefined number of generations and the change of fitness values are recorded. Figures 3.3, 3.4, and 3.5 are the performance results.

Three problems adopted from the literature are used as test problems. Our objective is to minimize the total distance of a tour; therefore, lower fitness corresponds to a better solution. Clearly, the results reveal the enhanced edge recombination operator outperforms other sequencing operators in our clustering problem. This may be due to the most important information for this cell formation problem is which machines/parts should be placed in close

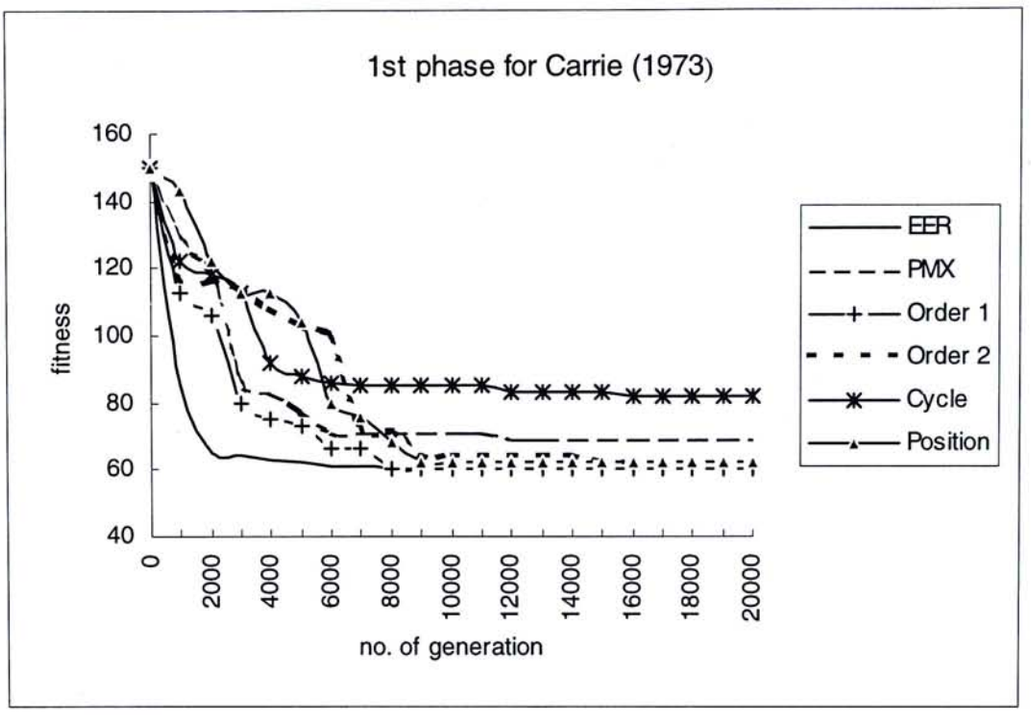

Figure 3.3: Performance analysis of different operators (I)

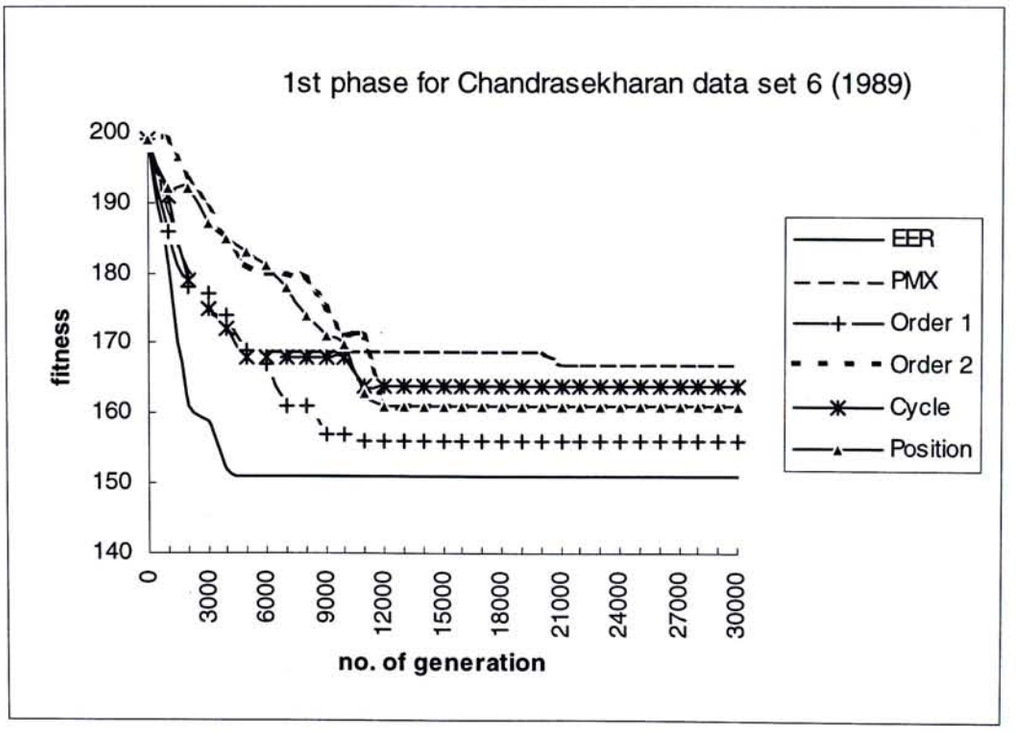

Figure 3.4: Performance analysis of different operators (II)

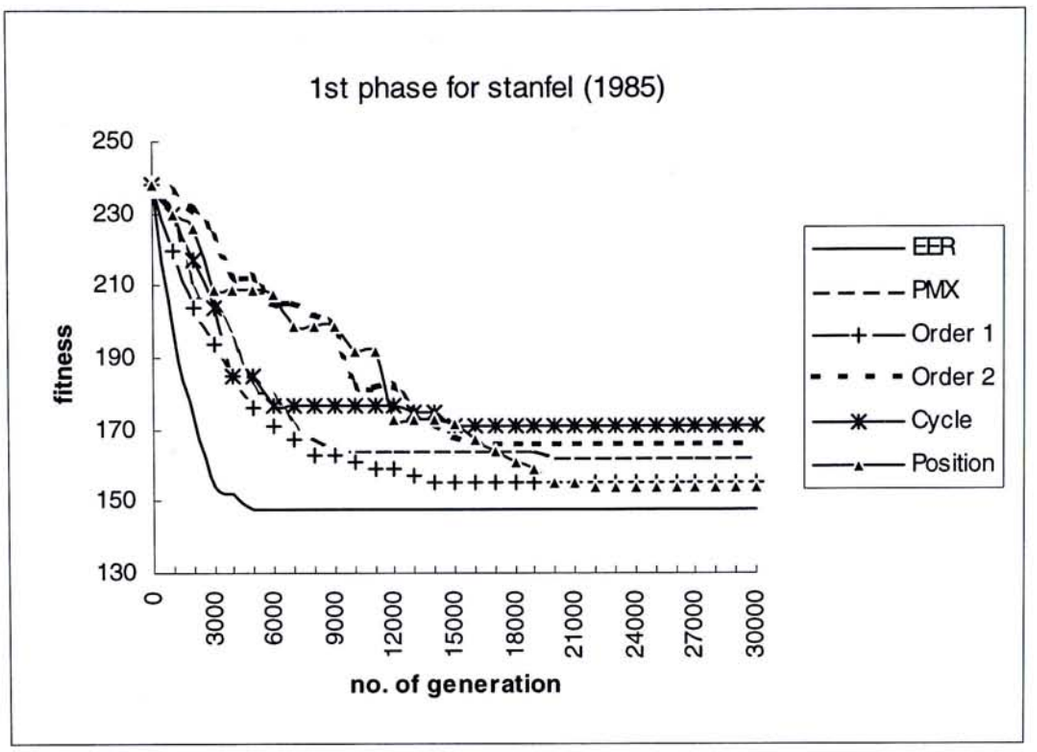

Figure 3.5: Performance analysis of different operators (III)

proximity. In other words, the adjacency is more important than the position. For example, if three machines should be close to each other, any sequence that they are arranged together will generate the same result no matter which position they are in. The enhanced edge recombination operator successfully captures this kind of information and therefore the GAs with this operator can generate better solution in less computational time.

As the enhanced edge recombination operator is demonstrated to be most efficient for the TSP-clustering problem. We will use this operator in our implementation. In the following section, we will discuss the operation of edge recombination operator. Consider the following tour:

$$3-1-2-8-7-4-6-9-5.$$

The edges are (3 1), (1 2), (2 8), (8 7), (7 4), (4 6), (6 9), (9 5), and (5 3). The first step is to build an edge table that stores the edges found in a tour. For each city c, all other cities connected to city c in at least one of the parents are listed. We can see that for each city c, there are at least two and at most four cities on the list. For example, given two parents:

$$parent\ 1 \ : \ 1-2-3-4-5-6-7-8-9$$

$$parent\ 2 \ : \ 4-1-2-8-7-6-9-3-5$$

the edge table can be constructed in Table 3.1.

City	edges connected to other cities
1	9,2,4
2	1,3,8
3	2,4,9,5
4	3,5,1
5	4,6,3
6	5,7,9
7	6,8
8	7,9,2
9	8,1,6,3

Table 3.1: An edge table

An offspring is constructed by selecting an initial city from one of the parents. Assume we have selected city 1, this city connected to three other cities: 9, 2, and 4. The next city selected depends on the number of edges in the edge table. The city with the smallest number of edges in the edge list is selected. Ties are broken arbitrarily. In our example, city 9 has four edges and cities 2 and 4 have

three. A random choice is made between cities 2 and 4 and we assume that city 4 is selected. Next, the cities available for selection are 3 and 5. Following the same principle city 5 is selected. Now, the first three genes of the offspring is constructed:

$$1 - 4 - 5 - x - x - x - x - x - x.$$

If we continue the procedure, an offspring with following sequence will be generated:

$$1 - 4 - 5 - 6 - 7 - 8 - 2 - 3 - 9.$$

With a random selection, there is a chance that a city may not have a continuing edge. This situation is called edge failure. When this occurs, a random city that presently not in the tour is selected as the next city.

An enhanced edge recombination operator is proposed to improve the performance of ER [78]. The modification is that if an edge appears in both parents, it will be first selected. The idea is to preserve the common subsequence in both parents. The element is still stored in an edge table but if an element is already present, a minus sign will be added to that city. This sign acts as a flag which indicates this edge appearing in both parents. The new edge table is shown in Table 3.2.

In selecting the next node, the city with the minus sign will be selected first. This enhanced operator is known to produce better solutions to TSP [78]. In our implementation, this enhanced ER is used as the crossover operator.

City	edges connected to other cities
1	9,-2,4
2	-1,3,8
3	2,4,9,5
4	3,-5,1
5	-4,6,3
6	5,-7,9
7	-6,-8
8	-7,9,2
9	8,1,6,3

Table 3.2: An edge table for enhanced ER

3.3.6 Mutation

Mutation (see line 7, Algorithm 3.1) is applied to each child individually after crossover according to the mutation rate. It provides a small amount of random search and helps ensure that no point in the search space has a zero probability of being examined. Several mutation operators are suggested for sequencing problem with path representation [61]:

- inversion — selects two points along the chromosome, the substring located in between these cut points is reversed. For example, in a chromosome:

1-2 | 3-4-5-6 | 7-8-9,

the two cut points are marked by "|". After inversion, the chromosome is changed to:

1-2 | 6-5-4-3 | 7-8-9;

- insertion — selects a gene and inserts it in a random place;

- displacement — selects a substring and inserts it in a random place;
- reciprocal exchange — swaps two genes in the chromosome.

In our implementation, no mutation operator is used. This is because the crossover operator incorporates random selection in completing a legal permutation and the effect is like a mutation.

3.3.7 Replacement

In most implementation [26], a whole population is replaced in each generation (see line 9, Algorithm 3.1). This is referred to as a generational approach. In GENITOR, however, a steady-state approach is adopted. In each generation only a few (typically two) individuals are replaced. In other words, parents and offsprings can co-exist in the population. The average fitness of the population will improve from generation to generation.

3.3.8 Termination

The processes of crossover, selection, and replacement are repeated until a termination criterion is met (see line 2, Algorithm 3.1). The simplest criterion is a pre-specified maximum number of generations. Other criterion involves calculating the variation of individuals, if the value below a certain threshold, the GA is terminated. In our case, maximum number of generations strategy is employed.

3.4 Formation of machine cells and part families

Once the machine sequence in the desirable permutation is generated by the genetic algorithm, the machine cells and part families can be determined based on the grouping measures. A heuristic that utilizes the distance information in the tour is used to partition machines into cells. The number of cells and families formed depends on the total number of machines (m). We assume that at least two cells will be formed. Therefore, the number of possible machine cells ranges from 2 to m. In each iteration, both machine and part assignments will be performed.

3.4.1 Objective functions

In order to compare the performance of different clustering techniques, we need some measures that can evaluate the quality of solutions generated by those algorithms. There are two measures frequently used in the literature. The first one is the grouping efficiency introduced by Chandrasekharan and Rajagopalan [11]. Grouping efficiency is a weighted average of two components, the mathematical formula is given as follows:

$$\eta = q\eta_1 + (1 - q)\,\eta_2$$

where

$$\eta_1 = \frac{e_d}{\sum_{r=1}^{k} M_r N_r}$$

$$\eta_2 = 1 - \left[\frac{e_o}{mn - \sum_{r=1}^{k} M_r N_r} \right]$$

m = number of machines

n = number of parts

M_r = number of machines in the rth cell

N_r = number of parts in the rth family

e_d = number of 1's within the machine/part groups

e_o = number of 1's outside the machine/part groups

k = number of cells

η = grouping efficiency

q = weighting factor $(0 \leq q \leq 1)$

Grouping efficiency ranges from 0 to 1. Higher grouping efficiency means that the more structure the solution is. In turn it means that solution contains fewer exceptional elements. The first element η_1 is the ratio of the number of '1' in the diagonal blocks of the rearranged matrix to the total number of possible '1' in all the diagonal blocks. This measure focuses on the within cell utilization or the within cell density. It is argued that the higher is this value, the greater is the similarity (in terms of processing requirements) between the components included in each cell and the greater is the utilization of the machines in this

cell. The second element η_2 is the ratio of the number of '1' in the off-diagonal blocks to the total number of possible '1' in the off-diagonal block. This measure focuses on the intercell material handling cost. Higher value of this measure means only a few operations are carried out in more than one cell. Therefore, maximizing this measure equals to minimizing the materials handling cost. If we try to maximize η_1, there will be more '1' in the off-diagonal block and η_2 will be reduced. Chandrasekharan and Rajagopalan [11] suggested the value of $q = 0.5$ and many researchers also use this value. We will follow this convention in calculating our results.

Although grouping efficiency can be used as a measure of the quality of solution, it has some limitations [40]. First, even a very bad solution with many exceptional elements showed efficiency figures around 75%. Second, the authors suggested $q = 0.5$ and intended to give equal weights to voids and exceptional elements. However, Kumar and Chandrasekharan [40] showed for large matrices, the denominator of the first term will be much or less of the same order. When the matrix size increases, the effect of exceptional elements becomes smaller, and in some cases, the effect of intercell moves is not reflected in the grouping efficiency. In the same paper, they proposed another measure called grouping efficacy (Γ). It can be expressed by the following formula:

$$\Gamma = \frac{1 - \Psi}{1 + \Phi}$$

where

$$\Psi = \frac{e_o}{e}$$

$$\Phi = \frac{e_v}{e}$$

e_o = number of 1's outside the machine/part groups

e_v = number of voids (zeros) within the machine/part groups

e = total number of operations (number of ones in the matrix)

Γ = grouping efficacy

Grouping efficacy also ranges from 0 to 1. When $\Gamma = 0$ implies that $\Psi = 1$ which means all the ones in the matrix are outside the machine/part group. When $\Gamma = 1$ that means $\Psi = \Phi = 0$ which corresponds to perfect grouping. As grouping efficiency is quite commonly used, we will report both grouping efficiency and grouping efficacy in our algorithm.

3.4.2 Machine assignment

If we examine a given machine sequence generated by the genetic algorithms, the machines that should be placed together will have a small distance measure. On the other hand, machines that should be allocated in different machine cells will have a large distance. The machine cells are formed by partitioning the sequence. If two cells are required, the machine sequence will be broken at the first two largest distance edges yielding two sub-sequences which correspond to the two machine cells. Ties are broken arbitrarily. Subsequently, additional machine cells can be formed by breaking the largest distance edges that have remained unbroken.

3.4.3 Part assignment

Both grouping measures emphasize on maximization of the number of '1' within the blocks. Therefore, parts are assigned to maximize operation within the machine cells. As the number of operations for a given part is fixed, attempts to maximize the operation in cells lead to reduced number of operation performed outside the cell (therefore reduced material handling cost).

Once machine cells and part families are generated, grouping efficiency or grouping efficacy can be calculated. The machine and part assignment iteration will be continued until all edges are broken. The best solution is the one that provide the best grouping measures.

3.5 Implementation

The TSP-clustering problem was solved based on a publicly available package called GENITOR [86]. The cell formation procedure was developed using C. All the program was run on a Sun SPARC 10 machine. In summary, our clustering approach contains two phases:

Algorithm 3.2 Generate-machine-sequence (in: incidence matrix; out: intermediate matrix)

▷ *Generate the machine sequence by genetic algorithms*

1 calculate the distance matrix ▷ *By equation 3.1*
2 generate an initial population of N random solutions
3 **for** i←1 to Generation ▷ *Set iteration counter*
4 select two parents P1 and P2 ▷ *By fitness ranking*
5 combine P1 and P2 to form a new offspring using enhanced edge recombination operator ▷ *Crossover*
6 replace the worst individual in the population with new offspring
7 **end**
8 output the intermediate matrix ▷ *Machine sequence reordered*

Algorithm 3.3 Cell-formation (in: intermediate matrix; out: final matrix

▷ *Machine cells and part families formation*

1 b ←2 ▷ *The best number of cells*
2 **for** c ← 2 to m ▷ *m is the number of machine*
▷ *The iteration starts from 2 to the total number of machine*
3 break first *c* longest edges of the machine sequence (a tour)
4 form the machine cells ▷ *Formation of machine cells*
5 assign each part to the cell that maximizes the operations within the machine cell ▷ *Formation of part families*
6 calculate the grouping efficiency or grouping efficacy
7 **if** the solution is better **then** ▷ *With higher grouping measures*
8 b ← c ▷ *The best known value*
9 **end**
10 output the results ▷ *Final matrix and the grouping measures*

3.6 An illustrative example

In order to provide a comprehensive understanding of our heuristic, an example adopted from Chandrasekharan and Rajagopalan [11] is used for illustration. Figure 3.6 is the initial machine-part matrix.

		Parts																			
											1	1	1	1	1	1	1	1	1	1	2
		1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
M	1	0	1	1	0	0	0	0	1	1	0	1	0	1	1	0	1	1	0	1	0
a	2	0	0	1	1	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	1
c	3	0	1	0	0	0	0	0	1	1	0	1	0	1	1	0	1	1	0	1	0
h	4	0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0	1	0	1
i	5	1	0	0	0	1	1	0	0	0	1	0	1	0	0	1	0	1	0	0	0
n	6	1	0	0	0	1	0	0	0	1	1	0	1	0	0	1	0	0	0	0	1
e	7	0	0	1	1	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	1
s	8	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1

Figure 3.6: Initial matrix

The first step is to calculate the distance matrix based on distance measure (the rectilinear metric). The distance matrix is in Figure 3.7.

		Machines							
		1	2	3	4	5	6	7	8
M	1	0	13	1	15	15	15	14	14
a	2	13	0	14	2	12	12	3	1
c	3	1	14	0	16	14	14	15	15
h	4	15	2	16	0	10	10	3	1
i	5	15	12	14	10	0	4	11	11
n	6	15	12	14	10	4	0	11	11
e	7	14	3	15	3	11	11	0	2
s	8	14	1	15	1	11	11	2	0

Figure 3.7: Distance matrix for machine grouping

After several experiments, the following parameters are shown to produce good results: the population size is 400 and the GA terminates after 500 generations.

```
  0 | Bst:  48.000000  Wst:  104.00000  Median:  82.000000  Avg:  79.610000
100 | Bst:  48.000000  Wst:  90.000000  Median:  76.000000  Avg:  73.835000
200 | Bst:  48.000000  Wst:  84.000000  Median:  68.000000  Avg:  68.945000
300 | Bst:  48.000000  Wst:  78.000000  Median:  66.000000  Avg:  63.995000
400 | Bst:  48.000000  Wst:  74.000000  Median:  60.000000  Avg:  59.330000
500 | Bst:  48.000000  Wst:  66.000000  Median:  56.000000  Avg:  55.955000

3 1 2 8 4 7 6 5   48.000000
```

Figure 3.8: Program output for phase 1

Figure 3.8 is the program output. In the figure 'Bst' stands for best value obtained so far, 'Wst' stands for worst individual encountered, 'Median' is the middle number of the population, and 'Avg' is the average fitness value of the whole population. The last sequence indicates the best permutation of machines found and the corresponding fitness value is 48.00. After phase 1, an intermediate matrix is generated (Figure 3.9). We can see that machines that process similar parts are already grouped together.

In phase 2, we try to find machine cells and part families based on the machine sequence formed in phase 1. Table 3.10 is the distance between two adjacent machine.

If two cells are formed, the first two largest distance edges will be removed yielding two sub-sequences: {3-1} and {2-8-4-7-6-5}. Parts are then assigned to the cells. The grouping efficiency is 80.04% and grouping efficacy is 62.92%. If we continue to three cells, three sub-sequences will be generated: {3-1}, {2-8-

		Parts																			
											1	1	1	1	1	1	1	1	1	1	2
		1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
M	3	0	1	0	0	0	0	0	1	1	0	1	0	1	1	0	1	1	0	1	0
a	1	0	1	1	0	0	0	0	1	1	0	1	0	1	1	0	1	1	0	1	0
c	2	0	0	1	1	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	1
h	8	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1
i	4	0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0	1	0	1
n	7	0	0	1	1	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	1
e	6	1	0	0	0	1	0	0	0	1	1	0	1	0	0	1	0	0	0	0	1
s	5	1	0	0	0	1	1	0	0	0	1	0	1	0	0	1	0	1	0	0	0

Figure 3.9: An intermediate matrix

Pair of machine	Distance
3-1	1
1-2	13
2-8	1
8-4	1
4-7	3
7-6	11
6-5	4
5-3	14

Figure 3.10: The distance between every pair of machines

4-7}, and {6-5}. The grouping measures are 95.83% and 85.25%, respectively. For four cells configuration, machine cells {3-1}, {2,-8-4-7}, {6}, and {5} are formed. Both grouping efficiency and grouping efficacy are decreased (93.81% and 77.05%). After testing the six possible configurations (from 2 cells to 7 cells), three cells configuration is shown to be the best solution in terms of the grouping measures.

After the two phases, a final matrix is generated. The solution has nine exceptional elements and, no voids within the blocks, and a grouping efficiency of 95.83%. This solution is also known as the optimal solution for this data set [11].

		Parts																			
					1	1	1	1	1	1					1	2			1	1	1
		2	8	9	1	3	4	6	7	9	3	4	6	7	8	0	1	5	0	2	5
M	3	1	1	1	1	1	1	1	1	1											
a	1	1	1	1	1	1	1	1	1	1	1										
c	2						1				1	1	1	1	1	1					
h	8										1	1	1	1	1	1					
i	4										1	1	1	1	1	1			1		
n	7				1						1	1	1	1	1	1				1	
e	6			1												1	1	1	1	1	1
s	5								1				1				1	1	1	1	1

Figure 3.11: The final matrix

The machine cells and part families formed are shown in Table 3.3.

Cell/family	Machines	Parts
1	1,3	2,8,9,11,13,14,16,17,19
2	2,4,7,8	3,4,6,7,18,20
3	5,6	1,5,10,12,15

Table 3.3: Machine cells and part families formed

3.7 Comparative Study

In order to test our heuristic, 25 data sets from the literature have been collected for evaluation. The size of matrices ranges from 5 x 7 to 40 x 100 and both well-structured and ill-structured forms are included. We will compare our results to ZODIAC [13]. Miltenburg and Zhang [63] compared nine clustering methods and showed ZODIAC outperformed array-based methods and similarity coefficient methods. It is a reliable clustering algorithm and commonly used in comparative studies.

The detailed results of the experiments are presented in Table 3.4. The table indicates that for small size problems, both ZODIAC and our algorithm produce similar results. This is because optimal solutions for small size problems are easier to obtain. But in some cases, improvement can still be found, the result of problem 1 reports higher values in both measures. For larger size problems (e.g. up to 20 x 20), significant improvement can be identified. Our algorithm performs better in both grouping efficiency and grouping efficacy. It is worthy to point out that some of the solutions generated are of better quality than any published results. For example, the solution of problem 9 has a grouping efficiency 88.83% and grouping efficacy 70.83%. Also, for the largest data set in our experiment (problem 25), most clustering algorithms [75, 76] generate result with grouping efficacy 83.92% and our heuristic successfully find out a solution with higher value. This value surpasses all the known results. It is clear that the GA-based approach has the ability to solve larger scale problems.

Only in problem 11, we find that ZODIAC performs better than our algorithm in both measures. The computational requirement is not high. For a 40 x 100 matrix, a solution can be obtained in less than 3 minutes.

Figures 3.12, 3.13, and 3.14 are some examples of varying densities and exceptional elements. Tables 3.5, 3.6, and 3.7 show the detailed assignment of machines and parts.

3.8 Conclusions

Machine-part clustering can be modeled by a 0-1 incidence matrix. A grouping algorithm involves rearrangement of rows and columns of such matrix. In this chapter, we showed that the clustering problem can be formulated as a traveling salesman problem. We proposed a genetic algorithm to solve the TSP grouping problem. After the generation of machine sequence, a heuristic is used to partition the machines into cells. The algorithm is compared to a well-known algorithm presented in the literature. The results showed our proposed heuristic can successfully yield a final matrix with better grouping measures.

			Grouping efficiency (η)		Grouping efficacy (Γ)	
No.	Source	Size	ZODIAC	GA	ZODIAC	GA
1	Waghodekar and Sahu [84]	5 x 7	72.20	77.10	56.52	68.00
2	Seifoddini [70]	5 x 18	86.76	89.14	77.36	79.59
3	Kusiak and Cho [49]	6 x 7	87.50	87.50	76.92	76.92
4	Kusiak and Chow [50]	7 x 11	65.01	81.40	39.13	58.92
5	Boctor [5]	7 x 11	86.08	86.08	70.37	70.37
6	Chandrasekharan and Rajagopalan [12]	8 x 20	95.83	95.83	85.25	85.25
7	Chandrasekharan and Rajagopalan [11]	8 x 20	71.88	72.79	58.33	56.73
8	Mosier and Taube [64]	10 x 10	85.29	85.29	70.59	70.59
9	Stanfel [77]	14 x 24	83.90	88.83	65.55	70.83
10	Chan and Milner [10]	15 x 10	96.00	96.00	92.00	92.00
11	King [37]	16 x 43	80.20	77.55	53.76	47.73
12	Mosier and Taube [65]	20 x 20	53.05	61.48	21.63	30.85
13	Carrie [9]	20 x 35	87.81	88.00	75.14	75.28
14	Boe and Cheng [6]	20 x 35	77.36	81.99	51.13	52.13
15	Kumar, Kusiak and Vannelli [41]	23 x 20	66.97	72.55	38.66	29.41
16	Chandrasekharan and Rajagopalan 1 [14]	24 x 40	100.00	100.00	100.00	100.00
17	Chandrasekharan and Rajagopalan 2 [14]	24 x 40	95.20	95.20	85.11	85.11
18	Chandrasekharan and Rajagopalan 3 [a] [14]	24 x 40	90.84	91.16	73.03	73.51
19	Chandrasekharan and Rajagopalan 5 [14]	24 x 40	77.31	84.84	20.42	44.37
20	Chandrasekharan and Rajagopalan 6 [14]	24 x 40	72.43	73.17	18.23	35.29
21	Chandrasekharan and Rajagopalan 7 [14]	24 x 40	69.33	75.57	17.61	34.88
22	Kumar and Vannelli [42]	30 x 41	68.14	86.43	33.46	57.69
23	Stanfel [77]	30 x 50	75.35	85.95	46.06	56.61
24	Stanfel [77]	30 x 50	62.92	79.12	21.11	42.27
25	Chandrasekharan and Rajagopalan [13]	40 x 100	95.07	95.10	83.92	84.03

[a]Data set 3 is as the same as data set 4.

Table 3.4: Performance comparison of our algorithm and ZODIAC

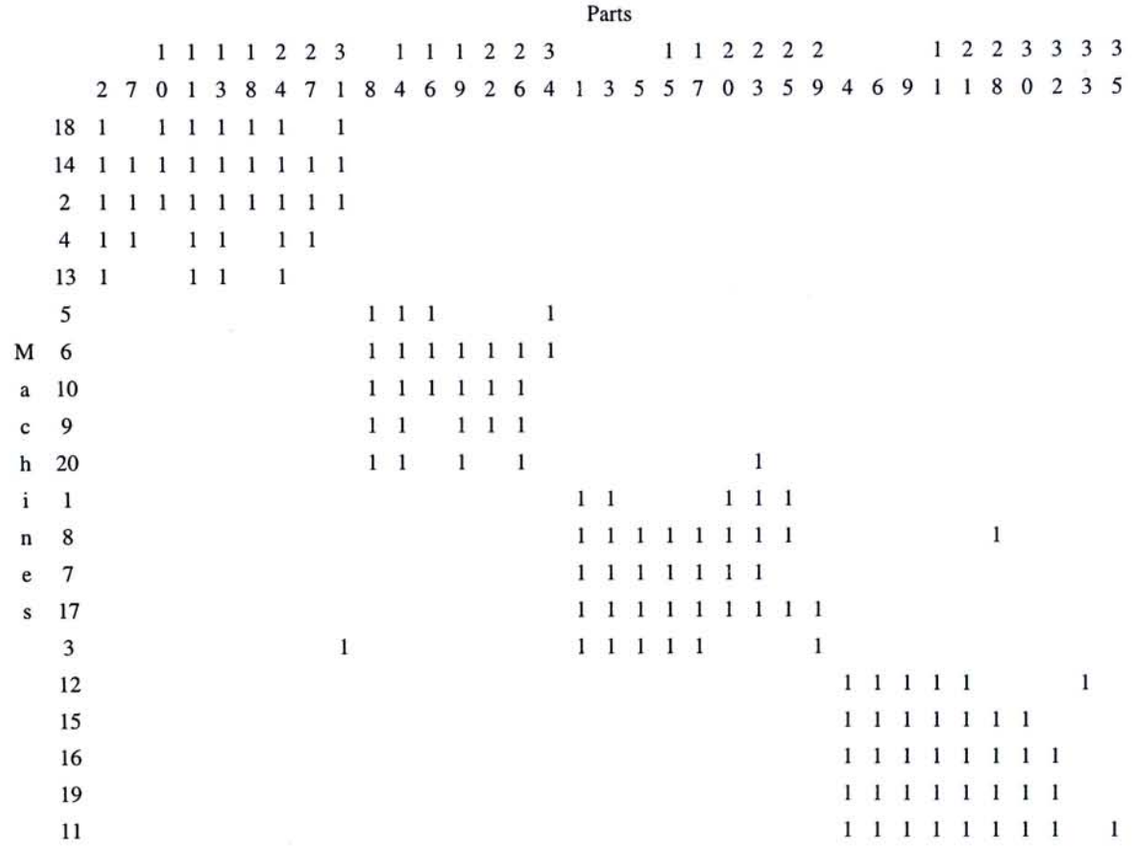

Figure 3.12: Rearranged matrix of problem 13 (20x35)

Cell/family	Machines	Parts
1	2,4,13,14,18	2,7,10,12,13,18,24,27,31
2	5,6,9,10,20	8,14,16,19,22,26,34
3	1,3,7,8,17	1,3,5,15,17,20,23,25,29
4	11,12,15,16,19	4,6,9,11,21,28,30,32,33,35

Table 3.5: Machine cells and part families formed

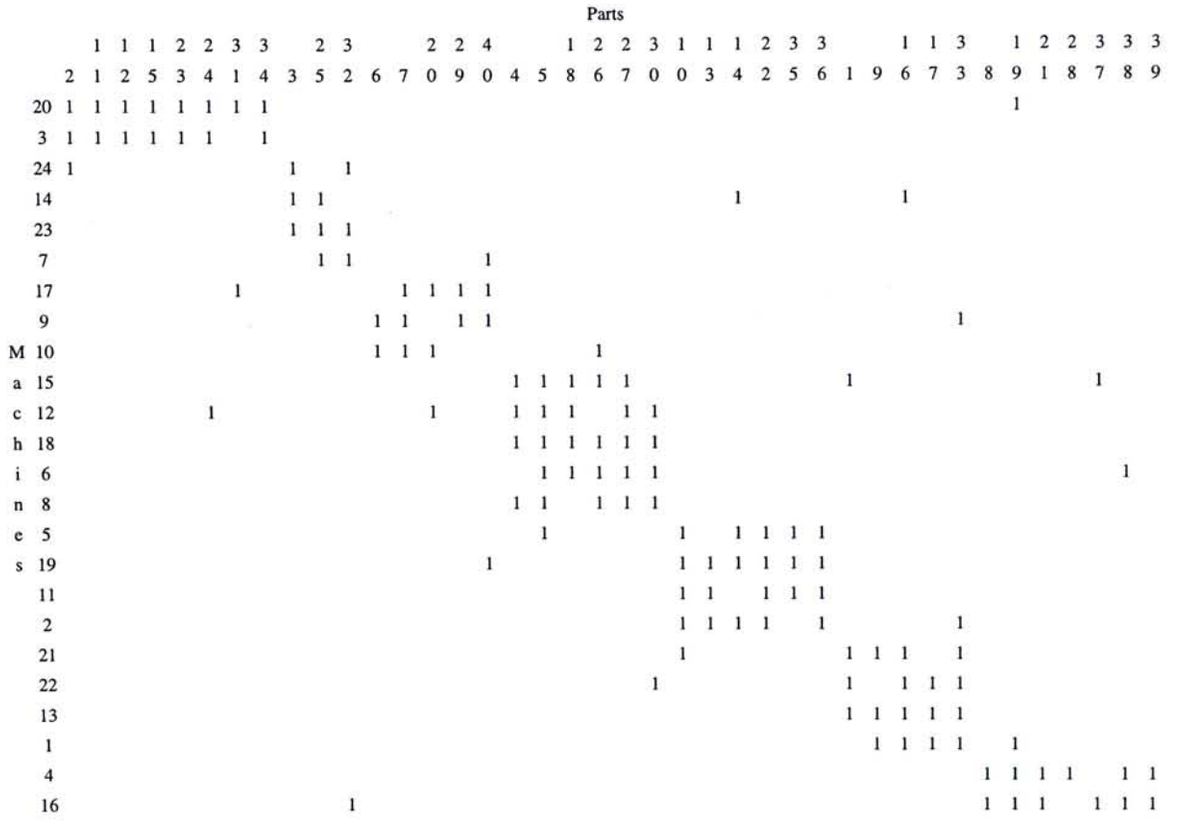

Figure 3.13: Rearranged matrix of problem 18 (24x40)

Cell/family	Machines	Parts
1	3,20	2,11,12,15,23,24,31,34
2	7,14,23,24	3,25,32
3	9,10,17	6,7,20,29,40
4	6,8,12,15,18	4,5,18,26,27,30
5	2,5,11,19	10,13,14,22,35,36
6	1,13,21,22	1,9,16,17,33
7	4,16	8,19,21,28,37,38,39

Table 3.6: Machine cells and part families formed

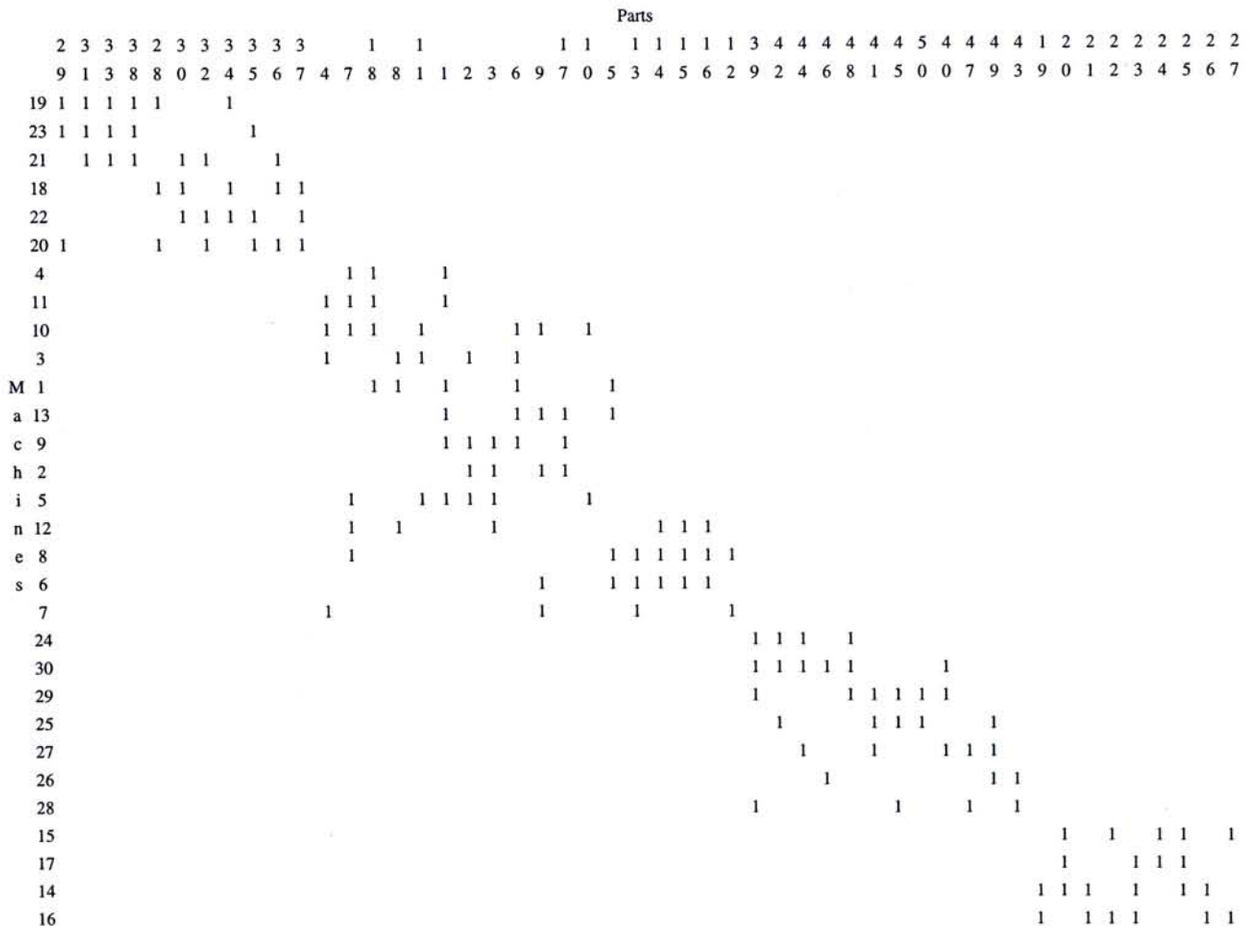

Figure 3.14: Rearranged matrix of problem 23 (30x50)

Cell/family	Machines	Parts
1	19,21,23	29,31,33,38
2	18,20,22	28,30,32,34,35,36,37
3	4,10,11	4,7,18
4	3	8,11
5	1,2,9,13	1,2,3,6,9,17
6	5	10
7	6,8,12	5,13,14,15,16
8	7	12
9	24,30	39,42,44,46,48
10	25,29	41,45,50
11	27	40,47,49
12	26,28	43
13	14,15,16,17	19,20,21,22,23,24,25,26,27

Table 3.7: Machine cells and part families formed

Chapter 4

A multi-chromosome GA for minimizing total intercell and intracell moves

4.1 Introduction

In reality, many objectives and constraints should be considered in designing a cellular manufacturing system; for example, material handling costs, machine utilizations, and number of cells formed. Generalized models deal with the cellular manufacturing problem more comprehensively.

There are many generalized models proposed in the literature. In this research, we consider the workload model proposed by Logendran [55]. This model aims at minimizing total intercell and intracell moves while also taking the machine utilization into consideration. As suggested by Logendran [55], the

intracell moves could be as important and unproductive as intercell moves.

Only a few research work considered the impact of intracell moves. Stanfel [77] proposed a mathematical model that included the concept of extraneous machine transitions to reflect intracell processing. Extraneous machines are machines in a machine cell not utilized by a part assigned to the same cell. The movements of parts from and to this machines incurred handling costs. The objective function of the Stanfel's approach is the total of inter-cell transitions and extraneous machine transitions. The problem with the Stanfel's approach is that extraneous machine transitions cannot truly represent intracell moves. Also the workload of machine is not taken into account in his approach. Logendran [55, 56] proposed a workload-based model focusing on minimizing the total moves that includes both intercell and intracell moves.

Furthermore, two important factors are often ignored by most literature: the sequence of operations and the layout of cells. These two factors are crucial in evaluating the intracell and intercell moves and can significantly affect the results. The workload-based model by Logendran [55, 56] takes these factors into account.

In reality, not just the movement of parts needs to be optimized. Other designing objectives are also the concerns of management. These objectives may include the workload of each workstation, the corresponding utilization rate, and the cell load variation, etc. These concerns can affect the choice made by management in selecting a cell configuration. Our algorithm will also take these factors into consideration.

In solving the workload model, Logendran proposed an algorithm involved four phases with two rules. In this chapter, we propose a genetic algorithm with multiple chromosome representation. Applying genetic algorithms to solve the workload model have been done by Gupta et al. [29, 30]. A major problem with their approach is the generation of illegal offsprings. Additional computational effects through mutation are required to handle illegal offsprings.

4.2 The model

In this research, the model proposed by Logendran [56] serves as the basic model. The objective function of the model is the total intracell and intercell moves. The underlying philosophy is clearly stated by Logendran:

> If a part is required to visit n cells ($n \geq 1$) to either partially or completely process its requirements, then it contributes to $(n-1)$ intercell moves in the total intercell moves equation. .... If a part is required to visit m machines ($m \geq 1$) dedicated to a cell as a portion of its processing requirements or in its entirety, then it contributes to $(m-1)$ intracell moves in the total intracell moves equation.

The sequence of operations and layout of cells are important factors in evaluating the movement but are commonly neglected by researchers. Consider the three cells in Figure 4.1, machines 1 and 3 are assigned to cell 1 while machine 2 in cell 2 and machine 4 in cell 3. Assume that a part requires to visit all the four machines to complete its processing requirements. If the operation

sequence is not taken into account when the cost is being minimized, there are one intracell move and two intercell moves. However, in reality, processing of parts requires an operation be performed before another operation. To truly reflect this situation, we further assume that a part requires the following processing sequence: M1→M2→M3→M4. As a result, this part requires three intercell moves and no intracell move. This example clearly demonstrates the importance of operation sequences.

Two cellular layouts are considered in this research: the linear single-row cellular layout and the linear double-row cellular layout. For simplicity, the linear single-row layout has been referred to as layout 1 (Figure 4.1) and the linear double-row layout as layout 2 (Figure 4.2).

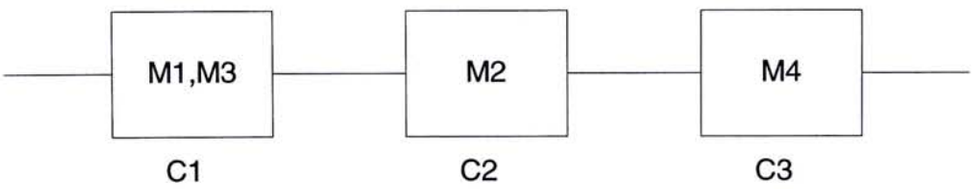

Figure 4.1: Linear single-row cellular layout

Consider layout 1. If the distances between cells are equal, then the distance travelled from cell 1 to 3 will be two times as from cell 1 to cell 2. For layout 2, the distance travelled from cell 1 to 3 and cell 2 to 4 is $\sqrt{2}$ times the distance travelled between any adjacent two cells.

To accurately analyze cellular manufacturing problem, intracell movements, sequence of operations, and layout of cells should be taken into consideration.

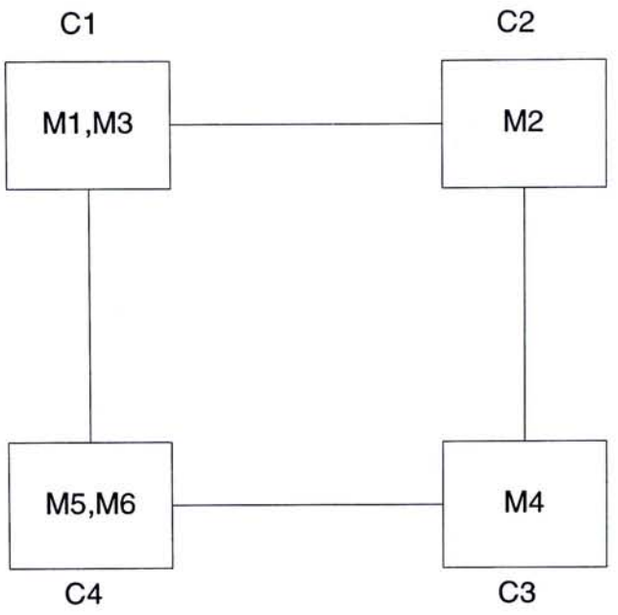

Figure 4.2: Linear double-row cellular layout

The total moves for the two layouts can be represented by the following equations:

Layout 1:

$$Total\ moves\ = \theta_1 * \sum_{i=1}^{p} \sum_{k=1}^{k_i-1} |c_k - c_{k+1}| + \theta_2 * \sum_{i=1}^{p} m_i \tag{4.1}$$

Layout 2:

$$Total\ moves\ = \theta_1 * \sum_{i=1}^{p} \sum_{k=1}^{k_i-1} \alpha_{k,k+1} + \theta_2 * \sum_{i=1}^{p} m_i \tag{4.2}$$

where:

c_k = the cell number in which operation k is performed on part i

c_{k+1} = the cell number in which operation $(k+1)$ is performed on part i

k_i = the total number of operations to be performed on part i

$$\alpha_{k,k+1} = \begin{cases} \sqrt{2} \text{ if } |c_k - c_{k+1}| = 2 \\ \\ 1 \text{ otherwise} \end{cases}$$

c = the number of cells

p = the number of parts

m_i = the total number of intracell moves performed by part i

θ_1 = the fractions representing the weights attributed to the intercell moves

θ_2 = the fractions representing the weights attributed to the intracell moves

As suggested by Logendran [55], the weight assigned to intracell moves may not be as high as that assigned to intercell moves. In this research, we will follow the value used by Logendran. That is the weight assigned to intracell moves is 0.3 and the weight assigned to intercell moves is 0.7.

One of the parameters required by the model is the number of cells. In general, as the number of cells formed increases, the effect of intercell moves increases and the effect of intracell moves decreases. In order to make the model realistic, we assume there will be at least two cells formed. On the other hand, the upper limit on the number of cells formed is arbitrary. The model can be evaluated in cell number equals to three, four, five, and so on. This actually provides different scenarios to a plant manager or designer who evaluates the solutions based on the constraints he/she faces. As pointed out by Gupta et al. [30], most of the manufacturing firms in US used six or less cells; hence, it is possible to evaluate all the alternatives. The decision for the number of cells is

primarily based on factors such as workforce, the space of shop floor, budgetary limitations, etc.

Once the machine cells are formed, parts are assigned to cells based on the accumulated processing time in all cells. The total accumulated processing time is the sum of all processing times of a part in each of the workstations of the cell. The part is assigned to the cell of highest value. Ties are broken arbitrary. When the part families are also formed, the machine utilization can be calculated. Machine utilization is determined as a ratio between the workload and the machine capacity. If the workload assigned to a machine exceeds its capacity, we allow multiple machines to be allocated. If we refer a machine type as a workstation. The utilization rate can be represented as follows.

$$U_j = \frac{\sum_{i \in c_j} t_{ij}}{C} \tag{4.3}$$

where

U_j = utilization rate of workstation j

t_{ij} = processing time of each part i in workstation j

C = available capacity of machines (hours)

c_j = cell to which workstation j is assigned

4.3 Solution techniques to the workload model

Two approaches were proposed in the literature to solve the workload model, the Logendran's four phases algorithm [55] and Gupta et al.'s genetic algorithm

[29].

4.3.1 Logendran's original approach

The original approach involves four phases: cell representation phase, clustering phase, improvement phase, and assignment phase. In the cell representation phase, different key workstations are identified. The number of key workstations equals to the number of cells formed. The key workstation is the seed for each cell. In phase 2, the remaining workstations are assigned to the cells based on the total moves resulted. A workstation is assigned to a cell that causes minimum total moves. This phase ends when all the workstations are assigned to cells. In the improvement phase, further reduction in total moves is attempted. Each workstation will be removed from the cell to which it was assigned and added to every other cell. The workstation will be assigned to the cell that gives the highest reduction. Finally, in the last phase, parts are assigned based on the accumulated processing time in all cells.

As suggested in by Del Valle et al.[23], the step in selecting the key workstations is important. A poor selection criterion will increase the computational overhead in the improvement phase. Two rules may be employed in selecting the key workstations. Rule 1 selects workstation with the highest total workload per machine. Rule 2 selects the workstation with the maximum number of parts.

One of the weakness of the algorithm is that the complexity of assigning a

machine from cell to cell in step 3 increases as the problem size increases [23]. For some real size problems, the cost to obtain a solution will be too high.

4.3.2 Standard representation - the GA approach

The cell formation problem is a combinatorial problem [53]. Heuristic approaches are required to solve the problem efficiently. Genetic algorithms seem to be a good candidate to solve combinatorial problems due to their ability to perform a parallel search and their robustness [53]. Applying genetic algorithms to solve the workload model was first proposed by Gupta et al. [29, 30].

In Gupta et al.'s implementation [29], standard (or group number) representation was employed. The value of a gene (allele) represents the cell number with the position of the gene corresponding to the machine number. For example, chromosome (2,3,1,1,2) represents that machine 1 is assigned to cell 2, machine 2 assigned to cell 3, machine 3 assigned to cell 1, and so on. The length of the chromosome is the total number of machines. The crossover operator used is the single point crossover. The crossing point in a chromosome is chosen randomly. The portions of the chromosomes after the crossing point are exchanged to produce the offsprings.

One weakness of this representation is the occurrence of empty cell and therefore illegal offsprings. Consider the crossover operation in Figure 4.3. The number of cells is assumed to be three. However, after crossover, cell 3 is empty

in offspring 1 and cell 2 is empty in offspring 2. This violates the constraint that each cell at least contains one machine. The empty cell is no longer a cell.

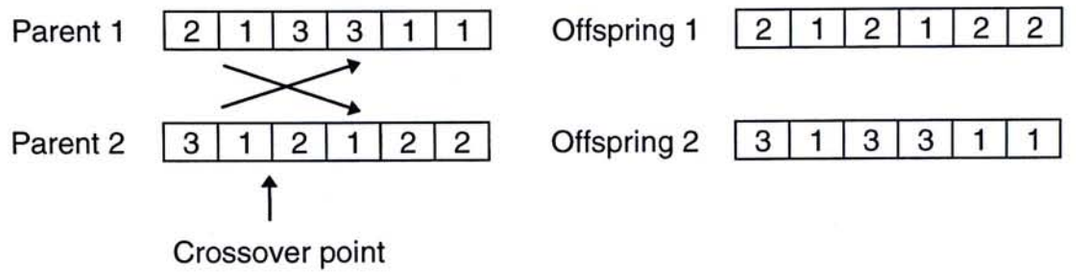

Figure 4.3: Occurrence of empty cell after crossover

To overcome this problem, Gupta et al. [29] introduced a mutation operator. Two random integers r_1 and r_2 are selected such that $1 \leq r_1 \leq m$ and $1 \leq r_2 \leq c$. The algorithm then removes machine number r_1 from the chromosome and assigns it to cell r_2. This process is repeated until no empty cell exists. Such a mutation operator increases the computational overhead and affects the performance of the genetic algorithm.

In this thesis, we develop a chromosome representation which will help reduce computational effort. The use of genetic algorithms replaces the first three steps in Logendran's approach and the solution is independent of the choice of the key workstations. The individual chromosome with the lowest total moves gives the solution to the cell formation problem. After the formation of cells, parts are assigned using the fourth step of the Logendran's method.

4.3.3 Multi-chromosome representation

Traditionally, the solution of a genetic algorithm is encoded in one chromosome. The gene of a chromosome can be a binary, an integer, a real number, or an alphabet. The choice of these encoding schemes depends on the domain of the problem. For example, for a sequencing problem like TSP, it is most natural to use an integer as a gene.

A genetic algorithm uses some decoding schemes to interpret the structure of a chromosome. In other words, a GA tries to extract information stored in a chromosome and provides a solution to the original problem. Ideally, the solution is directly represented in a chromosome and requires no decoding. However, some problems are difficult to have a direct representation. Even if it can be represented, specific or problem-dependent crossover operators are required.

Bruns compared the direct and indirect representations for the production scheduling problem [7]. The direct representation involves incorporation of problem-specific knowledge of the application domain in a genetic algorithm. The introduction of the expanded representation requires the definition of new domain-dependent crossover and mutation operators. He concluded in his paper that the addition of relevant domain information made a GA operate on the entire search space and produced very promising results.

In animal genetic, the number of chromosomes in one cell is always larger than 1. For example, there are 23 pairs of chromosomes in a somatic human

cell. Naturally, genes that encode very different information will be located at different chromosomes. Similar function genes will be placed together. For instance, most of the genes on X and Y chromosomes are related to the sex of a human. Encoding different characteristics to separate chromosomes may have advantages. In this research, we suggest a multi-chromosome representation that is considered to be more natural to the cell formation problem. Juliff [35] suggested a multi-chromosome GA to solve the pallet loading problem. He stated that the multi-chromosome GA outperformed its single-chromosome counterpart. He also concluded that his GA can sample the search space more productively.

In our representation, each individual contains two chromosomes. Table 4.1 shows the details, where m is the number of machines.

Chromosome	length	crossover used
machine sequence	m	edge-recombination operator
cell boundary	$m-1$	edge-recombination operator

Table 4.1: The chromosomes used

Each gene in a machine sequence chromosome represents a machine number and therefore, the value of a gene is an integer. The length of a chromosome equals to the number of machines. The crossover operator used is the edge-recombination operator. Edge-recombination operator transfers 95% of edge (information) from parent to offspring and is considered the most efficient operator for general sequencing problem [61].

Another chromosome is the cell boundary chromosome for a given machine

sequence. Its task is to partition machines into different machine cells. This chromosome groups the first x machines into a common cell, then the following y machines to another cell, and so on. All the machines will be assigned to the predefined number of cells. Gene 1 indicates the edge between machines 1 and 2, gene 2 indicates the edge between machines 2 and 3, and so on. Each gene in the cell boundary chromosome is an integer number. Assume the number of cells formed is c, then the integer value smaller than c represents the cell boundary. For example, Figure 4.4 is a possible individual assumed three cells to be formed:

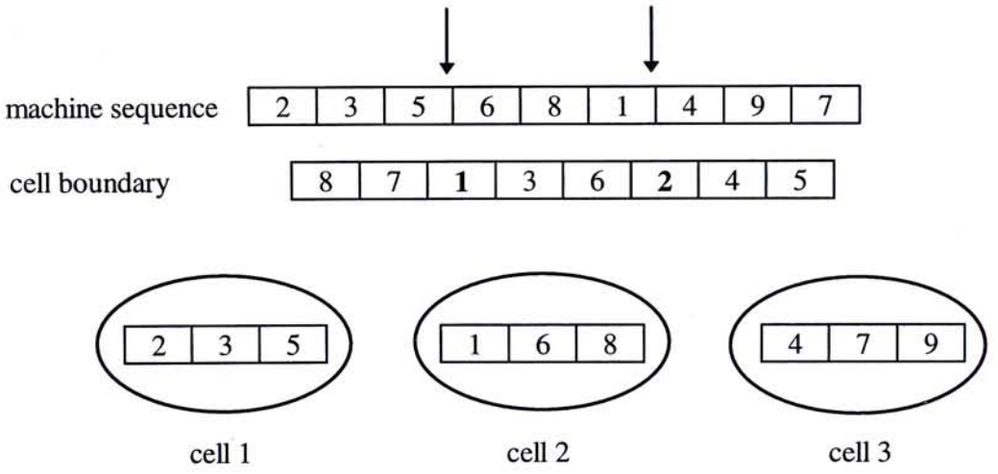

Figure 4.4: Interpretation of information in chromosomes

The third gene (1) and the sixth gene (2) of the cell boundary chromosome indicate the cutting points of the machine sequence. As a result, machines 2, 3, and 5 are assigned to cell 1, machines 1, 6, and 8 to cell 2, and cell 3 contains machines 4, 7, and 9. Compared to the representation proposed by Gupta et

al. [30], this representation does not generate illegal offsprings and therefore reduces the computational overhead required for restoration (see Section 4.3.2).

The proposed representation also facilitates a parallel search. A parallel search can result in a better solution in less computational time. The existence of several evolving chromosomes allows good chromosomes not to be disrupted while trying to optimize another chromosome. In addition, the use of machine sequence allows similar machines keep together during the search process and a kind of linkage among objects can be developed. This actually provides more information to GA and guides the search in a more favorable direction.

Consider the example in Figure 4.5. Assume that three machines (machines 2, 3 and 5) should be assigned to one cell. Chromosomes 1 and 3 represent the same individual that machines 2 and 3 are grouped together and chromosomes 2 and 4 represent an individual that machines 3 and 5 are together. After crossover, in our representation, machines 2, 3, and 5 may appear together in one cell. However, in standard representation, one crossover cannot give a satisfactory result. This is because the association of cell numbers to the machine restricts the GA to combine information in each chromosome.

In addition, our representation has a potential capability to extend to dynamic cell formation. In some generalized models, the number of cells formed is not a pre-defined number, for example, the model proposed by Askin and Subramanian [3]. Their algorithm determines the best cell configuration as well as the best number of cells. Since we have not imposed any restriction to the number of cells formed, we can use our representation to handle this extension.

On the contrary, the representation proposed by Gupta et al. strictly requires the number of cells be pre-defined.

Finally, handling one more chromosome in each individual is not very costly in terms of computation. In fact, such GAs allow parallel processing, which can further reduce computational efforts involving an additional chromosome in each individual. As a result, this multi-chromosome representation seems particularly suitable for solving generalized cell formation model. It is more natural and capture more information about the problem that actually aids the searching.

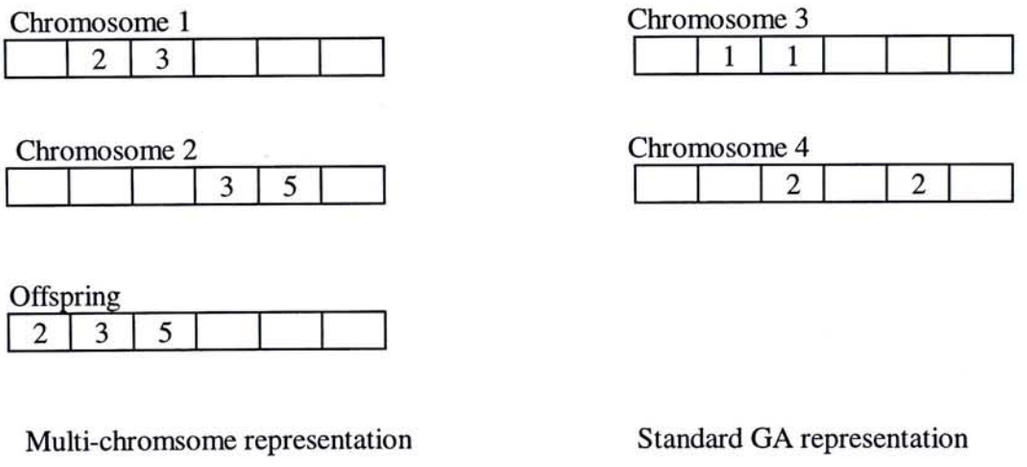

Figure 4.5: Problem of association of cell number to machines

Our implementation is based on a genetic package called GENITOR [86]. The fitness function is the total moves suggested by Logendran. Both chromosomes are randomly generated in the initial population. The parents are selected using fitness ranking. The processes of selection, crossover, and

replacement are continued until a maximum number of generation reached.

4.4 Comparative Study

We solved the three problems used in the Logendran's paper and one large problem in the Gupta's paper in order to establish the performance of the algorithm. The results are shown in the tables below. The tables include the assignment of workstations and parts, and the utilization rate of individual workstation. In addition, the average utilization rates were calculated. For the three problems identified by Logendran, the model was evaluated for two, three, and four cells. For the large problem proposed by Gupta et al., four, five, and six cells were considered. The capacity of each machine was assumed to be eight hours. If the solution results in lower total moves or higher workstation utilization, we consider it to be a better one. The results of the four problems are discussed separately.

4.4.1 Problem 1

This problem was considered by Balakur and Steudel [4]. It has seven parts and five machines. Table 4.2 shows the workstation-part load matrix. The table indicates the workload for workstation W2 is 10.5 hours and it exceeds the capacity of one machine. Therefore, two machines are allocated. The other workstations have a load less than 8 hours and only one machine in such workstations.

	Parts							Total workload	No. of
Workstation	P1	P2	P3	P4	P5	P6	P7	on workstation (h)	machines
W1		0.5			5.0		1.5	7.0	1
W2	2.5	2.0		4.5		1.5		10.5	2
W3		2.5			3.5	0.5	0.5	7.0	1
W4	2.5			1.0		0.5		4.0	1
W5			3.0		1.0		2.0	6.0	1

Table 4.2: Workstation-part load matrix for Problem 1

Tables 4.3 and 4.4 present the results and the comparison, respectively. Our genetic algorithm generated better results than Logendran and Gupta. The results of Gupta's approach are listed in Tables 4.12 and 4.13. For three cells, both the Gupta's and our solutions obtained a total number of moves equal to 5.0 which was less than the Logendran's solution. But our average utilization was higher than the Gupta's one. For four cells, not only the average utilization rates were better but the total moves obtained were the least among the three algorithms.

Logendran targeted the minimum acceptable value for utilization to 50%. However, all the algorithms failed to reach this value for all workstations. In addition, W4 had 0% utilization rate in Logendran's and Gupta's solution when 3 and 4 cells were formed. This was a possible waste of resource and would cause workload imbalance. However, in our case, 31% was obtained.

4.4.2 Problem 2

This problem was originally presented by Tabucannon and Ojha [82] and modified by Logendran. It has 14 parts and 7 workstations. Workstation W7

Cells, Layout, Moves	Workstation in cell Ci $i = 1, 2, 3, ..., N$				Parts in cell Ci $i = 1, 2, 3, ..., N$				Utilization of workstation Wi $i = 1, 2, 3, ..., M$				
	C1	C2	C3	C4	C1	C2	C3	C4	W1	W2	W3	W4	W5
2,1,3.8	1,2,3 4	5			1,2,4 5,6,7	3			0.88	1.31	0.88	0.50	0.38
2,2,3.8	1,2,3 4	5			1,2,4 5,6,7	3			0.88	1.31	0.88	0.50	0.38
3,1,5.0	5	1,2,3	4		3	2,4,5 6,7	1		0.88	1.00	0.88	0.31	0.38
3,2,5.0	4	1,2,3	5		1	2,4,5 6,7	3		0.88	1.00	0.88	0.31	0.38
4,1,6.9	4	2	1,3	5	1	4,6	2,5,7	3	0.88	0.75	0.81	0.31	0.38
4,2,6.49	1,3	2	4	5	2,5,7	4,6	1	3	0.88	0.75	0.81	0.31	0.38

Table 4.3: Result of Problem 1

No. of cells, Layout	Total moves				Average utilization of workstation			
	Logendran (rule 1)	Logendran (rule 2)	Gupta	GA	Logendran (rule 1)	Logendran (rule 2)	Gupta	GA
2,1	4.6	3.8	3.8	3.8	0.694	0.659	0.657	0.659
2,2	4.6	3.8	3.8	3.8	0.694	0.659	0.657	0.659
3,1[a]	6.8	7.1	5.0	5.0	0.620	0.559	0.559	0.588
3,2[a]	5.98	5.87	5.0	5.0	0.620	0.559	0.585	0.588
4,1[a]	10.4	13.2	7.0	6.9	0.520	0.520	0.520	0.550
4,2[a]	7.36	7.36	6.78	6.49	0.520	0.520	0.434	0.550

[a]Our approach is better

Table 4.4: Comparison of different approaches for Problem 1

has 3 machines and W1 and W5 have two machines. The remainders make up of one machine only. Table 4.5 shows the workload matrix.

Work-	Parts														Load	# of
station	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	(h)	m
W1		0.69			2.42			2.44	2.48					2.72	10.75	2
W2	0.50			0.61	0.90	2.09	1.35								5.45	1
W3										2.50	3.03	0.71	1.61		7.85	1
W4		3.10					1.35		1.03		0.58	0.99			7.05	1
W5		1.22						4.45						3.84	9.51	2
W6	0.50					4.55		2.26							7.31	1
W7	0.55			4.74	3.61	1.47			3.87	4.68					18.92	3

Table 4.5: Workstation-part load matrix for Problem 2

Tables 4.6, 4.7, and 4.8 summarize the results for this data set. For two cells configuration, the Logendran's approach employing rule 1 obtained the best known solution. Gupta's genetic algorithm generated a solution with the same amount of total moves but W1 failed to achieve the targeted minimum utilization rate (50%) and the average utilization of all workstations was lower. In our case, most of the utilization rates reached the minimum value. For three and four cells configuration, our approach generated better solution than Logendran's. The total moves were less and with a higher average utilization of workstations. Also, fewer workstations failed to meet the targeted utilization rate. For this data set, all the results produced by our algorithm were comparable to the best known solution available in the literature.

Cells, Layout, Moves	Workstation in cell Ci $i = 1, 2, 3, ..., N$				Parts in cell Ci $i = 1, 2, 3, ..., N$			
	C1	C2	C3	C4	C1	C2	C3	C4
2,1,6.3	1,2,4,5,6,7	3			1,2,3,4,5,6,7,8 9,10,12,14,15	11,13		
2,2,6.3	1,2,4,5,6,7	3			1,2,3,4,5,6,7,8 9,10,12,14,15	11,13		
3,1,7.5	1,5	2,4,6,7	3		2,8,14	1,3,4,5,6 7,9,10,12	11,13	
3,2,7.5	1,5	2,4,6,7	3		2,8,14	1,3,4,5,6 7,9,10,12	11,13	
4,1,9.4	3	2,4,6,7	1	5	11,13	1,3,4,5,6 7,9,10,12		2,8,14
4,2,8.88	2,6,7	1,5	3	4	1,4,5,6,7,9,10	2,8,14	11,13	3,12

Table 4.6: Result of Problem 2 (part 1)

No. of cells, Layout, Total moves	Utilization of workstation Wi $i = 1, 2, 3, ..., M$						
	W1	W2	W3	W4	W5	W6	W7
2,1,6.3	1.34	0.68	0.98	0.88	1.19	0.63	2.36
2,2,6.3	1.34	0.68	0.98	0.88	1.19	0.63	2.36
3,1,7.5	0.73	0.68	0.58	0.81	1.19	0.63	2.36
3,2,7.5	0.73	0.68	0.58	0.81	1.19	0.63	2.36
4,1,9.4	0.00	0.68	0.58	0.81	1.19	0.63	2.36
4,2,8.88	0.73	0.68	0.58	0.51	1.19	0.63	2.36

Table 4.7: Result of Problem 2 (part 2)

No. of cells, Layout	Total moves				Average utliization of workstation			
	Logendran (rule 1)	Logendran (rule 2)	Gupta	GA	Logendran (rule 1)	Logendran (rule 2)	Gupta	GA
2,1[a]	6.3	6.7	6.3	6.3	0.720	0.699	0.702	0.720
2,2[a]	6.3	6.7	6.3	6.3	0.720	0.699	0.702	0.720
3,1	8.9	8.9	7.5	7.5	0.676	0.609	0.635	0.635
3,2	8.08	8.08	7.5	7.5	0.676	0.609	0.635	0.635
4,1	12.3	13.7	9.4	9.4	0.527	0.550	0.583	0.583
4,2	10.66	9.39	8.88	8.88	0.527	0.576	0.592	0.592

[a]Our approach is better than Gupta et al.'s solution

Table 4.8: Comparison of different approaches for Problem 2

4.4.3 Problem 3

This problem was proposed by King and Nakornchai (1982) [38] and then considered by Waghodekar and Sahu (1984) [84]. As the original data set did not contain processing time. Logendran randomly generated data to fulfill the requirement of the workload model. The workstation-part load matrix is presented in table 4.9. This problem consists of 7 parts and 5 workstations and only W4 with a total workload larger than the capacity of one machine and therefore two machines exist in W4 and the remainder consists of only one machine each.

	Parts							Total workload	No. of
Workstation	P1	P2	P3	P4	P5	P6	P7	on workstation (h)	machines
W1		0.55		4.74			1.35	6.64	1
W2			1.22		3.61			4.83	1
W3	0.50	1.69		2.42			1.35	4.96	1
W4	0.51		3.10			4.55		8.16	2
W5			0.61	0.90	2.09	1.47		5.07	1

Table 4.9: Workstation-part load matrix for Problem 3

Results are tabulated in Table 4.10 and 4.11. Although our solutions are comparable to Gupta's approach, they are better than Logendran's. The total moves were significantly reduced. For four cells with layout 1, W5 results in 0% utilization in Logendran's solution while our approach still reached 26%.

Cells, Layout, Moves	Workstation in cell Ci $i = 1, 2, 3, ..., N$				Parts in cell Ci $i = 1, 2, 3, ..., N$				Utilization of workstation Wi $i = 1, 2, 3, ..., M$				
	C1	C2	C3	C4	C1	C2	C3	C4	W1	W2	W3	W4	W5
2,1,3.5	2,4,5	1,3			2,4,7	1,3,5,6			0.83	0.60	0.56	1.02	0.52
2,2,3.5	2,4,5	1,3			2,4,7	1,3,5,6			0.83	0.60	0.56	1.02	0.52
3,1,4.3	1,3	4,5	2		2,4,7	1,3,6	5		0.83	0.45	0.56	1.02	0.26
3,2,4.3	2	4,5	1,3		5	1,3,6	2,4,7		0.83	0.45	0.56	1.02	0.26
4,1,5.5	2	4,5	3	1	5	1,3,6	2	4,7	0.76	0.45	0.09	1.02	0.26
4,2,5.5	2	1	3	4,5	5	4,7	2	1,3,6	0.76	0.45	0.09	1.02	0.26

Table 4.10: Result of Problem 3

No. of cells, Layout	Total moves				Average utliization of workstation			
	Logendran (rule 1)	Logendran (rule 2)	Gupta	GA	Logendran (rule 1)	Logendran (rule 2)	Gupta	GA
2,1	3.5	3.5	3.5	3.5	0.604	0.604	0.604	0.604
2,2	3.5	3.5	3.5	3.5	0.604	0.604	0.604	0.604
3,1	6.8	6.8	4.3	4.3	0.496	0.494	0.522	0.522
3,2	5.57	5.57	4.3	4.3	0.496	0.494	0.522	0.522
4,1	9.8	11.2	5.5	5.5	0.362	0.360	0.414	0.414
4,2	7.64	7.64	5.5	5.5	0.414	0.412	0.414	0.414

Table 4.11: Comparison of different approaches for Problem 3

4.4.4 Problem 4

This problem was proposed by Gupta et al.[29]. The purpose of this data set is to demonstrate the generalized nature of genetic algorithms. As the sizes of the three problems considered by Logendran are relatively small, it is difficult

Problem	Cells, Layout, Moves	Workstation in cell Ci $i = 1, 2, 3, ..., N$				Parts in cell Ci $i = 1, 2, 3, ..., N$			
		C1	C2	C3	C4	C1	C2	C3	C4
1	2,1,3.8	5	1,2,3,4			3,7	1,2,4,5,6		
	2,2,3.8	5	1,2,3,4			3,7	1,2,4,5,6		
	3,1,5.0	5	2,3,4	1		3,7	1,2,4,6	5	
	3,2,5.0	5	2,3,4	1		3,7	1,2,4,6	5	
	4,1,7.2	5	1,3	2	4	3,7	2,5	1,4,6	
	4,2,6.78	1	3,5	2	4	5	2,3,7	1,4,6	
2	2,1,6.3	2,3,4, 6,7	1,5			1,3,4,5,6,7, 9,10,11,12,13	2,8,14		
	2,2,6.3	2,3,4, 6,7	1,5			1,3,4,5,6,7, 9,10,11,12,13	2,8,14		
	3,1,7.5	3	2,4,6,7	1,5		11,13	1,3,4,5,6, 7,9,10,12	2,8,14	
	3,2,7.5	3	2,4,6,7	1,5		11,13	1,3,4,5,6, 7,9,10,12	2,8,14	
	4,1,9.4	5	1	2,4,6,7	3	2,8,14		1,3,4,5,6, 7,9,10,12	11,13
	4,2,8.88	2,6,7	1,5	4	3	1,4,5,6, 7,9,10	2,8,14	3,12	11,13
3	2,1,3.5	1,3	2,4,5			2,4,7	1,3,5,6		
	2,2,3.5	1,3	2,4,5			2,4,7	1,3,5,6		
	3,1,4.3	1,3,	4,5,	2		2,4,7	1,3,6	5	
	3,2,4.3	2	4,5	1,3		5	1,3,6	2,4,7	
	4,1,5.5	2	4,5	3	1	5	1,3,6	2	4,7
	4,2,5.5	2	1	3	4,5	5	4,7	2	1,3,6

Table 4.12: Gupta's solutions for Problems 1, 2, and 3 (part 1). Reproduced from Gupta et al. (1996) [29]

Problem	No. of cells, Layout, Total moves	Utilization of workstation Wi $i = 1, 2, 3, ..., M$						
		W1	W2	W3	W4	W5	W6	W7
1	2,1,3,8	0.69	1.31	0.81	0.50	0.63		
	2,2,3,8	0.69	1.31	0.81	0.50	0.63		
	3,1,5,0	0.63	1.31	0.38	0.50	0.63		
	3,2,5,0	0.63	1.31	0.38	0.63	0.63		
	4,1,7,2	0.69	1.06	0.75	0.00	0.63		
	4,2,6.78	0.63	1.06	0.38	0.00	0.63		
2	2,1,6.3	0.73	0.68	0.98	0.88	1.19	0.63	2.36
	2,2,6.3	0.73	0.68	0.98	0.88	1.19	0.63	2.36
	3,1,7.5	0.73	0.68	0.58	0.81	1.19	0.63	2.36
	3,2,7.5	0.73	0.68	0.58	0.81	1.19	0.63	2.36
	4,1,9.4	0.00	0.68	0.58	0.81	1.19	0.63	2.36
	4,2,8.88	0.73	0.68	0.58	0.51	1.19	0.63	2.36
3	2,1,3.5	0.83	0.60	0.56	1.02	0.52		
	2,2,3.5	0.83	0.60	0.56	1.02	0.52		
	3,1,4.3	0.83	0.45	0.56	1.02	0.26		
	3,2,4.3	0.83	0.45	0.56	1.02	0.26		
	4,1,5.5	0.76	0.45	0.09	1.02	0.26		
	4,2,5.5	0.76	0.45	0.09	1.02	0.26		

Table 4.13: Gupta's solutions for Problems 1, 2, and 3 (part 2). Reproduced from Gupta et al. (1996) [29]

to make any valid conclusions about the performance of the algorithms. This problem is relatively large, Table 4.14 shows the load matrix. It has 30 parts and 15 workstations. As the number of workstations are increased, it is reasonable to increase the number of cells for evaluation as well. We followed Gupta's approach and considered four, five, and six cells. Tables 4.15, 4.16, and 4.19 present the result of our approach and the comparison with Gupta's solution, while Tables 4.17 and 4.18 are the results of Gupta's. We can see for all the cells and layouts, our approach generated better results. Most of the solutions were of lower total moves and higher average utilization rates and most of the workstations reached the minimum targeted utilization rate. Better results were produced was probably due to the fact that our representation successfully captured the important features of the cell formation problem and enabled the GA to explore the search space more effectively.

4.5 Bi-criteria Model

In an actual manufacturing environment, many factors may affect the efficiency and productivity of a cellular manufacturing system. The nature of these factors are always conflicting. In other words, optimizing one factor may result in destruction of the other. To recognize that the cell formation problem involves trade-off among conflicting objectives, multi-objective models were developed.

Wei and Gaither [85] proposed a model with four objective functions for optimization. The four objectives are minimization of bottleneck cost,

		Workstations														
		W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
	P1		0.1	0.9			1.2	0.1	0.6			0.3				
	P2	0.8		0.8			1.1		1.2			0.4			1.6	
	P3				0.4				1.7							
	P4					0.9		1.3						1.4		
	P5	1.2		1.8				0.2								0.3
	P6			0.7						0.9						
	P7	0.5				1.5	1.0		1.4	0.4	1.5			0.3		
	P8		1.7		0.6	0.9							0.1			
	P9	1.7				0.1				1.3	1.4		1.5		1.3	0.4
	P10	1.8	1.3													
	P11			0.5	0.8				2.0					1.7		1.3
	P12	2.0								0.8			1.5	1.6		
	P13						0.6			1.2	1.8		0.2	1.5		0.3
P	P14		1.7			0.6										0.5
a	P15						1.9								1.0	
r	P16			0.2			0.4	1.9						1.3		
t	P17					0.5								1.2		
s	P18										0.7			0.4		
	P19		0.8	1.4					1.8					0.4		
	P20			0.3	1.1	0.6	2.0				0.2					
	P21			0.3	1.7				1.0				1.1	0.7		
	P22	0.5			1.4						1.7			0.1		1.9
	P23				1.7	1.2						1.7				
	P24									1.3			1.4			1.5
	P25		1.1									1.8				0.2
	P26									0.9	1.5					
	P27		1.9		0.3							1.5			0.3	0.3
	P28					0.7		1.8		1.4		1.3				
	P29			1.7												
	P30						1.3			1.9	0.7		1.8		1.6	
	Workload	8.5	8.6	8.6	8.0	7.0	9.5	5.3	9.7	10.1	9.5	7.0	7.6	10.6	5.8	6.7
	No. of m	2	2	2	1	1	2	1	2	2	2	1	1	2	1	1

Table 4.14: Workstation-part load matrix for problem 4 (In order to fit to a page, the orientation is reversed with parts in rows and workstations in columns)

Cells, Layout, Moves	Workstation in cell Ci $i = 1, 2, 3, ..., N$						Parts in cell Ci $i = 1, 2, 3, ..., N$					
	C1	C2	C3	C4	C5	C6	C1	C2	C3	C4	C5	C6
4,1, 37.30	14	11	2,3,4,5, 6,7,8,9, 10,12,13	1			15	25	1-9,11-14, 16-24, 26-30	10		
4,2, 34.67	11	14	15	1,2,3,4, 5,6,7,8 9,10, 12,13			25	15	24	1-14, 16-23, 26-30		
5,1, 44.20	14	11	3,4,5,6, 7,8,9,10, 12,13,15	1	2		15	25	1-7,9, 11-13, 16-24, 26, 28-30	10	8,14, 27	
5,2, 37.07	1,3,4,5, 6,7,8,9, 10,12, 13	11	14	15	2		1-7, 9-13, 16-23, 28-30	25	15	24	8,14, 27	
6,1, 54.90	1	2	3,4,5,6 7,8,11	9,10,12 13	15	14	10	8,14, 27	1-5,7,11, 16,19-21, 23,25,26, 28,29	6,9,12, 13,17, 18,22, 30	24	15
6,2, 39.47	2	1	15	14	11	3,4,5, 6,7,8,9 10,12 13	8,14,27	10	24	15	25	1-7,9, 11-13, 16-23, 26, 28-30

Table 4.15: Result of Problem 4 (part 1)

Cells, Layout, Moves	Utilization of workstation Wi $i = 1, 2, 3, ..., M$														
	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
4,1,37.30	0.23	0.78	1.08	1.00	0.88	0.95	0.66	1.21	1.26	1.19	0.23	0.95	1.33	0.13	0.84
4,2,34.67	1.06	0.94	1.08	1.00	0.88	0.95	0.66	1.21	1.10	1.19	0.23	0.78	1.33	0.13	0.19
5,1,44.20	0.23	0.66	1.08	0.89	0.69	0.95	0.66	1.21	1.26	1.19	0.23	0.94	1.33	0.13	0.74
5,2,37.07	1.06	0.66	1.08	0.89	0.69	0.95	0.66	1.21	1.10	1.19	0.23	0.76	1.33	0.13	0.55
6,1,54.90	0.23	0.66	0.99	0.71	0.61	0.71	0.66	1.21	0.76	0.79	0.69	0.63	0.60	0.13	0.19
6,2,39.47	0.23	0.66	1.08	0.89	0.69	0.95	0.66	1.22	1.10	1.19	0.23	0.76	1.33	0.13	0.19

Table 4.16: Result of Problem 4 (part 2)

Cells, Layout, Moves	Workstation in cell Ci $i = 1, 2, 3, ..., N$						Parts in cell Ci $i = 1, 2, 3, ..., N$					
	C1	C2	C3	C4	C5	C6	C1	C2	C3	C4	C5	C6
4,1, 38.80	1	2	3-14	15			9,10,12	8,14,27	1-7,11, 13,15-21, 23,25,26, 28,29,30	22,24		
4,2, 36.34	2	1	3-14	15			8,14,27	9,10,12	1-7,11 13,15-21, 23,25,26, 28,29,30	22,24		
5,1, 48.70	2	15	4-14	3	1		8,14,27	22,24	1-4,6, 7,11,13, 15-21,23, 25,26 28,30	5,29	9,10,12	
5,2, 43.77	5,6,7, 8,9,10, 12,13	15	14	4,11	1,2,3		1,3,4, 6,7,11, 13,15-20, 26,28,30	22,24	2	21,23,25	5,8,9, 10,12, 14,27, 29	
6,1, 59.70	2	3	1,6	4,5,7, 8-13	15	14	8,14,27	5,29	1,9,10, 12,15,20	3,4,6, 7,11,13, 16-19,21, 23,25,26, 28,30	22,24	2
6,2, 42.16	1	13	15	14	3-12	2	9.10.12	4.17	22.24	2	1.3.5. 6.7.11. 13.15.16. 18-21.23 25.26. 26.30	8.14. 27

Table 4.17: Gupta's solution for Problem 4 (part 1). Reproduced from Gupta et al. (1996) [29]

Cells, Layout, Moves	Utilization of workstation Wi $i = 1, 2, 3, ..., M$														
	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
4,1,38.80	0.69	0.66	1.07	0.71	0.67	1.19	0.66	1.21	0.84	0.80	0.69	0.39	1.11	0.52	0.42
4,2,36.34	0.69	0.66	1.07	0.71	0.67	1.19	0.66	1.21	0.84	0.80	0.69	0.39	1.11	0.52	0.42
5,1,48.70	0.69	0.66	1.07	0.71	0.67	1.19	0.66	1.21	0.84	0.80	0.69	0.39	1.11	0.52	0.42
5,2,43.77	0.84	0.82	0.44	0.42	0.52	1.05	0.64	0.94	0.84	0.80	0.44	0.25	1.02	0.20	0.42
6,1,59.70	0.69	0.66	0.44	0.57	0.60	0.64	0.62	0.99	0.84	0.77	0.60	0.39	1.11	0.20	0.42
6,2,42.16	0.69	0.66	0.97	0.71	0.50	1.05	0.50	1.06	0.84	0.80	0.64	0.39	0.32	0.20	0.42

Table 4.18: Gupta's solution to Problem 4 (part 2). Reproduced from Gupta et al. (1996) [29]

No. of	Total moves		Average utliization	
cells, Layout	Gupta	GA	Gupta	GA
4,1[a]	38.80	37.30	0.523	0.581
4,2[a]	36.34	36.47	0.523	0.553
5,1[a]	48.70	44.20	0.501	0.549
5,2[a]	43.77	37.07	0.418	0.547
6,1[a]	59.70	54.90	0.431	0.440
6,2[a]	42.16	39.47	0.437	0.495

[a]Our approach is better

Table 4.19: Comparison of different approaches for Problem 4

maximization of the average cell utilization, minimization of intracell load imbalances, and minimization of intercell load imbalances. The overall objective function for maximization is a weighted additive utility function comprised of the four optimization objectives. With different constraints, the decision maker can set different weights to each objective. A linear integer programming enumeration scheme was used to solve the model. One problem with this approach is that the cost to find a solution to a large problem is very high.

In our case, we consider another objective which is to minimize the within cell load variation. This aids smooth flow of materials inside each cell and reduces the Work-In-Process (WIP) within it [83]. Equation 4.4 represents the mathematical formulation of this objective.

$$\textit{Cell Load Variation} = \sum_{i=1}^{m}\sum_{l=1}^{c} x_{il} \sum_{j=1}^{p}(w_{ij} - m_{lj})^2 \tag{4.4}$$

where

m = the total number of machines

c = the total number of cells

p = the total number of parts

$W = [w_{ij}]$ is an m x p machine-part incidence matrix, where w_{ij} is workload on machine i induced by part j and is equal to $(t_{ij} * N_j)/T_i$

t_{ij} = the processing time (hour/piece) of part j on machine i

T_i = the available time on machine i in a given period of time

N_j = the production requirement of part j in a given period of time

$X = [x_{il}]$ is an m x c cell membership matrix, where

$$x_{il} = \begin{cases} 1 \text{ if } ith \text{ machine is in cell } l \\ 0 \text{ otherwise} \end{cases}$$

$M = [m_{lj}]$ is a c x p matrix of average cell load, where $m_{lj} = \frac{\sum_{i=1}^{m} x_{il} * wij}{\sum_{i=1}^{m} x_{il}}$

A common problem to cellular manufacturing systems is the workload imbalance. Minimization of cell load variation helps to solve this problem. Therefore the objectives of our model is similar to Wei and Gaither's.

Fonseca and Fleming [24] pointed out that genetic algorithms searching from a population of points seemed particularly suited to multiobjective optimization. We apply the GA we developed to solve this bi-criteria model. The advantage of using these two objectives (minimize the total moves and minimize the cell load variations) is that comparable data are available in the literature [29]. This can demonstrate the performance of our algorithm and the versatility in adding different objective functions.

4.5.1 Experimental results

As suggested by Gupta et al. [29], the objectives could not be simply added up together to form the final objective functions. Instead, the two objective functions were evaluated separately. Common strings (same cellular configuration) were selected as a solution to the bi-objective model. The common chromosome structure meant that this individual was an acceptable solution to both objectives. Finally, a list of pair (value of objective 1, value of objective 2) was obtained and the ultimate solution selected was subjected to the decision maker.

In this experiment, we again consider the three problems identified by Logendran and the large problem proposed by Gupta. Tables 4.20 and 4.21 present the results for problems 1-4. We can see each solution is a pair of values. It is difficult to have a direct comparison with Gupta's solution as no single solution was obtained. However, in some cases, our approach generated solutions with smaller number of total moves provided that the cell load variations were the same. For example, for problem 4, if there were 4 cells arranged in layout 2, the total moves was 37.30 which was smaller than 42.64 given by Gupta's approach (the cell load variation was 1.60).

4.6 Conclusions

In this chapter, we proposed a multichromosome representation for cell formation problem. This representation has the advantages that no illegal

	Problem 1				Problem 2				Problem 3			
#	Layout 1		Layout 2		Layout 1		Layout 2		Layout 1		Layout 2	
of cells	Total moves	Cell load	Total moves	Cell load	Total moves	Cell load	Total moves	Cell load	Total moves	Cell load	Total moves	Cell load
2	3.8	0.72	3.8	0.72	6.3	1.92	6.7	1.84	3.5	0.38	3.5	0.38
	4.2	0.69	4.2	0.69	7.5	1.37	7.5	1.37	3.9	0.54	4.3	0.52
3	5.0	0.43	5.0	0.43	7.5	1.59	8.7	1.12	4.3	0.21	4.3	0.21
	5.4	0.19	5.4	0.19	7.9	1.47	10.5	0.89	4.7	0.33	4.7	0.33
4	6.9	0.06	6.9	0.06	9.4	1.26	9.4	1.25	5.5	0.17	5.5	0.17
	7.3	0.46	7.6	0.13	10.60	0.84	12.7	0.64	6.5	0.04	6.5	0.04

Table 4.20: Results of bi-criteria for Problem 1-3

	Problem 4			
Number	Layout 1		Layout 2	
of cells	Total moves	Cell load	Total moves	Cell load
4	37.30	1.60	37.30	1.60
	38.50	1.62	40.00	1.50
	39.20	1.52	40.80	1.64
	40.20	1.50	41.90	1.58
5	44.30	1.45	45.59	1.49
	46.20	1.42	48.60	1.36
	47.00	1.40	49.50	1.38
	47.30	1.32	50.20	1.32
6	59.80	1.25	59.80	1.25
	61.60	1.23	60.60	1.31
	62.50	1.22	61.70	1.18
	64.80	1.20	64.80	1.19

Table 4.21: Results of bi-criteria for Problem 4

offsprings are generated which in turn reduces computational overhead, and a kind of linkage may be developed among objects which aids the searching of a genetic algorithm. This representation is considered to be more natural to the GT problem. We tested our representation on a workload model. Experimental results showed our approach generates better solutions.

Chapter 5

Integrated design of cellular manufacturing systems in the presence of alternative process plans

5.1 Introduction

In most of the literature, cell formation is performed on the basis of a given set of part routings that are assumed to be fixed. Each operation of part must be performed on a specific machine. An incidence matrix is often used to represent such relationship between machines and parts. A '1' in the incidence matrix $[a_{ij}]$ indicates the part j utilizes machine i. However, in practical environment,

each part can have more than one process plan and each operation on a part can be performed on alternative machines.

Rajamani and Aneja [69] indicated that fixing a machine for an operation did not select the machine optimally which resulted in increased manufacturing costs. Another reason for the alternative process plans is the existence of functionally similar workcentres [66]. In a functional layout, this is not a consideration because in such layout, all the functionally similar workcentres are grouped together and the parts can be routed to any such available workcentre. However, in a cellular manufacturing environment, a manufacturing cell usually consists of functionally dissimilar workcentres. Similar workcentres are likely placed in different cells. We would prefer a part route to the workcentre in the cell it is assigned. Therefore, the functionally similar workcentres must have a unique identification. This identification leads to the existence of alternative process plans. Kusiak [45] has shown that the incorporation of multiple process plans resulted in improved quality of part families and machine cells.

Rajamani and Aneja [69] presented an example to further illustrate the situation. Consider the manufacturing of a gear. The initial raw material is in the form of a bar stock, eight processing steps are required to transform the raw material into a finished gear. A different set of processing steps is identified if the raw material is in a different form, say blanks either cast or forged. Once the processing steps are identified, the process planner determines the possible sequences of processing before grouping the processing steps into operations.

The eight processing steps can be grouped into different sets as follows:

Processing steps (PS)

PS 1: Facing

PS 2: Turning

PS 3: Parting-off

PS 4: Facing

PS 5: Centring

PS 6: Drilling

PS 7: Slotting

PS 8: Gear teeth cutting

	Plan 1	Plan 2
Operation 1	PS 1,2,3	PS 1,2,3
Operation 2	PS 4,5,6	PS 4,5,6
Operation 3	PS 7	PS 7,8
Operation 4	PS 8	

Each operation in the plans can be performed on a number of compatible machines. For example, P8, the gear-teeth-cutting operation can be performed on either a milling or a gear hobbing machine if plan 1 is used. If plan 2 in which the gear-teeth-cutting and slotting operations are combined is used, it can only be performed on a milling machine.

5.1.1 Literature review

Very few studies take the alternative process plans into consideration. Kusiak [43] showed that for a single part, it was possible to generate a set of process plans. The costs of these process plans may vary largely in some cases. In another paper [45], Kusiak formulated an integer programming model to

address the presence of a set of process plans. The model developed focused on generating a better diagonal structure and ignored other factors. An illustrative example allowing the consideration of different process plans led to improved chance of getting a better diagonal structure.

Nagi et al. [66] proposed an algorithm to solve the cell formation problem in the presence of multiple part routings. The objective is to select the part routings while minimizing the inter-cell traffic. The problem is decoupled to two problems: the selection of routings and formation of cells. Nagi et al. [66] formulated the problem as a linear programming problem. The solution procedure was iterative until the part routings and manufacturing cells are obtained with minimal inter-cell traffic. However, the solution was dependent on the partition chosen initially.

Rajamani and Aneja [69] developed three integer programming models to study the effect of alternative process plans and simultaneous formation of part families and machine groups. The objective function of this study was a cost function aimed at minimizing the total investment. All the models were solved using LINDO. The weakness of this study was that some nonlinear constraints were linearized. This results in increased number of variables and constraints. The results indicated the consideration of alternative process plans and simultaneous formation of part families and machine groups led to efficient resource utilization.

Logendran et al.[57] followed the model developed by Rajamani and Aneja [69]. The objective function, however, focused on the minimization of total

annual cost evaluated as the sum of the amortized cost of machines and the operating cost of producing all parts. A major difference of the Logendran's approach was to view the cell formation problem as being divided into two phases. The first phase focused on selection of a process plan for each part and the second phase on formation of part families and machine cells. The first problem was solved with a tabu-search-based heuristic. Once the process plans were fixed, the second problem was solved by one of the known clustering algorithms.

5.1.2 Motivation

We argue that the selection of process plans and the formation of cells are interrelated. For example, the number of a specific workcentre required depends on the configuration of cells because we allow machine duplication. It is more realistic to integrate the two problems together. In this chapter, we solve the model developed by Rajamani and Aneja [69] and incorporate the objective function used by Logendran [57]. The consideration of annual operating costs provides additional information to the cell designer. The complexity of the problem is shown to be NP-hard because the three-partition problem is polynomially reducible to a special case of this problem [57]. It is not possible to use any enumerative methods even for middle size problems. Also, Rajamani and Aneja [69] indicated that the formulation of his model required heuristic techniques to solve. We propose a genetic algorithm based on the

multichromosome representation developed. The use of GAs makes the model applicable to large scale problems.

5.2 Mathematical models

The model is based on the one proposed by Rajamani and Aneja [69]. The model is subjected to the following assumptions:

1. Annual demand for each part type is known and stable over the planning horizon.
2. Available capacity of each unit of a machine types is known.
3. For each process plan, the time and cost required to perform a specific operation of a part on a machine is known.
4. The number of cells formed is known.

The planning horizon is assumed to be 1 year. The model considers two objective functions: the total investment and the annual operating costs.

5.2.1 Notation

Indexing sets

k = 1,2,...K part

m = 1,2,...M machine

p = 1,2,...P_k process plans for part k

s = 1,2,...$S(k,p)$ operations for (k,p) combination

c = 1,2,... C cell

Decision variables

N_{mc} = number of machines of type m in cell c

$$Y_{kp} = \begin{cases} 1 \text{ if part } k \text{ is manufactured using plan } p \\ 0 \text{ otherwise} \end{cases}$$

$$X_{ms}(kp) = \begin{cases} 1 \text{ if machine } m \text{ is used to perform operation } s \text{ for } (k,p) \\ \quad \text{combination} \\ 0 \text{ otherwise} \end{cases}$$

$$r_{kc} = \begin{cases} 1 \text{ if part } k \text{ is a member of cell } c \\ 0 \text{ otherwise} \end{cases}$$

Coefficients

$c_{ms}(kp)$ = operating cost for machine m performing operations for (k,p) combination

$t_{ms}(kp)$ = time for machine m to perform operation s for (k,p) combination

$$a_s(kp) = \begin{cases} 1 \text{ if operation } s \text{ has to be performed for the } (k,p) \text{ combination} \\ 0 \text{ otherwise} \end{cases}$$

$$\alpha_{ms} = \begin{cases} 1 \text{ if machine } m \text{ can perfrom operation } s \\ 0 \text{ otherwise} \end{cases}$$

C_m = cost per machine of type m

b_m = time available on each machine of type m

d_k = demand for part k

5.2.2 Objective functions

The first objective function of the model is to minimize **capital investment**:

$$f_1(x) = \sum_{mc} C_m N_{mc} \tag{5.1}$$

The second objective function of the model is to minimize **total annual cost** which consists of two parts:

The total amortized cost of machine equals

$$\sum_{mc} C_m N_{mc}$$

The total annual operating cost for processing all parts equals

$$\sum_{kpms} d_k X_{ms}(kp) C_{ms}(kp)$$

Thus, the model can be represented as follows:

Minimize:

$$f_2(x) = \sum_{mc} C_m N_{mc} + \sum_{kpms} d_k X_{ms}(kp) C_{ms}(kp) \tag{5.2}$$

both objective functions are subjected to

$$\sum_{p} Y_{kp} = 1 \qquad \forall k \tag{5.3}$$

$$\sum_{m} \alpha_{ms} X_{ms}(kp) = a_s(kp) Y_{kp} \qquad \forall s, k, p \tag{5.4}$$

$$\sum_{kps} (r_{kc} d_k) X_{ms}(kp) t_{ms}(kp) \leq b_m N_{mc} \qquad \forall m, c \tag{5.5}$$

$$\sum_{c} r_{kc} = 1 \qquad \forall k \tag{5.6}$$

$$N_{mc} \geq 0 \text{ and integer} \qquad \forall m, c \tag{5.7}$$

$$Y_{kp} = (0, 1) \qquad \forall k, p \tag{5.8}$$

$$X_{ms}(kp) = (0, 1) \qquad \forall m, s, k, p \tag{5.9}$$

The constraints of the model are given by (5.3) - (5.9). Constraint (5.3) guarantees that only one process plan is selected for a given part. Constraint (5.4) ensures that an operation in the selected process plan is performed on one of the available machines. Constraint (5.5) ensures that the capacity of each machine type is not violated. Constraint (5.6) guarantees that a part only belongs to one cell. Constraints (5.7) - (5.9) indicate the 0, 1 and integer variables.

5.3 Our solution

To solve this problem, the algorithm determines the part families while minimizing the objective functions. The number of specific machines are determined according to the workload and the capacity of the machines. In this model, we duplicate machines to completely eliminate intercell moves.

In the previous chapter, we introduced a multichromosome genetic algorithm that is able to capture more information than the traditional approach. In addition to the part sequence chromosome and the cell boundary chromosome, we add one more to represent the process plan selected for each part. That means an individual in our genetic algorithm consists of three chromosomes. The characteristics of the chromosomes are presented in Table 5.1, where n is

the number of parts.

Chromosome	length	crossover used
part sequence	n	edge-recombination operator
cell boundary	$n-1$	edge-recombination operator
process plan	n	simple crossover operator

Table 5.1: The three chrosmosomes of an individual

The genes in the part sequence and cell boundary chromosomes are integer numbers. The interpretation is the same as in the previous chapter. In the part sequence chromosome, the integer number of a gene represents the corresponding part. In the cell boundary chromosome, if the number of cells formed is c, then the integer values smaller than c represents the cell boundaries. For the process plan chromosome, the range of each gene indicates the possible process plans of a part. For instance, if part 1 has 8 process plans to be selected then the range of the first gene is 1-8. The operator used for this chromosome is the simple crossover operator. For the part sequence and cell boundary chromosomes, edge recombination operator is employed. This operator can successfully transfer the information stored in the parent chromosomes to the offsprings. Figure 5.1 is a possible individual assuming three cells to be formed.

In this example, process plan 6 will be selected for part 1 and process plan 5 for part 2, and so on. Representing different characteristics of the solution enables the GA to sample the search space more productively. The GA can explore the partitions that explicitly represent different features of the problem [35]. Therefore, it is a more natural and direct representation of this problem.

Although the details of the operation of genetic algorithms are discussed

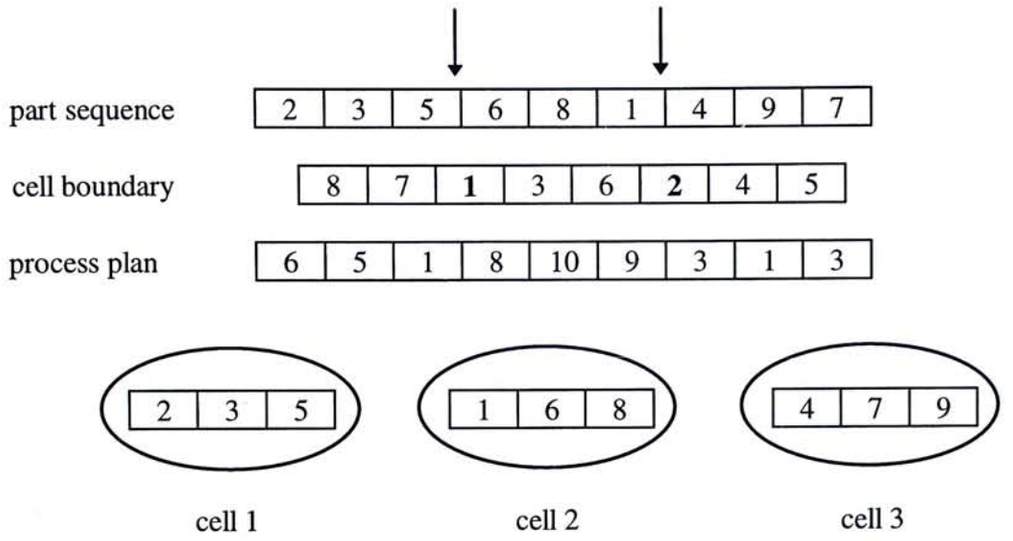

Figure 5.1: A possible individual of three chromosomes

in the previous chapter, we include a brief discussion here to complete our algorithm. The initial individuals are generated randomly. Parents are selected using fitness ranking. Crossover is then performed on the two parents, each chromosome will crossover with the corresponding chromosome in the other individual. The crossover operators used are listed in Table 5.1. These processes are repeated until a predefined number of generation is reached. The individual with the lowest fitness value (we are solving a minimization problem) becomes the solution.

5.4 Illustrative example and analysis of results

We consider an example previously presented by Rajamani and Aneja [69] to illustrate the capability of our algorithm. The problem has four parts and

three machines, and the corresponding data are presented in Tables 5.2-5.4. Consider the part 4 in Table 5.2 which has two plans. Operations 1, 2, and 3 are required for plan 1 and only operations 1 and 2 for plan 2. Table 5.3 indicates that operation 1 can be performed on either machine 1 (M1) or M3, operation 2 can be performed on M2 or M3. There are totally eight different ways that P4 can be processed using plan 1. They are M1, M2, M1 or M1, M3, M1, or M1, M2, M2, or M1, M3, M2, or M3, M2, M1, or M3, M2, M2, or M3, M3, M1, or M3, M3, M2. Similarly, there are four different ways that P4 can be processed by the second plan. So the value of 4th gene of the process plan chromosome ranging from 1 to 12. There is a one-to-one mapping for the process plan and the gene value. For the rest of parts, there are 8 process plans for part 1, 12 process plans for part 2 and part 3 has 16 process plans. The initialization of process plan chromosome depends on these values. The simple crossover operator exchanges the information stored in both parents and will not generate any illegal offsprings (see Figure 5.2).

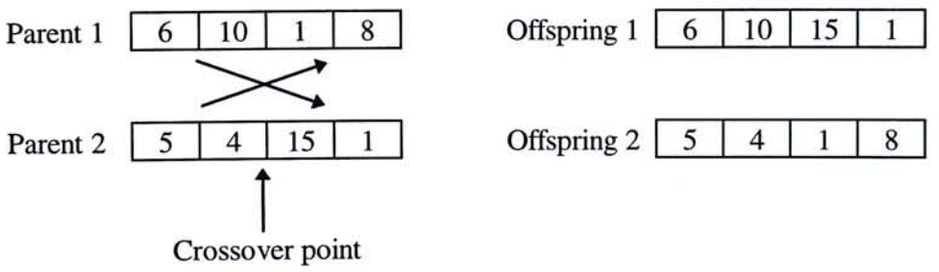

Figure 5.2: Crossover of the process plan chromosome

	$k=1$		$k=2$		$k=3$			$k=4$	
Operation	$p=1$	$p=2$	$p=1$	$p=2$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$
$s=1$	1		1		1		1	1	1
$s=2$	1	1	1	1	1	1	1	1	1
$s=3$		1	1	1		1	1	1	
Demand	10		10		10			10	

Table 5.2: Data on $a_s(kp)$ indicating if operation s of part k to be performed for the process plan p, and the demand d_k for part k

	Machine		
	$m=1$	$m=2$	$m=3$
$s=1$	1		1
$s=2$		1	1
$s=3$	1	1	
Capacity	100	100	100
Cost	100	250	300

Table 5.3: Data on α_{ms} indicating if operation s can be performed on machine m; capacity (b_m) on machine m; and the cost (C_m) of machine m

	$t_{ms}(kp), c_{ms}(kp)$								
	$k=1$		$k=2$		$k=3$			$k=4$	
	$p=1$	$p=2$	$p=1$	$p=2$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$
$s=1, m=1$	5,3		3,4		2,2		8,1	1,2	9,7
$s=1, m=3$	7,2		4,3		2,2		9,2	2,1	8,9
$s=2, m=2$	3,5	9,8	7,8	3,3	3,3	1,2	5,9	2,3	9,8
$s=2, m=3$	4,3	7,9	7,7	2,3	4,4	2,4	3,10	2,4	10,9
$s=3, m=1$		8,8	10,9	6,5		11,7	7,4	3,5	
$s=3, m=2$		7,7	8,9	6,6		8,8	9,5	2,6	

Table 5.4: The processing time $t_{ms}(kp)$ and operating cost $c_{ms}(kp)$ required for machine m to perform operation s on part k using process plan p

5.4.1 Solution for objective function 1

The objective function 1 includes only the machine costs. The result is tabulated in Table 5.5-5.7. Table 5.5 indicates the process plan selected for each part. For example, part 1 uses plan 1, in which operation 1 is performed on machine 1 and operation 2 is performed on machine 2. Table 5.6 indicates the part families formed, parts 1, 3, and 4 are in cell 2 and part 2 is in cell 1. Table 5.7 presents the number of machines required for each cell. A machine 2 is required for cell 1. One machine 1 and one machine 2 are allocated to cell 2. The total cost is 600. This solution is known to be optimal for this data set as indicated in Rajamani and Aneja [69].

	$k=1$	$k=2$	$k=3$	$k=4$
	$p=1$	$p=2$	$p=1$	$p=1$
$s=1$	$m=1$		$m=1$	$m=1$
$s=2$	$m=2$	$m=2$	$m=2$	$m=2$
$s=3$		$m=2$		$m=2$

Table 5.5: Indicates the plan selected p and machine selected m for operation s

	Part			
Cell	$k=1$	$k=2$	$k=3$	$k=4$
$c=1$		1		
$c=2$	1		1	1

Table 5.6: Data on r_{kc} indicating if part k is a member of cell c

	Cell	
	$c=1$	$c=2$
$m=1$	0	1
$m=2$	1	1
$m=3$	0	0

Table 5.7: Optimum number of each machine type m assigned to each cell c

5.4.2 Solution for objective function 2

The objective function 2 includes the amortized cost and the operating cost. The amortized cost is 600, the operating cost is 330, and the total annual cost is 930. Table 5.8-5.10 present the results. The cell configuration generated is the same as that using objective function 1. We cannot have a direct comparison because Logendran did not take the formation of cells into account.

	$k=1$	$k=2$	$k=3$	$k=4$
	$p=1$	$p=2$	$p=1$	$p=1$
$s=1$	$m=1$		$m=1$	$m=1$
$s=2$	$m=2$	$m=2$	$m=2$	$m=2$
$s=3$		$m=2$		$m=2$

Table 5.8: Indicates the plan selected p and machine selected m for operation s

	Part			
Cell	$k=1$	$k=2$	$k=3$	$k=4$
$c=1$		1		
$c=2$	1		1	1

Table 5.9: Data on r_{kc} indicating if part k is a member of cell c

	Cell	
	$c = 1$	$c = 2$
$m = 1$	0	1
$m = 2$	1	1
$m = 3$	0	0

Table 5.10: Number of each machine type m assigned to each cell c

5.5 Conclusions

Including alternative process plans makes the cell formation problem more complicated An exhaustive search technique is not possible even for moderate number of parts and process plans. The use of genetic algorithm seems to have a potential to tackle large scale problems and is a good heuristic method to solve the integrated model.

Chapter 6

Conclusions

6.1 Summary of achievements

In this research, we developed heuristic approaches to address various aspects of the group technology problem. From standard models that utilize a binary machine-part matrix to generalized models that incorporate different objectives and constraints. The cell formation problem is difficult to solve and therefore, requires heuristic methods. There are extensive algorithms available in the literature. However, the practical application of many methods is limited by the problem scale. Also most of the methods only address one or a few aspects of the problem.

Genetic algorithm (GA) is an optimization technique with great versatility and extensibility. These features of GA lie in its ability to substitute different representations and evaluation functions. In designing a cellular manufacturing system, the cell designer needs to consider many factors, for example, the inter-

and intra-cell moves, the workload and the capacity of workcentres, the workload variation of cells, and the possible processing plans of parts. In different situations, the constraints faced by the cell designer are different. Also, a firm has its own parameter settings in designing a cellular manufacturing system. These can partially explain why so many models have been proposed in the literature. GA seems particularly suitable for the cell formation problem as the objective functions can be easily changed.

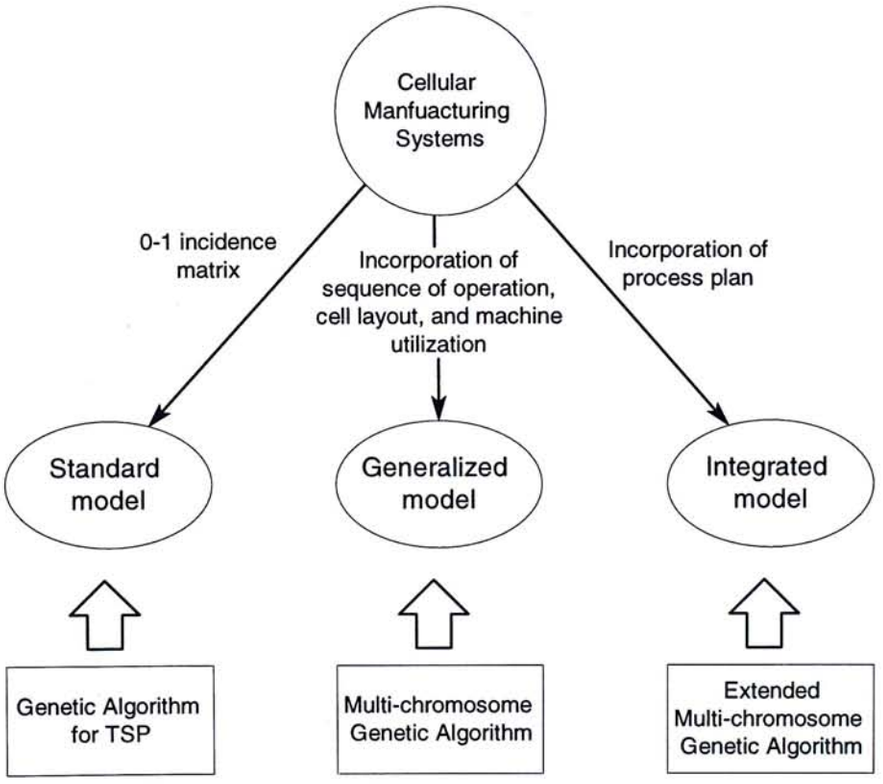

Figure 6.1: Overview of the research

Figure 6.1 is a overview of the research. In solving the standard model, the problem is formulated as a traveling salesman problem and a genetic algorithm-based approach is employed. The solutions obtained are of better quality in

terms of grouping efficiency and grouping efficacy. The improvements are more significant for larger scale problems.

In solving generalized models, a different representation scheme is required. We introduce a multiple chromosomes representation. In this representation, the structure of the problem is captured. Our approach generates better solution in a workload-based model.

To further illustrate the extensibility of the developed genetic algorithm, we consider a more practical aspect of group technology and incorporate alternative process plans. Experimental results indicate that our GA approach is satisfactory and that this solution technique has a potential to tackle large size problems.

6.2 Future works

There are some possible future works on this project. In the whole research, we employ existing crossover operators. We can develop new crossover operators that tailor for the GT problem. These operators can increase the peformance. Also, we can incorporate existing clustering heuristic into the GAs to produce hybrid approaches. Existing heuristic can provide additional information and such hybrid techniques may further improve the results. One possible way is to seed the initial population of GAs with solutions from other clustering techniques. Furthermore, we can try to apply the techniques to other models. In this research, a workload model is selected. Actually, there are other models

exist, e.g. cost model. We can have more experiments to further demonstrate the applicability of GAs to GT problem.

Bibliography

[1] Askin, R. G., and Chiu, K. A graph partitioning procedure for machine assignment and cell formation. *International Journal of Production Research 28* (1990), 1555–1572.

[2] Askin, R. G., and Standridge, C. R. *Modeling and analysis of manufacturing systems.* John Wiley and Sons, New York, 1993.

[3] Askin, R. G., and Subramanian, S. P. A cost-based heuristic for group technology configuration. *International Journal of Production Research 25* (1987), 101–113.

[4] Ballakur, A., and Steudel, H. J. A within-cell utilization based heuristic for designing cellular manufacturing systems. *International Journal of Production Research 25* (1987), 639–665.

[5] Boctor, F. F. A linear formulation of the machine-part cell formation problem. *International Journal of Production Research 29* (1991), 343–356.

[6] Boe, W. J., and Cheng, C. H. A close neighbour algorithm for a designing cellular manufacturing system. *International Journal of Production Research 29* (1991), 2097–2116.

[7] Bruns, R. Direct chromosome representation and advanced genetic operators for production scheduling. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (San Mateo, California, 1993), Morgan Kaufmann Publishers, pp. 352–359.

[8] Burbidge, J. L. A manual method of production flow analysis. *The Production Engineer 56* (1977), 34–38.

[9] Carrie, A. S. Numerical taxonomy applied to group technology and plant layout. *International Journal of Production Research 11* (1973), 399–416.

[10] Chan, H. M., and Milner, D. A. Direct clustering algorithm for group formation in cellular manufacturing. *Journal of Manufacturing Systems 1* (1982), 65–74.

[11] Chandrasekharan, M. P., and Rajagopalan, R. An ideal seed nonhierarchical clustering algorithm for cellular manufacturing. *International Journal of Production Research 24* (1986), 451–464.

[12] Chandrasekharan, M. P., and Rajagopalan, R. Modroc-an extension of rank order clustering for group technology. *International Journal of Production Research 24* (1986), 1221–1233.

[13] Chandrasekharan, M. P., and Rajagopalan, R. Zodiac - an algorithm for concurrent formation of part-families and machine-cells. *International Journal of Production Research 25* (1987), 835–850.

[14] Chandrasekharan, M. P., and Rajagopalan, R. Groupability: an analysis of the properties of binary data matrices for group technology. *International Journal of Production Research 27* (1989), 1035–1052.

[15] Chen, C. L., Cotruvo, N. A., and Baek, W. A simulated annealing solution to cell formation problem. *International Journal of Production Research 33* (1995), 2601–2614.

[16] Chen, S. J., and Cheng, C. S. A neural network-based cell formation algorithm in cellular manufacturing. *International Journal of Production Research 33* (1995), 293–318.

[17] Cheng, C. H. Algorithms for grouping machine groups in group technology. *Omega 20* (1992), 493–501.

[18] Cheng, C. H., Kumar, A., and Motwani, J. A comparative examination of selected cellular manufacturing clustering algorithms. *International Journal of Operations and Production Management 15*, 12 (1995), 86–97.

[19] Cheng, C. H., Kusiak, A., and Boe, J. W. A branch-and-bound algorithm for solving the machine allocation problem. In *Advances in Manufacturing and Automation Systems*, C. T. Leondes, Ed. Academic Press, New York, 1991.

[20] Chu, C. H., and Tsai, M. A comparison of three array-based clustering techniques for manufacturing cell formation. *International Journal of Production Research 28* (1990), 1417–1433.

[21] Davis, L. Job shop scheduling with genetic algorithms. In *Proceedings of the First International Conference on Genetic Algrorithms* (San Mateo, California, 1985), Morgan Kaufmann Publishers, pp. 136–140.

[22] De Witte, J. The use of similarity coefficients in production flow analysis. *International Journal of Production Research 18* (1980), 503–514.

[23] Del Valle, A. G., Balarezo, S., and Tejero, J. A heuristic workload-based model to form cells by minimizing intercellular movements. *International Journal of Production Research 32* (1994), 2275–2285.

[24] Fonseca, C. M., and Fleming, P. J. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (San Mateo, California, 1993), Morgan Kaufmann Publishers, pp. 416–423.

[25] Glover, F. Artificial intelligence, heuristic frameworks and tabu search. *Managerial and decision economics 11* (1990).

[26] Goldberg, D. E. *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley, Reading, MA, 1989.

[27] Goldberg, D. E., and Lingle, R. Alleles loci and the tsp. In *Proceeding of the First International Conference on Genetic Algorithms* (San Mateo, California, 1985), Morgan Kaufmann Publishers, pp. 154–159.

[28] Gunasingh, R. K., and Lashkari, R. S. Machine grouping problem in cellular manufacturing systems - an integer programming approach. *International Journal of Production Research 27* (1989), 1465–1473.

[29] Gupta, Y. P., Gupta, M. C., Kumar, A., and Sundaram, C. A genetic algorithm-based approach to cell composition and layout design problems. *International Journal of Production Research 34* (1996), 447–482.

[30] Gupta, Y. P., Gupta, M. C., Kumar, A., and Sundram, C. Minimizing total intercell and intracell moves in cellular manufacturing: a genetic algorithm approach. *International Journal of Computer Intergrated Manufacturing 8* (1995), 92–101.

[31] Harhalakis, G., Proth, J. M., and Xie, X. L. Manufacturing cell design using simulated annealing: an industrial application. *Journal of Intelligent Manufacturing 1* (1990), 185–191.

[32] Heragu, S. S., and Gupta, Y. P. A heuristic for designing cellular manufacturing facilities. *International Journal of Production Research 32* (1994), 125–140.

[33] Holland, J. H. *Adaptation in natural and artificial systems.* Ann Arbor: The University of Michigan Press, 1975.

[34] Hyer, N. L., and Wemmerlov, U. Group technology in the us manfacturing industry: a survey of current practices. *International Journal of Production Research 27* (1989), 1287–1304.

[35] Juliff, K. A multi-chromosome genetic algorithm for pallet loading. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (San Mateo, California, 1993), Morgan Kaufmann Publishers, pp. 467–473.

[36] Karparthi, S., and Suresh, N. C. Machine-component cell formation in group technology: a neural network approach. *International Journal of Production Research 25* (1992), 1353–1367.

[37] King, J. R. Machine-component group formation in production flow analysis: An approach using a rank order clustering algorithm. *International Journal of Production Research 18* (1980), 213–232.

[38] King, J. R., and Nakornchai, V. Machine-component group formation in group technology: review and extension. *International Journal of Production Research 20* (1982), 117–133.

[39] Kirkpatrick, S., Jr Gelatt, C. D., and Vecchi, M. P. Configuration space analysis of traveling salesman problems. *J. Physique 46* (1985), 1277–1292.

[40] Kumar, C. S., and Chandrasekharan, M. P. Grouping efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. *International Journal of Production Research 28* (1990), 233–243.

[41] Kumar, K. R., Kusiak, A., and Vannelli, A. Grouping of parts and components in flexible manufacturing systems. *European Journal of Operational Research 24* (1986), 387–397.

[42] Kumar, K. R., and Vannelli, A. Strategic subcontracting for efficient disaggregated manufacturing. *International Journal of Production Research 25* (1987), 1715–1728.

[43] Kusiak, A. Integer programming approach to process planning. *International Journal of Advanced Manufacturing Technology 1* (1985), 73.

[44] Kusiak, A. The part-families problem in flexible manufacturing systems. *Annals of Operational Research 3* (1985), 279–300.

[45] Kusiak, A. The generalized group technology concept. *International Journal of Production Research 25* (1987), 561–569.

[46] Kusiak, A. Branching algorithms for solving the group technology problem. *Journal of Manufacturing Systems 10* (1991), 332–343.

[47] Kusiak, A., Boe, W. J., and Cheng, C. H. Designing cellular manufacturing systems: branch-and-bound and a* approaches. *IIE Transactions 25* (1993), 46–56.

[48] Kusiak, A., and Cheng, C. H. A branch-and-bound algorithm for solving the group technology problem. *Annual of Operations Research 26* (1990), 415–431.

[49] Kusiak, A., and Cho, M. Similarity coefficient algorithm for solving the group technology problem. *International Journal of Production Research 30* (1992), 2633–2646.

[50] Kusiak, A., and Chow, W. S. Efficient solving of the group technology problem. *Journal of manufacturing systems 6* (1987), 117–124.

[51] Lawler, E. L., Lenstra, J. K., Rinnooy, A. H. G. K., and Shmoys, D. B. *The Traveling Salesman problem: A guided Tour of Combinatorial Optimization.* Wiley, New York, 1985.

[52] Lenstra, J. K., and Rinnooy, A. H. G. K. Some simple applications of the travelling salesman problem. *Operations Research Quarterly 26* (1975), 717–733.

[53] Levine, D. M. A genetic algorithm for the set partitioning problem. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (San Mateo, California, 1993), Morgan Kaufmann Publishers, pp. 481–487.

[54] Lin, S., and Kernighan, B. W. An effective heuristic algorithms for traveling salesman problem. *Operation Research* (1973), 498–516.

[55] Logendran, R. A workload based model for minimizing total intercell and intracell moves in cellular manufacturing. *International Journal of Production Reserach 28* (1990), 913–925.

[56] Logendran, R. Impact of sequence of operations and layout of cells in cellular manufacturing. *International Journal of Production Reserach 29* (1991), 375–390.

[57] Logendran, R., Ramakrishna, P., and Sriskandarajah, C. Tabu search-based heuristics for cellular manufacturing systems in the presence of alternative process plans. *International Journal of Production Research 32* (1994), 273–297.

[58] Martin, O., Otto, S. W., and Felten, E. W. Large-step markov chains for the traveling salesman problem. *complex Systems 2* (1991), 299–326.

[59] McAuley, A. Machine grouping for efficient production. *Production Engineering 51* (1972), 53–57.

[60] McCormick, W. T., Jr. Schweitzer, P. J., and White, T. W. Problem decomposition and data reorganization by a clustering technique. *Operations Research 20* (1972), 993–1009.

[61] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Hong Kong, 1992.

[62] Miliotis, P. Using cutting planes to solve the symmertic travelling salesman problem. *Mathmatical Programming 15* (1978), 177–188.

[63] Miltenburg, J., and Zhang, W. A comparative evalution of nine well-known algorithms for solving the cell formation in group technology. *Journal of Operations Management 10* (1991), 44–72.

[64] Mosier, C., and Taube, L. The facets of group technology and their impacts on implementation. *Omega 13* (1985), 381–391.

[65] Mosier, C., and Taube, L. Weighted similarity measure heuristics for the group technology clustering problem. *Omega 13* (1985), 577–579.

[66] Nagi, R., Harhalakis, G., and Proth, J. M. Multiple routings and capacity considerations in group technology applications. *International Journal of Production Research 28* (1990), 2243–2257.

[67] Oliver, I. M., Smith, D. J., and Holland, J. R. C. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms* (San Mateo, California, 1987), Morgan Kaufmann Publishers, pp. 224–230.

[68] Padberg, M., and Giovanni, R. A branch-and-cut algorithm for the resolution of large-scale symmetric travelling salesman problem. *SIAM Review 33* (1991), 60–100.

[69] Rajamani, S. N., and Aneja, Y. P. Integrated design of cellular manufacturing systems in the presence of alternative process plans. *International Journal of Production Research 28* (1990), 1541–1554.

[70] Seifoddini, H. A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications. *International Journal of Production Research 27* (1989), 1161–1165.

[71] Seifoddini, H., and Wolfe, P. M. Application of the similarity coefficient method in group technology. *IIE Transactions 18* (1986), 271–277.

[72] Shafer, S. M., and Rogers, D. F. Similarity and distance measures for cellular manufacturing part i: a survey. *International Journal of Production Research 31* (1993), 1133–1142.

[73] Shargal, M., Shekhar, S., and Irani, S. A. Evaluation of search algorithms and clustering efficiency measures for machine-part matrix clustering. *IIE Transactions 27* (1995), 43–59.

[74] Shirrish, B., Nigel, J., and Kabuka, M. R. A boolean neural network approach for the travelling salesman problem. *IEEE Transaction on Computers 42* (1993), 1271–1278.

[75] Srinivasan, G. A clustering algorithm for machine cell formation in group technology using minimum spanning trees. *International Journal of Production Research 32* (1994), 2149–2158.

[76] Srinivasan, G., and Narendran, T. T. Grafics - a nonhierarchical clustering algorithm for group technology. *International Journal of Production Research 29* (1991), 463–478.

[77] Stanfel, L. E. Machine clustering for economic production. *Engineering Costs and Production Economics 9* (1985), 73–81.

[78] Starkweather, T., McDaniel, S., Mathias, K., Whitley, D., and Whitley, C. A comparison of genetic sequencing operators. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (San Mateo, California, 1991), Morgan Kaufmann Publishers, pp. 69–76.

[79] Sule, D. R. Machine capacity planning in group technology. *International Journal of Production Research 29* (1991), 1909–1922.

[80] Suresh, N. C., Slomp, J., and Kaparthi, S. The capacitated cell formation problem: a new hierarchical methodology. *International Journal of Production Research 33* (1995), 1761–1784.

[81] Syswerda, G. Schedule optimization using genetic algorithms. In *Handbook of Genetic Algorithms. 1* (1990).

[82] Tabucanon, M. T., and Ojha, R. Icrma - a heuristic approach for intercell flow reduction in cellular manufacturing systems. *Material Flow 4* (1987), 189.

[83] Venugopal, V., and Narendran, T. T. A genetic algorithm approach to the machine-component grouping problem with multiple objectives. *Computers Industrial Engineering 22* (1992), 469–480.

[84] Waghodekar, P. H., and Sahu, S. Machine-component cell formation in group technology mace. *International Journal of Production Research 22* (1984), 937–948.

[85] Wei, J. C., and Gaither, N. A capacity constrained multiobjective cell formation method. *Journal of manufacturing systems 9* (1990), 222–232.

[86] Whitley, D. The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms* (San Mateo, California, 1989), Morgan Kaufmann Publishers, pp. 116–121.

[87] Whitley, D., Starkweather, T., and Fuquay, D. A. Scheduling problems and travelling salesman: the genetic edge recombination operator. In *Proceedings of the Third International Conference on Genetic Algorithms* (San Mateo, California, 1989), Morgan Kaufmann Publishers, pp. 133–140.

CUHK Libraries
003510817